MAX78002 User Guide

UGxxxx; Rev a1; 4/2022

Abstract: This user guide provides application developers information on how to use the memory and peripherals of the MAX78002 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.

MAX78002 User Guide

Table of Contents

MAX78002 User Guide ---- 2

1. Introduction ---- 28

1.1 Related Documentation ---- 28

1.2 Document Conventions ---- 28

1.2.1 Number Notations ---- 28
1.2.2 Register and Field Access Definitions ---- 28
1.2.3 Register Lists ---- 29
1.2.4 Register Detail Tables ---- 29

2. Overview ---- 30

2.1 Block Diagram ---- 31

3. Memory, Register Mapping, and Access ---- 32

3.1 Memory, Register Mapping, and Access Overview ---- 32

3.2 Standard Memory Regions ---- 37

3.2.1 Code Space ---- 37
3.2.2 Internal Cache Memory ---- 37
3.2.3 Information Block Flash Memory ---- 37
3.2.4 SRAM Space ---- 38
3.2.5 Peripheral Space ---- 39
3.2.6 AES Key and Working Space Memory ---- 39
3.2.7 System Area (Private Peripheral Bus) ---- 39

3.3 AHB Interfaces ---- 39

3.3.1 Arm Core AHB Interfaces ---- 39
3.3.2 AHB Slaves ---- 40
3.3.3 AHB Slave Base Address Map ---- 40

3.4 Peripheral Register Map ---- 40

3.4.1 APB Peripheral Base Address Map ---- 40

3.5 Error Correction Coding (ECC) Module ---- 42

3.5.1 SRAM ---- 42
3.5.2 Limitations ---- 42

4. System, Power, Clocks, Reset ---- 43

4.1 Oscillator Sources ---- 43

4.1.1 120MHz Internal Primary Oscillator (IPO) ---- 43
4.1.2 60MHz Internal Secondary Oscillator (ISO) ---- 43
4.1.3 8kHz-30kHz Internal Nano-Ring Oscillator (INRO) ---- 43
4.1.4 7.3728MHz Internal Baud Rate Oscillator (IBRO) ---- 44
4.1.5 100MHz/200MHz Internal Phase Lock Loop Oscillator (IPLL) ---- 44
4.1.6 25MHz External Base Oscillator (EBO) ---- 44
4.1.7 32.768kHz External Real-Time Clock Oscillator (ERTCO) ---- 44
4.1.8 External Clock (EXT_CLK) ---- 45

4.2 System Oscillator (SYS_OSC) ---- 45

4.2.1 System Oscillator Selection ---------- 45
4.2.2 System Clock (SYS_CLK) ---------- 45
4.3 Operating Modes ---------- 48
4.3.1 ACTIVE Mode ---------- 48
4.3.2 Low-Power Modes ---------- 48
4.4 Wake-Up Sources for Each Operating Mode ---------- 60
4.5 Device Resets ---------- 60
4.5.1 Peripheral Reset ---------- 61
4.5.2 Soft Reset ---------- 61
4.5.3 System Reset ---------- 61
4.5.4 Power-On Reset ---------- 62
4.6 Unified Internal Cache Controllers ---------- 62
4.6.1 Enabling the Internal Cache Controllers ---------- 62
4.6.2 Disabling the ICC ---------- 62
4.6.3 Invalidating the ICC Cache and Tag RAM ---------- 62
4.6.4 Flushing the ICC ---------- 62
4.6.5 ICC Registers ---------- 62
4.6.6 ICC Register Details ---------- 63
4.7 RAM Memory Management ---------- 64
4.7.1 On-Chip Cache Management ---------- 64
4.7.2 RAM Zeroization ---------- 64
4.8 Miscellaneous Control Registers (MCR) ---------- 64
4.8.1 Miscellaneous Control Register Details ---------- 65
4.9 Single Inductor Multiple Output Power Supply (SIMO) ---------- 69
4.9.1 Power Supply Monitor ---------- 70
4.9.2 Single Inductor Multiple Output Registers (SIMO) ---------- 70
4.9.3 Single Inductor Multiple Output (SIMO) Registers Details ---------- 71
4.10 Low-Power General Control Registers (LPGCR) ---------- 75
4.10.1 Low-Power General Control Registers Details ---------- 75
4.11 Power Sequencer Registers (PWRSEQ) ---------- 77
4.11.1 Power Sequencer Register Details ---------- 77
4.12 Trim System Initialization Registers (TRIMSIR) ---------- 84
4.12.1 TRIM System Initialization Register Details ---------- 85
4.13 Global Control Registers (GCR) ---------- 87
4.13.1 Global Control Register Details (GCR) ---------- 87
4.14 System Initialization Registers (SIR) ---------- 105
4.14.1 System Initialization Register Details ---------- 105
4.15 Function Control Registers (FCR) ---------- 107
4.15.1 Function Control Register Details ---------- 107
4.16 General Control Function Registers (GCFR) ---------- 112
4.16.1 General Control Function Register Details ---------- 112
5. Interrupts and Exceptions ---------- 116
5.1 CM4 Interrupt and Exception Features ---------- 116
5.2 CM4 Interrupt Vector Table ---------- 116
5.3 RV32 Interrupt Vector Table ---------- 118

6. General-Purpose I/O and Alternate Function Pins (GPIO) 120

6.1 Instances 121

6.2 Configuration 121

6.2.1 Power-On-Reset Configuration 121
6.2.2 Serial Wire Debug Configuration 122
6.2.3 Pin Function Configuration 122
6.2.4 Input Mode Configuration 122
6.2.5 Output Mode Configuration 123

6.3 Reference Tables 123

6.4 Usage 124

6.4.1 Reset State 124
6.4.2 Input Mode Configuration 125
6.4.3 Output Mode Configuration 125
6.4.4 Alternate Function Configuration 125

6.5 Configuring GPIO (External) Interrupts 125

6.5.1 GPIO Interrupt Handling 126
6.5.2 Using GPIO for Wakeup from Low-Power Modes 126

6.6 Registers 127

6.6.1 GPIO Register Details 128

7. Flash Controller (FLC) 136

7.1 Instances 136

7.2 Usage 136

7.2.1 Clock Configuration 136
7.2.2 Lock Protection 137
7.2.3 Flash Write Width 137
7.2.4 Flash Write 137
7.2.5 Page Erase 138
7.2.6 Mass Erase 138

7.3 Registers 139

7.3.1 Register Details 139

8. Debug Access Port (DAP) 146

8.1 Instances 146

8.2 Access Control 146

8.2.1 Factory Disabled DAP 146
8.2.2 Software Accessible DAP 146

8.3 Pin Configuration 146

9. Semaphores 147

9.1 Instances 147

9.2 Multiprocessor Communications 147

9.2.1 Reset 147
9.2.2 CM4 Semaphore Interrupt Generation 147
9.2.3 RV32 Semaphore Interrupt Generation 147

9.3 Registers 148

9.3.1 Register Details 148

10. Standard DMA (DMA) ---- 153

10.1 Instances ---- 153

10.2 DMA Channel Operation (DMA_CH) ---- 153

10.2.1 DMA Channel Arbitration and DMA Bursts ---- 153
10.2.2 DMA Source and Destination Addressing ---- 154
10.2.3 Data Movement from Source to DMA ---- 156
10.2.4 Data Movement from DMA to Destination ---- 156

10.3 Usage ---- 157

10.4 Count-To-Zero (CTZ) Condition ---- 158

10.5 Chaining Buffers ---- 158

10.6 DMA Interrupts ---- 160

10.7 Channel Timeout Detect ---- 160

10.8 Memory-to-Memory DMA ---- 161

10.9 DMA Registers ---- 161

10.9.1 Register Details ---- 161

10.10 DMA Channel Register Summary ---- 162

10.11 DMA Channel Registers ---- 162

10.11.1 Register Details ---- 162

11. ADC ---- 168

11.1 Operation ---- 168

11.1.1 Input Channels ---- 168

11.2 Clocks and Timing ---- 169

11.3 Operating Modes ---- 171

11.3.1 ADC Initialization ---- 173

11.4 ADC SFR Interface ---- 174

11.4.1 Determination of Bias and Wake-up Counter Settings ---- 174
11.4.2 Using the ADC SFR Interface to Load the Reference Trim, and Bias/Wake-up Counter Settings ---- 175
11.4.3 1.25V Internal Reference Trim ---- 175
11.4.4 2.048V Internal Reference Trim ---- 176
11.4.5 External Reference Trim ---- 177

11.5 Interrupts ---- 177

11.6 FIFO Operation ---- 179

11.7 Averaging ---- 179

11.8 Conversion Results ---- 179

11.9 Conversions ---- 181

11.9.1 Conversion Sequence Triggers ---- 181
11.9.2 Single Conversion Sequences ---- 183
11.9.3 Continuous Conversion Sequences ---- 185

11.10 Low-Power Analog Wake-Up Comparators ---- 188

11.11 ADC Registers ---- 189

11.11.1 Register Details ---- 190

11.12 Low-Power Comparator Registers ---- 200

11.12.1 Low-Power Comparator Register Details 201
12. UART (UART) 202
12.1 Instances 203
12.2 DMA 204
12.3 UART Frame 204
12.4 FIFOs 204
12.4.1 TX FIFO Operation 204
12.4.2 RX FIFO Operation 205
12.4.3 Flushing 205
12.5 Interrupt Events 205
12.5.1 Frame Error 205
12.5.2 Parity Error 206
12.5.3 CTS Signal Change 207
12.5.4 Overrun 207
12.5.5 Receive FIFO Threshold 207
12.5.6 Transmit FIFO Half-Empty 207
12.6 LPUART Wakeup Events 207
12.6.1 Receive FIFO Threshold 207
12.6.2 Receive FIFO Full 207
12.6.3 Receive Not Empty 207
12.7 Inactive State 208
12.8 Receive Sampling 208
12.9 Baud Rate Generation 208
12.9.1 UART Clock Sources 208
12.9.2 LPUART Clock Sources 209
12.9.3 Baud Rate Calculation 209
12.9.4 Low-Power Mode Operation of LPUARTs for 9600 Baud and Below 210
12.10 Hardware Flow Control 211
12.10.1 Automated HFC 212
12.10.2 Software Controlled HFC 212
12.11 Registers 213
12.11.1 Register Details 213
13. Serial Peripheral Interface (SPI) 220
13.1 Instances 221
13.2 Formats 222
13.2.1 Four-Wire SPI 222
13.2.2 Three-Wire SPI 223
13.3 Pin Configuration 224
13.3.1 SPI Alternate Function Mapping 224
13.3.2 Four-Wire Format Configuration 224
13.3.3 Three-Wire Format Configuration 224
13.3.4 Dual-Mode Format Configuration 225
13.3.5 Quad-Mode Format Pin Configuration 225
13.4 Clock Configuration 225
13.4.1 Serial Clock 225

13.4.2 Peripheral Clock ---------- 226
13.4.3 Controller Mode Serial Clock Generation ---------- 226
13.4.4 Clock Phase and Polarity Control ---------- 227
13.4.5 Transmit and Receive FIFOs ---------- 227
13.4.6 Interrupts and Wakeups ---------- 228
13.5 Registers ---------- 228
13.5.1 Register Details ---------- 229
14. I^{2}C Controller/Peripheral Serial Communications Peripheral ---------- 239
14.1 I^{2}C Controller/Peripheral Features ---------- 239
14.2 Instances ---------- 239
14.3 I^{2}C Overview ---------- 240
14.3.1 I^{2}C Bus Terminology ---------- 240
14.3.2 I^{2}C Transfer Protocol Operation ---------- 240
14.3.3 START and STOP Conditions ---------- 240
14.3.4 Controller Operation ---------- 240
14.3.5 Acknowledge and Not Acknowledge ---------- 241
14.3.6 Bit Transfer Process ---------- 241
14.4 Configuration and Usage ---------- 242
14.4.1 SCL and SDA Bus Drivers ---------- 242
14.4.2 SCL Clock Configurations ---------- 242
14.4.3 SCL Clock Generation for Standard, Fast and Fast-Plus Modes ---------- 242
14.4.4 SCL Clock Generation for Hs-Mode ---------- 243
14.4.5 Controller Mode Addressing ---------- 244
14.4.6 Controller Mode Operation ---------- 244
14.4.7 Peripheral Mode Operation ---------- 247
14.4.8 Interrupt Sources ---------- 253
14.4.9 Transmit FIFO and Receive FIFO ---------- 253
14.4.10 Transmit FIFO Preloading ---------- 254
14.4.11 Interactive Receive Mode (IRXM) ---------- 255
14.4.12 Clock Stretching ---------- 256
14.4.13 Bus Timeout ---------- 256
14.4.14 DMA Control ---------- 257
14.5 Registers ---------- 258
14.5.1 Register Details ---------- 258
15. Inter-Integrated Sound Interface (I^{2}S) ---------- 273
15.1 Instances ---------- 273
15.1.1 I^{2}S Bus Lines and Definitions ---------- 273
15.2 Details ---------- 274
15.3 Controller and Peripheral Mode Configuration ---------- 275
15.4 Clocking ---------- 275
15.4.1 BCLK Generation for Controller Mode ---------- 276
15.4.2 LRCLK Period Calculation ---------- 276
15.5 Data Formatting ---------- 276
15.5.1 Sample Size ---------- 276
15.5.2 Word Select Polarity ---------- 277
15.5.3 First Bit Location Control ---------- 277
15.5.4 Sample Adjustment ---------- 278

15.5.5 Stereo/Mono Configuration ---------- 278
15.6 Transmit and Receive FIFOs ---------- 279
15.6.1 FIFO Data Width ---------- 279
15.6.2 Transmit FIFO ---------- 279
15.6.3 Receive FIFO ---------- 280
15.6.4 FIFO Word Control ---------- 280
15.6.5 FIFO Data Alignment ---------- 282
15.6.6 Typical Audio Configurations ---------- 282
15.7 Interrupt Events ---------- 283
15.7.1 Receive FIFO Overrun ---------- 283
15.7.2 Receive FIFO Threshold ---------- 283
15.7.3 Transmit FIFO Half-Empty ---------- 283
15.7.4 Transmit FIFO One Entry Remaining ---------- 283
15.8 Direct Memory Access ---------- 284
15.9 Block Operation ---------- 284
15.10 Registers ---------- 284
15.10.1 Register Details ---------- 285
16. Camera Interface (CAMERAIF) ---------- 290
16.1 Instances ---------- 290
16.2 Capture Modes ---------- 291
16.2.1 Single Image Capture ---------- 291
16.2.2 Continuous Capture ---------- 291
16.3 Timing Modes ---------- 291
16.3.1 Horizontal and Vertical Synchronization Timing Mode ---------- 291
16.3.2 Data Stream Timing Mode ---------- 291
16.4 Data Width ---------- 291
16.4.1 8-Bit Width ---------- 291
16.4.2 10 and 12-bit Width ---------- 292
16.5 Data FIFO ---------- 293
16.6 Usage ---------- 293
16.6.1 DMA ---------- 293
16.6.2 Interrupts ---------- 294
16.7 Registers ---------- 294
16.7.1 Parallel Camera Register Details ---------- 294
17. MIPI CSI-2 Camera Interface (CSI2) ---------- 299
17.1 CSI-2 Registers ---------- 299
17.1.1 Register Details ---------- 301
18. 1-Wire Master (OWM) ---------- 317
18.1 1-Wire Master Features ---------- 317
18.2 1-Wire Pins and Configuration ---------- 318
18.2.1 1-Wire I/O (OWM_IO) ---------- 318
18.2.2 Pullup Enable (OWM_PE) ---------- 318
18.2.3 Clock Configuration ---------- 318
18.3 1-Wire Protocol ---------- 318
18.3.1 Networking Layers ---------- 318

18.3.2 Read ROM Command ---- 323
18.3.3 Skip ROM and Overdrive Skip ROM Commands ---- 324
18.3.4 Match ROM and Overdrive Match ROM Commands ---- 324
18.3.5 Search ROM Command ---- 324
18.3.6 Search ROM Accelerator Operation ---- 325
18.3.7 Resume Communication Command ---- 325
18.4 1-Wire Operation ---- 326
18.4.1 Resetting the OWM ---- 326
18.5 1-Wire Data Reads ---- 326
18.5.1 Reading a Single Bit Value from the 1-Wire Bus ---- 326
18.5.2 Reading an 8-Bit Value from the 1-Wire Bus ---- 327
18.6 Registers ---- 327
18.6.1 Register Details ---- 327
19. Real-Time Clock (RTC) ---- 332
19.1 Overview ---- 332
19.2 Instances ---- 333
19.3 Register Access Control ---- 333
19.3.1 RTC_SEC and RTC_SSEC Read Access Control ---- 333
19.3.2 RTC Write Access Control ---- 334
19.4 RTC Alarm Functions ---- 334
19.4.1 Time-of-Day Alarm ---- 334
19.4.2 Sub-Second Alarm ---- 334
19.4.3 RTC Interrupt and Wakeup Configuration ---- 335
19.5 Square Wave Output ---- 336
19.6 RTC Calibration ---- 337
19.7 Registers ---- 339
19.7.1 Register Details ---- 339
20. Timers (TMR/LPTMR) ---- 343
20.1 Instances ---- 344
20.2 Basic Timer Operation ---- 344
20.3 32-Bit Single / 32-Bit Cascade / Dual 16-Bit ---- 345
20.4 Timer Clock Sources ---- 345
20.5 Timer Pin Functionality ---- 346
20.6 Wake-Up Events ---- 348
20.7 Operating Modes ---- 349
20.7.1 One-Shot Mode (0) ---- 351
20.7.2 Continuous Mode (1) ---- 353
20.7.3 Counter Mode (2) ---- 355
20.7.4 PWM Mode (3) ---- 357
20.7.5 Capture Mode (4) ---- 358
20.7.6 Compare Mode (5) ---- 361
20.7.7 Gated Mode (6) ---- 363
20.7.8 Capture/Compare Mode (7) ---- 365
20.7.9 Dual Edge Capture Mode (8) ---- 367
20.7.10 Inactive Gated Mode (14) ---- 367

20.8 Registers ---- 367

20.8.1 Register Details ---- 368

21. Wake-Up Timer (WUT) ---- 376

21.1 Instances ---- 376

21.2 Basic Operation ---- 376

21.3 One-Shot Mode (0) ---- 377

21.3.1 One-Shot Mode Timer Period ---- 377

21.3.2 One-Shot Mode Configuration ---- 378

21.4 Continuous Mode (1) ---- 378

21.4.1 Continuous Mode Timer Period ---- 379

21.4.2 Continuous Mode Configuration ---- 379

21.4.3 Compare Mode (5) ---- 379

21.5 Registers ---- 380

21.5.1 Register Details ---- 381

22. Watchdog Timer (WDT) ---- 383

22.1 Instances ---- 384

22.2 Usage ---- 384

22.2.1 Using the WDT as a Long-Interval Timer ---- 385

22.2.2 Using the WDT as a Long-Interval Wakeup Timer ---- 385

22.3 WDT Feed Sequence ---- 385

22.4 WDT Events ---- 386

22.4.1 WDT Early Reset ---- 386

22.4.2 WDT Early Interrupt ---- 387

22.4.3 WDT Late Reset ---- 387

22.4.4 WDT Late Interrupt ---- 388

22.5 Initializing the WDT ---- 389

22.6 Resets ---- 389

22.7 Registers ---- 390

22.7.1 Register Details ---- 390

23. Pulse Train Engine (PT) ---- 394

23.1 Instances ---- 394

23.2 Features ---- 394

23.3 Engine ---- 394

23.3.1 Pulse Train Output Modes ---- 394

23.4 Enabling and Disabling a Pulse Train Output ---- 396

23.5 Atomic Pulse Train Output Enable and Disable ---- 396

23.5.1 Pulse Train Atomic Enable ---- 396

23.5.2 Pulse Train Atomic Disable ---- 396

23.6 Halt and Disable ---- 397

23.7 Interrupts ---- 397

23.8 Registers ---- 397

23.8.1 Register Details ---- 398

24. Cyclic Redundancy Check (CRC) ---- 405
24.1 Instances ---- 405
24.2 Usage ---- 405
24.3 Polynomial Generation ---- 406
24.4 Software CRC Calculations ---- 407
24.5 DMA CRC Calculations ---- 408
24.6 Registers ---- 408
24.6.1 Register Details ---- 409
25. Advanced Encryption Standard (AES) ---- 411
25.1 Instances ---- 411
25.2 Encryption of 128-Bit Blocks of Data Using FIFO ---- 411
25.3 Encryption of 128-Bit Blocks Using DMA ---- 412
25.4 Encryption of Blocks Less Than 128-Bits ---- 413
25.5 Decryption ---- 413
25.6 Interrupt Events ---- 413
25.6.1 Data Output FIFO Overrun ---- 414
25.6.2 Key Zero ---- 414
25.6.3 Key Change ---- 414
25.6.4 Calculation Done ---- 414
25.7 Registers ---- 414
25.7.1 Register Details ---- 414
26. TRNG Engine ---- 417
26.1 Registers ---- 417
26.1.1 Register Details ---- 417
27. Secure Digital Host Controller (SDHC) ---- 419
27.1 Instances ---- 420
27.2 SDHC Peripheral Clock Selection ---- 421
27.3 Usage ---- 422
27.4 SD Command Generation ---- 423
27.5 Registers ---- 423
27.6 Register Details ---- 425
27.6.2 Normal Interrupt Status Register ---- 440
27.6.3 Error Interrupt Status Register ---- 443
27.6.4 Auto CMD Error Status Register ---- 448
27.6.5 Preset Value Registers ---- 456
28. Convolutional Neural Network ---- 458
29. Bootloader ---- 462
29.1 Instances ---- 462
29.2 Bootloader Operating States ---- 462
29.2.1 UNLOCKED ---- 463
29.2.2 LOCKED ---- 463

29.2.3 PERMLOCKED-- 463
29.2.4 CHALLENGE (Secure Bootloader Only)-- 463
29.2.5 APPVERIFY (Secure Bootloader only) --- 464
29.3 Creating the Motorola SREC File--- 464
29.4 Bootloader Activation -- 465
29.5 Bootloader-- 465
29.6 Secure Bootloader --- 466
29.6.1 Secure Boot --- 466
29.6.2 Secure Challenge/Response Authentication -- 467
29.7 Command Protocol-- 467
29.8 General Commands -- 468
29.8.1 General Command Details -- 468
29.9 Secure Commands--- 478
29.9.1 Secure Command Details--- 478
29.10 Challenge/Response Commands -- 485
29.10.1 Challenge/Response Command Details -- 485
30. Revision History-- 487

Table of Figures

Figure 2-1: MAX78002 Block Diagram 31
Figure 3-1: CM4 Code Memory Mapping 33
Figure 3-2: RISC-V IBUS Code Memory Mapping 34
Figure 3-3: CM4 Peripheral and Data Memory Mapping 35
Figure 3-4: RV32 Peripheral and Data Memory Mapping 36
Figure 3-5: Unique Serial Number Format 37
Figure 4-1: MAX78002 Clock Block Diagram 47
Figure 4-2: SLEEP Mode Clock Control 49
Figure 4-3: LPM Clock and State Retention Diagram 51
Figure 4-4: UPM Clock and State Retention Block Diagram 53
Figure 4-5: STANDBY Mode Clock and State Retention Block Diagram 55
Figure 4-6: BACKUP Mode Clock and State Retention Block Diagram 57
Figure 4-7: PDM Clock and State Retention Block Diagram 59
Figure 10-1: DMA Block-Chaining Flowchart 159
Figure 11-1: ADC Sample Clock 170
Figure 11-2: ADC Operating Modes State Diagram 172
Figure 11-3: Interrupt Event Signal Generation 178
Figure 11-4: ADC Result Formats (Single-Ended) 180
Figure 11-5: ADC Result Formats (Differential, Temperature Sensor Only) 181
Figure 11-6: Analog Wakeup Comparators 188
Figure 12-1: UART Block Diagram 203
Figure 12-2: UART Frame Structure 204
Figure 12-3: UART Interrupt Functional Diagram 205
Figure 12-4: Oversampling Example 208
Figure 12-5: UART Baud Rate Generation 209
Figure 12-6: LPUART Timing Generation 209
Figure 12-7: HFC Physical Connection 211
Figure 12-8: HFC Signaling for Transmitting to an External Receiver 212
Figure 13-1: SPI Block Diagram 221
Figure 13-2: 4-Wire SPI Connection Diagram 223
Figure 13-3: Generic 3-Wire SPI Controller to Peripheral Connection 224
Figure 13-4: Dual Mode SPI Connection Diagram 225
Figure 13-5: SCK Clock Rate Control 226
Figure 13-6: SPI Clock Polarity 227
Figure 14-1: I^{2}C Write Data Transfer 241
Figure 14-2: I^{2}C SCL Timing for Standard, Fast and Fast-Plus Modes 243
Figure 15-1: I^{2}S Controller Mode 274
Figure 15-2: I^{2}S Peripheral Mode 274
Figure 15-3: Audio Interface I^{2}S Signal Diagram 275
Figure 15-4: Audio Mode with Inverted Word Select Polarity 277
Figure 15-5: Audio Controller Mode Left-Justified First Bit Location 277
Figure 15-6: MSB Adjustment when Sample Size is Less Than Bits Per Word 278
Figure 15-7: LSB Adjustment when Sample Size is Less Than Bits Per Word 278
Figure 15-8: I^{2}S Mono Left Mode 279
Figure 15-9: I^{2}S Mono Right Mode 279
Figure 16-1: Horizontal and Vertical Synchronization Timing Mode with 8-Bit Data Width 292
Figure 16-2: Data Stream Timing Mode with 8-Bit Data Width 292
Figure 16-3: 10 or 12-bit PCIF_VSYNC/PCIF_HSYNC 293
Figure 18-1: 1-Wire Signal Interface 319

Figure 18-2: 1-Wire Reset Pulse....320
Figure 18-3: 1-Wire Write Time Slot....321
Figure 18-4: 1-Wire Read Time Slot....321
Figure 18-5: 1-Wire ROM ID Fields....323
Figure 19-1: MAX78002 RTC Block Diagram....332
Figure 19-2: RTC Interrupt/Wakeup Diagram Wake-up Function....335
Figure 19-3: Internal Implementation of 4kHz Digital Trim....337
Figure 20-1: MAX78002 TimerA Output Functionality, Modes 0/1/3/5....347
Figure 20-2: MAX78002 TimerA Input Functionality, Modes 2/4/6/7/8/14....348
Figure 20-3: Timer I/O Signal Naming Conventions....349
Figure 20-4: One-Shot Mode Diagram....352
Figure 20-5: Continuous Mode Diagram....354
Figure 20-6: Counter Mode Diagram....356
Figure 20-7: Capture Mode Diagram....360
Figure 20-8: Compare Mode Diagram....362
Figure 20-9: Gated Mode Diagram....364
Figure 20-10: Capture/Compare Mode Diagram....366
Figure 21-1: One-Shot Mode Diagram....377
Figure 21-2: Continuous Mode Diagram....378
Figure 21-3: Compare Mode Diagram....380
Figure 22-1: Windowed Watchdog Timer Block Diagram....384
Figure 22-2: WDT Early Interrupt and Reset Event Sequencing Details....387
Figure 22-3: WDT Late Interrupt and Reset Event Sequencing Details....388
Figure 27-1: SDHC Block Diagram....421
Figure 27-2: SD Bus Protocol - No Response and No Data Operations....422
Figure 27-3: SD Bus Protocol - Multi-Block Read Operation....422
Figure 27-4: SD Bus Protocol - Multi Block Write Operation....423
Figure 29-1: MAX78002 Combined Bootloader Flow....466

List of Tables

Table 1-1: Field Access Definitions 28
Table 1-2: Example Registers 29
Table 1-3: Example Name 0 Register 29
Table 3-1: System SRAM Configuration 38
Table 3-2: AHB Slave Base Address Map 40
Table 3-3: APB Peripheral Base Address Map 41
Table 4-1: Available System Oscillators 45
Table 4-2: Reset Sources and Effect on Oscillator and System Clock 46
Table 4-3 System RAM Retention in BACKUP Mode 56
Table 4-4: Wake-Up Sources for Each Operating Mode in the MAX78002 60
Table 4-5: Reset and Low-Power Mode Effects 61
Table 4-6: Instruction Cache Controller Register Summary 63
Table 4-7: ICC0 Cache Information Register 63
Table 4-8: ICC0 Memory Size Register 63
Table 4-9: ICC0 Cache Control Register 63
Table 4-10: ICC0 Invalidate Register 64
Table 4-11: Miscellaneous Control Register Summary 65
Table 4-12: Error Correction Coding Enable Register 65
Table 4-13: IPO Manual Register 65
Table 4-14: Output Enable Register 66
Table 4-15: Comparator 0 Control Register 66
Table 4-16: Miscellaneous Control Register 67
Table 4-17: GPIO3 Pin Control Register 67
Table 4-18: Code Word 0 Register 68
Table 4-19: Code Word 1 Register 68
Table 4-20: ADC Configuration Register 0 68
Table 4-21: ADC Configuration Register 1 69
Table 4-22: ADC Configuration Register 2 69
Table 4-23: LDO Control Register 69
Table 4-24: SIMO Power Supply Device Pin Connectivity 70
Table 4-25: SIMO Controller Register Summary 70
Table 4-26: SIMO Buck Voltage Regulator A Control Register 71
Table 4-27: SIMO Buck Voltage Regulator B Control Register 71
Table 4-28: SIMO Buck Voltage Regulator C Control Register 72
Table 4-29: SIMO High Side FET Peak Current V_{REGO_A} V_{REGO_B} Register 72
Table 4-30: SIMO High Side FET Peak Current V_{REGO_C} Register 73
Table 4-31: SIMO Maximum High Side FET Time On Register 73
Table 4-32: SIMO Buck Cycle Count V_{REGO_A} Register 73
Table 4-33: SIMO Buck Cycle Count V_{REGO_B} Register 73
Table 4-34: SIMO Buck Cycle Count V_{REGO_C} Register 73
Table 4-35: SIMO Buck Cycle Count Alert V_{REGO_A} Register 73
Table 4-36: SIMO Buck Cycle Count Alert V_{REGO_B} Register 74
Table 4-37: SIMO Buck Cycle Count Alert V_{REGO_C} Register 74
Table 4-38: SIMO Buck Regulator Output Ready Register 74
Table 4-39: SIMO Zero Cross Calibration V_{REGO_A} Register 74
Table 4-40: SIMO Zero Cross Calibration V_{REGO_B} Register 75
Table 4-41: SIMO Zero Cross Calibration V_{REGO_C} Register 75
Table 4-42: Low-Power Control Register Summary 75
Table 4-43: Reset Control Register 75
Table 4-44: Clock Disable Register 76

Table 4-45: Power Sequencer Register Summary....77
Table 4-46: Low Power Control Register....77
Table 4-47: GPIO0 Low Power Wakeup Status Flags....79
Table 4-48: GPIO0 Low Power Wakeup Enable Registers....79
Table 4-49: GPIO1 Low Power Wakeup Status Flags....79
Table 4-50: GPIO1 Low Power Wakeup Enable Registers....80
Table 4-51: GPIO2 Low Power Wakeup Status Flags....80
Table 4-52: GPIO2 Low Power Wakeup Enable Registers....80
Table 4-53: GPIO3 Low Power Wakeup Status Flags....81
Table 4-54: GPIO3 Low Power Wakeup Enable Registers....81
Table 4-55: Low Power Peripheral Wakeup Status Flags....81
Table 4-56: Low Power Peripheral Wakeup Enable Registers....82
Table 4-57: Low Power General Purpose 0 Register....84
Table 4-58: Low Power General Purpose 1 Register....84
Table 4-59: Trim System Initialization Register Summary....84
Table 4-60: RTC Trim System Initialization Register....85
Table 4-61: SIMO Trim System Initialization Register....85
Table 4-62: IPO Low Trim System Initialization Register....85
Table 4-63: Control Trim System Initialization Register....86
Table 4-64: INRO Trim System Initialization Register....86
Table 4-65: Global Control Register Summary....87
Table 4-66: System Control Register....87
Table 4-67: Reset Register 0....89
Table 4-68: Clock Control Register....91
Table 4-69: Power Management Register....92
Table 4-70: PLL Control Register....94
Table 4-71: Peripheral Clock Divisor Register....94
Table 4-72: Peripheral Clock Disable Register 0....95
Table 4-73: Memory Clock Control Register....97
Table 4-74: Memory Zeroize Control Register....97
Table 4-75: System Status Flag Register....99
Table 4-76: Reset Register 1....99
Table 4-77: Peripheral Clock Disable Register 1....101
Table 4-78: Event Enable Register....103
Table 4-79: Revision Register....103
Table 4-80: System Status Interrupt Enable Register....103
Table 4-81: Error Correction Coding Error Register....103
Table 4-82: Error Correction Coding Correctable Error Detected Register....104
Table 4-83: Error Correction Coding Interrupt Enable Register....104
Table 4-84: Error Correction Coding Error Address Register....104
Table 4-85: General Purpose 0 Register....105
Table 4-86: System Initialization Register Summary....105
Table 4-87: System Initialization Status Register....105
Table 4-88: System Initialization Address Error Register....106
Table 4-89: System Initialization Function Status Register....106
Table 4-90: System Initialization Security Function Status Register....106
Table 4-91: Function Control Register Summary....107
Table 4-92: Function Control 0 Register....107
Table 4-93: IPO Automatic Calibration 0 Register....108
Table 4-94: IPO Automatic Calibration 1 Register....109
Table 4-95: IPO Automatic Calibration 2 Register....109
Table 4-96: RV32 Boot Address Register....109
Table 4-97: RV32 Control Register....109

Table 4-98: Temperature Sensor Gain Register 110
Table 4-99: Temperature Sensor Offset Register 110
Table 4-100: ADC 1.25V Reference Trim Register 110
Table 4-101: ADC 2.048V Reference Trim Register 111
Table 4-102: ADC External Reference Trim Register 111
Table 4-103: General Control Function Register Summary 112
Table 4-104: General Control Function Register 0 112
Table 4-105: General Control Function Register 1 113
Table 4-106: General Control Function Register 2 113
Table 4-107: General Control Function Register 3 115
Table 5-1: MAX78002 CM4 Interrupt Vector Table 116
Table 5-2: MAX78002 RV32 Interrupt Vector Table 118
Table 6-1: MAX78002 GPIO Pin Count 121
Table 6-2: MAX78002 GPIO Pin Function Configuration 122
Table 6-3: MAX78002 Input Mode Configuration 122
Table 6-4: MAX78002 Output Mode Configuration 123
Table 6-5: MAX78002 GPIO Alternate Function Configuration Reference 123
Table 6-6: MAX78002 GPIO Output/Input Configuration Reference 124
Table 6-7: MAX78002 GPIO Interrupt Configuration Reference 124
Table 6-8: MAX78002 GPIO Pullup/Pulldown/Drive Strength/Voltage Configuration Reference 124
Table 6-9: MAX78002 GPIO Port Interrupt Vector Mapping 126
Table 6-10: MAX78002 GPIO Wakeup Interrupt Vector 126
Table 6-11: GPIO Register Summary 127
Table 6-12: GPIO Port n Configuration Enable Bit 0 Register 128
Table 6-13: GPIO Port n Configuration Enable Atomic Set Bit 0 Register 128
Table 6-14: GPIO Port n Configuration Enable Atomic Clear Bit 0 Register 128
Table 6-15: GPIO Port n Output Enable Register 129
Table 6-16: GPIO Port n Output Enable Atomic Set Register 129
Table 6-17: GPIO Port n Output Enable Atomic Clear Register 129
Table 6-18: GPIO Port n Output Register 129
Table 6-19: GPIO Port n Output Atomic Set Register 129
Table 6-20: GPIO Port n Output Atomic Clear Register 130
Table 6-21: GPIO Port n Input Register 130
Table 6-22: GPIO Port n Interrupt Mode Register 130
Table 6-23: GPIO Port n Interrupt Polarity Register 130
Table 6-24: GPIO Port n Input Enable Register 131
Table 6-25: GPIO Port n Interrupt Enable Register 131
Table 6-26: GPIO Port n Interrupt Enable Atomic Set Register 131
Table 6-27: GPIO Port n Interrupt Enable Atomic Clear Register 131
Table 6-28: GPIO Port n Interrupt Status Register 131
Table 6-29: GPIO Port n Interrupt Clear Register 132
Table 6-30: GPIO Port n Wakeup Enable Register 132
Table 6-31: GPIO Port n Wakeup Enable Atomic Set Register 132
Table 6-32: GPIO Port n Wakeup Enable Atomic Clear Register 132
Table 6-33: GPIO Port n Interrupt Dual Edge Mode Register 132
Table 6-34: GPIO Port n Pad Configuration 1 Register 132
Table 6-35: GPIO Port n Pad Configuration 2 Register 133
Table 6-36: GPIO Port n Configuration Enable Bit 1 Register 133
Table 6-37: GPIO Port n Configuration Enable Atomic Set Bit 1 Register 133
Table 6-38: GPIO Port n Configuration Enable Atomic Clear Bit 1 Register 133
Table 6-39: GPIO Port n Configuration Enable Bit 2 Register 134
Table 6-40: GPIO Port n Configuration Enable Atomic Set Bit 2 Register 134
Table 6-41: GPIO Port n Configuration Enable Atomic Clear Bit 2 Register 134

Table 6-42: GPIO Port n Hysteresis Enable Register 134
Table 6-43: GPIO Port n Output Drive Strength Bit 0 Register 134
Table 6-44: GPIO Port n Output Drive Strength Bit 0 Register 134
Table 6-45: GPIO Port n Output Drive Strength Bit 1 Register 135
Table 6-46: GPIO Port n Pulldown/Pullup Strength Select Register 135
Table 6-47: GPIO Port n Voltage Select Register 135
Table 7-1: MAX78002 Internal Flash Memory Organization 136
Table 7-2: Valid Addresses Flash Writes 137
Table 7-3: Flash Controller Register Summary 139
Table 7-4: Flash Controller Address Pointer Register 139
Table 7-5: Flash Controller Clock Divisor Register 139
Table 7-6: Flash Controller Control Register 140
Table 7-7: Flash Controller Interrupt Register 141
Table 7-8: Flash Controller Data 0 Register 142
Table 7-9: Flash Controller Data Register 1 142
Table 7-10: Flash Controller Data Register 2 142
Table 7-11: Flash Controller Data Register 3 142
Table 7-12: Flash Controller Access Control Register 142
Table 7-13: Flash Write/Lock 0 Register 142
Table 7-14: Flash Write/Lock 1 Register 143
Table 7-15: Flash Write/Lock 2 Register 143
Table 7-16: Flash Write/Lock 3 Register 143
Table 7-17: Flash Write/Lock 4 Register 143
Table 7-18: Flash Read Lock 0 Register 144
Table 7-19: Flash Read Lock 1 Register 144
Table 7-20: Flash Read Lock 2 Register 144
Table 7-21: Flash Read Lock 3 Register 144
Table 7-22: Flash Read Lock 4 Register 145
Table 8-1: MAX78002 DAP Instances 146
Table 9-1: MAX78002 Semaphore Instances 147
Table 9-2: Semaphore Register Summary 148
Table 9-3: Semaphore 0 Register 148
Table 9-4: Semaphore 1 Register 148
Table 9-5: Semaphore 2 Register 149
Table 9-6: Semaphore 3 Register 149
Table 9-7: Semaphore 4 Register 149
Table 9-8: Semaphore 5 Register 149
Table 9-9: Semaphore 6 Register 149
Table 9-10: Semaphore 7 Register 150
Table 9-11: Semaphore Interrupt 0 Register 150
Table 9-12: Semaphore Mailbox 0 Register 150
Table 9-13: Semaphore Interrupt 1 Register 151
Table 9-14: Semaphore Mailbox 1 Register 151
Table 9-15: Semaphore Status Register 151
Table 10-1: MAX78002 DMA and Channel Instances 153
Table 10-2: MAX78002 DMA Source and Destination by Peripheral 155
Table 10-3: Data Movement from Source to DMA FIFO 156
Table 10-4: Data Movement from the DMA FIFO to Destination 156
Table 10-5: DMA Channel Timeout Configuration 160
Table 10-6: DMA Register Summary 161
Table 10-7: DMA Interrupt Enable Register 161
Table 10-8: DMA Interrupt Flag Register 161
Table 10-9: Standard DMA Channel 0 to Channel 7 Register Summary 162

Table 10-10: DMA Channel Registers Summary 162
Table 10-11: DMA Channel n Control Register 162
Table 10-12: DMA Status Register 164
Table 10-13: DMA Channel n Source Register 165
Table 10-14: DMA Channel n Destination Register 166
Table 10-15: DMA Channel n Count Register 166
Table 10-16: DMA Channel n Source Reload Register 166
Table 10-17: DMA Channel n Destination Reload Register 166
Table 10-18: DMA Channel n Count Reload Register 167
Table 11-1: MAX78002 Channel Assignments (All tables need updates) 168
Table 11-2: ADC Voltage Divider Configuration for Channels 0 through 12 169
Table 11-3: MAX78002 ADC Clock Sources 169
Table 11-4: ADC Operating States 171
Table 11-5: Bias and Wake-up Clock Cycle Selection 174
Table 11-6: MAX78002 Interrupt Events 178
Table 11-7: ADC_DATA Register Result Formatting 180
Table 11-8: MAX78002 Hardware Conversion Triggers 181
Table 11-9: Conversion Sequence Configurations 182
Table 11-10: MAX78002 Analog Comparator 0 Input Selection 188
Table 11-11: ADC Register Summary 189
Table 11-12: ADC Control 0 Register 190
Table 11-13: ADC Control 1 Register 190
Table 11-14: ADC Clock Control Register 191
Table 11-15: ADC Sample Clock Control Register 191
Table 11-16: ADC Channel Select 0 Register 192
Table 11-17: ADC Channel Select 1 Register 192
Table 11-18: ADC Channel Select 2 Register 193
Table 11-19: ADC Channel Select 3 Register 193
Table 11-20: ADC Channel Select 4 Register 193
Table 11-21: ADC Channel Select 5 Register 194
Table 11-22: ADC Channel Select 6 Register 194
Table 11-23: ADC Channel Select 7 Register 195
Table 11-24: ADC Restart Count Register 195
Table 11-25: ADC Data Format Register 195
Table 11-26: ADC FIFO and DMA Control Register 196
Table 11-27: ADC Data Register 196
Table 11-28: ADC Status Register 196
Table 11-29: ADC Channel Status Register 197
Table 11-30: ADC Interrupt Enable Register 197
Table 11-31: ADC Interrupt Flags Register 198
Table 11-32: ADC SFR Address Offset Register 199
Table 11-33: ADC SFR Address Register 199
Table 11-34: ADC SFR Write Data Register 199
Table 11-35: ADC SFR Read Data Register 199
Table 11-36: ADC SFR Status Register 200
Table 11-37: Low-Power Comparator Registers Summary 200
Table 11-38: Low-Power Comparator n Registers 201
Table 12-1: MAX78002 UART/LPUART Instances 203
Table 12-2: MAX78002 Interrupt Events 205
Table 12-3: Frame Error Detection for Standard UARTs and LPUART 206
Table 12-4: Frame Error Detection for LPUARTs with UARTn_CTRL.fdm = 1 and UARTn_CTRL.dpfe_en = 1 206
Table 12-5: MAX78002 Wakeup Events 207
Table 12-6: LPUART Low Baud Rate Generation Examples (UARTn_CTRL.fdm = 1) 210

Table 12-7: UART/LPUART Register Summary 213

Table 12-8: UART Control Register 213

Table 12-9: UART Status Register 215

Table 12-10: UART Interrupt Enable Register 216

Table 12-11: UART Interrupt Flag Register 216

Table 12-12: UART Clock Divisor Register 217

Table 12-13: UART Oversampling Control Register 217

Table 12-14: UART Transmit FIFO Register 217

Table 12-15: UART Pin Control Register 217

Table 12-16: UART Data Register 218

Table 12-17: UART DMA Register 218

Table 12-18: UART Wakeup Enable 218

Table 12-19. UART Wakeup Flag Register 219

Table 13-1: MAX78002 SPI Instances 221

Table 13-2: MAX78002 SPI Peripheral Pins 222

Table 13-3: Four-Wire Format Signals 222

Table 13-4: Three-Wire Format Signals 223

Table 13-5: SPI Modes Clock Phase and Polarity Operation 227

Table 13-6: SPI Register Summary 228

Table 13-7: SPI FIFO32 Register 229

Table 13-8: SPI 16-bit FIFO Register 229

Table 13-9: SPI 8-bit FIFO Register 229

Table 13-10: SPI Control 0 Register 230

Table 13-11: SPI Control 1 Register 231

Table 13-12: SPI Control 2 Register 231

Table 13-13: SPI Peripheral Select Timing Register 233

Table 13-14: SPI Controller Clock Configuration Registers 233

Table 13-15: SPI DMA Control Registers 234

Table 13-16: SPI Interrupt Status Flags Registers 235

Table 13-17: SPI Interrupt Enable Registers 236

Table 13-18: SPI Wakeup Status Flags Registers 237

Table 13-19: SPI Wakeup Enable Registers 237

Table 13-20: SPI Peripheral Select Timing Registers 238

Table 14-1: MAX78002 I^{2}C Peripheral Pins 239

Table 14-2: I^{2}C Bus Terminology 240

Table 14-3: Calculated I^{2}C Bus Clock Frequencies 244

Table 14-4: I^{2}C Peripheral Address Format 244

Table 14-5: Register Summary 258

Table 14-6: I^{2}C Control Register 258

Table 14-7: I^{2}C Status Register 260

Table 14-8: I^{2}C Interrupt Flag 0 Register 260

Table 14-9: I^{2}C Interrupt Enable 0 Register 263

Table 14-10: I^{2}C Interrupt Flag 1 Register 264

Table 14-11: I^{2}C Interrupt Enable 1 Register 265

Table 14-12: I^{2}C FIFO Length Register 265

Table 14-13: I^{2}C Receive Control 0 Register 265

Table 14-14: I^{2}C Receive Control 1 Register 266

Table 14-15: I^{2}C Transmit Control 0 Register 266

Table 14-16: I^{2}C Transmit Control 1 Register 268

Table 14-17: I^{2}C Data Register 268

Table 14-18: I^{2}C Controller Control Register 268

Table 14-19: I^{2}C SCL Low Control Register 269

Table 14-20: I^{2}C SCL High Control Register 269

Table 14-21: I^{2}C Hs-Mode Clock Control Register 269
Table 14-22: I^{2}C Timeout Register 270
Table 14-23: I^{2}C DMA Register 270
Table 14-24: I^{2}C Peripheral Address 0 Register 270
Table 14-25: I^{2}C Peripheral Address 1 Register 271
Table 14-26: I^{2}C Peripheral Address 2 Register 271
Table 14-27: I^{2}C Peripheral Address 3 Register 271
Table 15-1: MAX78002 I^{2}S Instances 273
Table 15-2: MAX78002 I^{2}S Pin Mapping 274
Table 15-3: I^{2}S Mode Configuration 275
Table 15-4: Data Ordering for Byte Data Size (Stereo Mode) 281
Table 15-5: Data Ordering for Half-Word Data Size (Stereo Mode) 281
Table 15-6: Data Ordering for Word Data Size (Stereo Mode) 281
Table 15-7: Configuration for Typical Audio Width and Samples per WS Clock Cycle 282
Table 15-8. I^{2}S Interrupt Events 283
Table 15-9: I^{2}S Register Summary 284
Table 15-10: I^{2}S Control 0 Register 285
Table 15-11: I^{2}S Controller Mode Configuration Register 286
Table 15-12: I^{2}S DMA Control Register 287
Table 15-13: I^{2}S FIFO Register 288
Table 15-14: I^{2}S Interrupt Flag Register 288
Table 15-15: I^{2}S Interrupt Enable Register 289
Table 16-1: MAX78002 CAMERAIF Instances 290
Table 16-2: MAX78002 CAMERAIF Signals 290
Table 16-3: Parallel Camera Interface Register Summary 294
Table 16-4: CAMERAIF Version Register 294
Table 16-5: CAMERAIF FIFO Size Register 295
Table 16-6: CAMERAIF Configuration Register 295
Table 16-7: CAMERAIF Interrupt Enable Register 296
Table 16-8: CAMERAIF Status Flags Register 297
Table 16-9: CAMERAIF Timing Codes Register 297
Table 16-10: CAMERAIF FIFO Data Register 298
Table 17-1: MIPI CSI2 Register Summary 299
Table 17-2: Number of Lanes Configuration Register 301
Table 17-3: Configuration Clock Lane Enable Register 301
Table 17-4: Configuration Data Lane Enable Register 301
Table 17-5: Configuration Flush Count Register 301
Table 17-6: Configuration Bit Error Register 301
Table 17-7: Interrupt Status Register 302
Table 17-8: Interrupt Enable Register 302
Table 17-9: Interrupt Clear Register 302
Table 17-10: ULPS Clock Status Register 303
Table 17-11: ULPS Status Register 303
Table 17-12: ULPS Clock Mark Status Register 303
Table 17-13: ULPS Mark Status Register 303
Table 17-14: PHY Protocol Interface(PPI) Start of Transmission (SoT) Error Register 304
Table 17-15: PPI SoT Synchronization Error Register 304
Table 17-16: PPI Escape Entry Error Register 304
Table 17-17: PPI Escape Synchronization Error Register 304
Table 17-18: PPI Control Error Register 305
Table 17-19: Configuration C-PHY Enable Register 305
Table 17-20: Configuration PPI 16 Bit Enable Register 305
Table 17-21: Packet Interface Configuration Register 305

Table 17-22: Virtual Channel Extension Configuration Register 305
Table 17-23: Byte Data Configuration Register 306
Table 17-24: Disable Payload 0 Configuration Register 306
Table 17-25: Disable Payload 1 Configuration Register 308
Table 17-26: CSI-2 RX Controller IGNORE_VC Configuration Register 309
Table 17-27: CSI-2 RX Controller VC Configuration Register 309
Table 17-28: CSI-2 RX Controller P_FIFO Send Level Register 309
Table 17-29: CSI-2 RX Controller VSYNC Configuration Register 309
Table 17-30: CSI-2 RX Controller HSYNC_FP Configuration Register 309
Table 17-31: CSI-2 RX Controller HSYNC Configuration Register 310
Table 17-32: CSI-2 RX Controller HSYNC_BP Configuration Register 310
Table 17-33: High Speed Mode Data Bus Configuration Register 310
Table 17-34: Data Lane 0 Configuration Register 310
Table 17-35: Data Lane 1 Configuration Register 310
Table 17-36: Data Lane 2 Configuration Register 310
Table 17-37: Data Lane 3 Configuration Register 310
Table 17-38: Clock Lane Control Configuration Register 310
Table 17-39: Data Lane Swap Configuration Register 311
Table 17-40: Reference Clock Counter Configuration Register 311
Table 17-41: Pre-Zero Timing Clock Lane 0 Configuration Register 311
Table 17-42: Pre-Zero Timing Data Lanes Configuration Register 311
Table 17-43: Reset De-Skew Configuration Register 311
Table 17-44: PMA Circuit Ready Register 311
Table 17-45: DSI Receive Enable Register 311
Table 17-46: Sub-LVDS Mode Enable Register 311
Table 17-47: High-Speed Receive Termination Enable Register 312
Table 17-48: Debug MUX Selection Register 312
Table 17-49: Debug MUX Selection Register 312
Table 17-50: Debug MUX Output Register 312
Table 17-51: Debug MUX Output Register 312
Table 17-52: Power Ready Signal to DPHY Register 312
Table 17-53: Reset Control to DPHY Register 312
Table 17-54: Invert PPI Input Clock from DPHY Register 312
Table 17-55: Video FIFO Configuration 0 Register 313
Table 17-56: Video FIFO Configuration 1 Register 313
Table 17-57: Video FIFO Control Register 313
Table 17-58: Video FIFO Status Register 313
Table 17-59: Video FIFO CSI Line Number Per Frame Register 313
Table 17-60: Video FIFO CSI Pixel Number Per Line Register 313
Table 17-61: Video FIFO CSI Line Count Register 313
Table 17-62: Video FIFO CSI Pixel Count Register 313
Table 17-63: Video FIFO Frame Status Register 314
Table 17-64: Video FIFO RAW-to-RGB Control Register 314
Table 17-65: Video FIFO RAW-to-RGB Line Buffer 0 Address Register 314
Table 17-66: Video FIFO RAW-to-RGB Line Buffer 1 Address Register 314
Table 17-67: Video FIFO AHB Master Control Register 314
Table 17-68: Video FIFO AHB Master Status Register 314
Table 17-69: Video FIFO AHB Master Start Address Register 314
Table 17-70: Video FIFO AHB Master Address Range Register 314
Table 17-71: Video FIFO AHB Master Maximal Transfer Number Register 315
Table 17-72: Video FIFO AHB Master Transfer Count Register 315
Table 17-73: CSI2 Video FIFO Interrupt Enable Register 315
Table 17-74: CSI2 Video FIFO Interrupt Flag Register 315

Table 17-75: CSI2 DPHY Interrupt Enable Register 315
Table 17-76: CSI2 DPHY FIFO Interrupt Flag Register 315
Table 17-77: CSI2 RX Controller Interrupt Enable Register....315
Table 17-78: CSI2 RX Controller Interrupt Flag Register 315
Table 17-79: DPHY PPI Stop State Register....316
Table 17-80: DPHY PPI Turn-Around Configuration Register....316
Table 18-1: MAX78002 1-Wire Master Peripheral Pins 318
Table 18-2: 1-Wire ROM Commands 322
Table 18-3: 1-Wire Slave Device ROM ID Field 323
Table 18-4: OWM Register Summary 327
Table 18-5: OWM Configuration Register....327
Table 18-6: OWM Clock Divisor Register 328
Table 18-7: OWM Control Status Register....329
Table 18-8: OWM Data Buffer Register 330
Table 18-9: OWM Interrupt Flag Register....330
Table 18-10: OWM Interrupt Enable Register 330
Table 19-1: RTC Seconds, Sub-Seconds, Time-of-Day Alarm, and Sub-Second Alarm Register Details....333
Table 19-2: RTC Register Access 333
Table 19-3: MAX78002 RTC Square Wave Output Configuration....336
Table 19-4: RTC Register Summary....339
Table 19-5: RTC Seconds Counter Register....339
Table 19-6: RTC Sub-Second Counter Register 339
Table 19-7: RTC Time-of-Day Alarm Register....339
Table 19-8: RTC Sub-Second Alarm Register....339
Table 19-9: RTC Control Register 340
Table 19-10: RTC 32KHz Oscillator Digital Trim Register 342
Table 19-11: RTC 32KHz Oscillator Control Register....342
Table 20-1: MAX78002 TMR/LPTMR Instances 344
Table 20-2: MAX78002 TMR/LPTMR Instances Capture Events....344
Table 20-3: TimerA/TimerB 32-Bit Field Allocations....345
Table 20-4: MAX78002 Wake-Up Events....348
Table 20-5: MAX78002 Operating Mode Signals for Timer 0 and Timer 1 349
Table 20-6: MAX78002 Operating Mode Signals for Timer 2 and Timer 3 350
Table 20-7: MAX78002 Operating Mode Signals for Low-Power Timer 0 and Low-Power Timer 1 350
Table 20-8: Timer Register Summary....367
Table 20-9: Timer Count Register 368
Table 20-10: Timer Compare Register 368
Table 20-11: Timer PWM Register....368
Table 20-12: Timer Interrupt Register 368
Table 20-13: Timer Control 0 Register....369
Table 20-14: Timer Non-Overlapping Compare Register....372
Table 20-15: Timer Control 1 Register....372
Table 20-16: Timer Wake-Up Status Register....374
Table 21-1: MAX78002 WUT Clock Period....376
Table 21-2: Wake-Up Timer Register Summary....381
Table 21-3: Wake-Up Timer Count Register 381
Table 21-4: Wake-Up Timer Compare Register 381
Table 21-5: Wake-Up Timer PWM Register....381
Table 21-6: Wake-Up Timer Interrupt Register 381
Table 21-7: Wake-Up Timer Control Register....381
Table 21-8: Wake-Up Timer Non-Overlapping Compare Register....382
Table 22-1: MAX78002 WDT Instances Summary 384
Table 22-2: WDT Event Summary 386

Table 22-3: WDT Register Summary 390
Table 22-4: WDT Control Register 390
Table 22-5: WDT Reset Register 393
Table 22-6: WDT Clock Source Select Register 393
Table 22-7: WDT Count Register....393
Table 23-1: Pulse Train Engine Register Summary 397
Table 23-2: Pulse Train Engine Global Enable/Disable Register 398
Table 23-3: Pulse Train Engine Resync Register....398
Table 23-4: Pulse Train Engine Stopped Interrupt Flag Register 399
Table 23-5: Pulse Train Engine Interrupt Enable Register 400
Table 23-6: Pulse Train Engine Safe Enable Register 401
Table 23-7: Pulse Train Engine Safe Disable Register 401
Table 23-8: Pulse Train Engine Configuration Register....402
Table 23-9: Pulse Train Mode Bit Pattern Register....403
Table 23-10: Pulse Train n Loop Configuration Register....403
Table 23-11: Pulse Train n Automatic Restart Configuration Register 403
Table 24-1: MAX78002 CRC Instances 405
Table 24-2: Organization of Calculated Result in the CRC_VAL.value Field....406
Table 24-3: Common CRC Polynomials....406
Table 24-4: CRC Register Summary....408
Table 24-5: CRC Control Register....409
Table 24-6: CRC 8-Bit Data Input Register 409
Table 24-7: CRC 16-Bit Data Input Register 409
Table 24-8: CRC 32-Bit Data Input Register 409
Table 24-9: CRC Polynomial Register 410
Table 24-10: CRC Value Register....410
Table 25-1: MAX78002 AES Instances 411
Table 25-2: Interrupt Events....413
Table 25-3: AES Register Summary....414
Table 25-4: AES Control Register 414
Table 25-5: AES Status Register 415
Table 25-6: AES Interrupt Flag Register 415
Table 25-7: AES Interrupt Enable Register....416
Table 25-8: AES FIFO Register....416
Table 26-1: TRNG Register Summary....417
Table 26-2: TRNG Control Register 417
Table 26-3: TRNG Status Register 417
Table 26-4: TRNG Data Register 418
Table 27-1: MAX78002 SDHC Alternate Function Names to SDHC Specification Pin Names....420
Table 27-2: Registers Used to Generate SD Commands....423
Table 27-3: SDHC Register Offsets, Names and Descriptions....423
Table 27-4: SDHC SDMA System Address / Argument Register....425
Table 27-5: SDHC SDMA Block Size Register....425
Table 27-6: SDHC SDMA Block Count Register 427
Table 27-7: SDHC SDMA Argument 1 Register 427
Table 27-8: SDHC SDMA Transfer Mode Register....427
Table 27-9: Summary of how register settings determine type of data transfer 429
Table 27-10: SDHC Command Register....429
Table 27-11: Relationship between Parameters and the Name of Response Type....429
Table 27-12: SDHC Response 0 Register 430
Table 27-13: SDHC Response 1 Register 430
Table 27-14: SDHC Response 2 Register 430
Table 27-15: SDHC Response 3 Register 430

Table 27-16: SDHC Response 4 Register 431
Table 27-17: SDHC Response 5 Register 431
Table 27-18: SDHC Response 6 Register 431
Table 27-19: SDHC Response 7 Register 431
Table 27-20: SDHC Response Register Mapping to SD Host Controller Response Register Convention 431
Table 27-21: Kind of SD Card Response Mapping to SDHC Response Registers 432
Table 27-22: SDHC Buffer Data Port Register 432
Table 27-23: SDHC Present State Register 432
Table 27-24: SDHC Host Control 1 Register 435
Table 27-25: SDHC Power Control Register 435
Table 27-26: SDHC Block Gap Control Register 436
Table 27-27: SDHC Wakeup Control Register 437
Table 27-28: SDHC Clock Control Register 438
Table 27-29: SDHC Timeout Control Register 439
Table 27-30: SDHC Software Reset Register 439
Table 27-31: SDHC Normal Interrupt Status Register 441
Table 27-32: Transfer Complete and Data Timeout Error Priority and Status 442
Table 27-33: Command Complete and Command Timeout Error Priority and Status 443
Table 27-34: SDHC Error Interrupt Status Register 443
Table 27-35: SDHC Normal Interrupt Status Register 445
Table 27-36: SDHC Error Interrupt Status Enable Register 446
Table 27-37: SDHC Normal Interrupt Signal Enable Register 446
Table 27-38: SDHC Error Interrupt Signal Enable Register 447
Table 27-39: SDHC Auto CMD Error Status Register 448
Table 27-40: SDHC Host Control 2 Register 449
Table 27-41: SDHC Capabilities Register 0 450
Table 27-42: SDHC Capabilities Register 1 452
Table 27-43: SDHC Maximum Current Capabilities Register 453
Table 27-44: SDHC Force Event Register for Auto CMD Error Status Register 453
Table 27-45: SDHC Force Event Register for Error Interrupt Status 453
Table 27-46: SDHC ADMA Error Status Register 454
Table 27-47: SDHC ADMA System Address Register 0 455
Table 27-48: SDHC ADMA System Address Register 1 455
Table 27-49: Preset Value Register Example 456
Table 27-50: Preset Value Register Selection Conditions 456
Table 27-51: SDHC Preset Value 0 to Preset Value 7 Registers 457
Table 27-52: SDHC Slot Interrupt Status Register 457
Table 27-53: SDHC Host Controller Version Register 457
Table 29-1: MAX78002 Bootloader Instances 462
Table 29-2: MAX78002 Bootloader Operating States and Prompts 462
Table 29-3: PERMLOCK Command Summary 463
Table 29-4: CHALLENGE Command Summary 467
Table 29-5: MAX78002 General Command Summary 468
Table 29-6: L - Load 468
Table 29-7: P – Page Erase 469
Table 29-8: V – Verify 470
Table 29-9: LOCK – Lock Device 471
Table 29-10: PLOCK – Permanent Lock 472
Table 29-11: UNLOCK – Unlock Device 473
Table 29-12: H – Check Device 474
Table 29-13: I – Get ID 475
Table 29-14: S – Status 476
Table 29-15: Q – Quit 477

Table 29-16: MAX78002 Secure Command Summary 478
Table 29-17: LK – Load Application Key 478
Table 29-18: LK – Load Challenge Key 479
Table 29-19: VK – Verify Application Key 480
Table 29-20: VC – Verify Challenge Key 481
Table 29-21: AK – Activate Application Key 482
Table 29-22: AC – Activate Challenge Key 483
Table 29-23: WL – Write Code Length 484
Table 29-24: MAX78002 Challenge/Response Command Summary 485
Table 29-25: GC – Get Challenge 485
Table 29-26: SR – Send Response 486

Table of Equations

Equation 4-1: System Clock Scaling 46
Equation 7-1: FLC Clock Frequency 136
Equation 11-1: ADC Clock Generation 169
Equation 11-2: Sample Clock Frequency Calculation 170
Equation 11-3: T_{TRACK} Calculation 170
Equation 11-4: T_{HOLD} Calculation 170
Equation 11-5: Temperature Conversion Equation 187
Equation 12-1: UART Transmit FIFO Half-Empty Condition 207
Equation 12-2: UART Clock Divisor Formula 209
Equation 12-3: LPUART Clock Divisor Formula for UARTn_CTRL.fdm = 1 210
Equation 13-1: SPI Peripheral Clock 226
Equation 13-2: SCK High Time 226
Equation 13-3: SCK Low Time 227
Equation 14-1: I^2C Clock Frequency 242
Equation 14-2: I^2C Clock High Time Calculation 242
Equation 14-3: I^2C Clock Low Time Calculation 242
Equation 14-4: I^2C Target SCL Frequency 243
Equation 14-5: Determining the I2Cn_HSCLK.lo Register Value 243
Equation 14-6: Determining the I2Cn_HSCLK.hi Register Value 244
Equation 14-7: The Calculated Frequency of the I^2C Bus Clock Using the Results of Equation 14-5 and Equation 14-6 244
Equation 14-8: I^2C Timeout Maximum 256
Equation 14-9: I^2C Timeout Minimum 256
Equation 14-10: DMA Burst Size Calculation for I^2C Transmit 257
Equation 14-11: DMA Burst Size Calculation for I^2C Receive 257
Equation 15-1: CD Audio Bit Frequency Calculation 275
Equation 15-2: Calculating the Bit Clock Frequency for Audio 276
Equation 15-3: Controller Mode BCLK Generation Using the I^2S External Clock 276
Equation 15-4: Controller Mode Clock Divisor Calculation for a Target Bit Clock Frequency 276
Equation 15-5: Bits Per Word Calculation 276
Equation 15-6: LRCLK Frequency Calculation 276
Equation 15-7: Sample Size Relationship Bits per Word 282
Equation 15-8: Transmit FIFO Half-Empty Condition 283
Equation 18-1: OWM 1MHz Clock Frequency 318
Equation 20-1: Timer Peripheral Clock Equation 345
Equation 20-2: One-shot Mode Timer Period 351
Equation 20-3: Continuous Mode Timer Period 353
Equation 20-4: Counter Mode Maximum Clock Frequency 355
Equation 20-5: Counter Mode Timer Input Transitions 356

Equation 20-6: Timer PWM Period 358
Equation 20-7: Timer PWM Output High Time Ratio with Polarity 0 358
Equation 20-8: Timer PWM Output High Time Ratio with Polarity 1 358
Equation 20-9: Capture Mode Elapsed Time Calculation in Seconds 359
Equation 20-10: Capture Mode Elapsed Time Calculation in Seconds 361
Equation 20-11: Compare Mode Timer Period 361
Equation 20-12: Capture Mode Elapsed Time 365
Equation 21-1: Wake-Up Timer Clock Frequency 376
Equation 21-2: One-Shot Mode Timer Period 378
Equation 21-3: Continuous Mode Timer Period 379
Equation 21-4: Compare Mode Timer Initial Period 379
Equation 23-1: Pulse Train Mode Output Function 395
Equation 27-1: SDHC Peripheral Clock 421

1. Introduction

For ordering information, mechanical and electrical characteristics for the MAX78002 family of devices please refer to the data sheet.

1.1 Related Documentation

The MAX78002 data sheet and errata are available from the Analog Devices website, http://www.maximintegrated.com/MAX78002.

1.2 Document Conventions

1.2.1 Number Notations

Notation	Description
0xNN	Hexadecimal (Base 16) numbers are preceded by the prefix 0x.
0bNN	Binary (Base 2) numbers are preceded by the prefix 0b.
NN	Decimal (Base 10) numbers are represented using no additional prefix or suffix.
V[X:Y]	Bit field representation of a register, field, or value (V) covering Bit X to Bit Y.
Bit N	Bits are numbered in little-endian format; that is, the least significant bit of a number is referred to as Bit 0.
[0xNNNN]	An address offset from a base address is shown in bracket form.

1.2.2 Register and 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 1-1*.

Table 1-1: Field Access Definitions

Access Type	Definition
RO	**Reserved** This access type is reserved for static fields. Reads of this field return the reset value. Writes are ignored.
DNM	**Reserved. Do Not Modify** Software must first read this field and write the same value whenever writing to this register.
R	**Read Only** Reads of this field return a value. Writes to the field do not affect device operation.
W	**Write Only** Reads of this field return indeterminate values. Writes to the field change the field's state to the value written and can affect device operation.
R/W	**Unrestricted Read/Write** Reads of this field return a value. Writes to the field change the field's state to the value written and can affect device operation.
RC	**Read to Clear** Reading this field clears the field to 0. Writes to the field do not affect device operation.
RS	**Read to Set** Reading this field sets the field to 1. Writes to the field do not affect device operation.
R/W0O	**Read/Write 0 Only** Writing 0 to this field set the field to 0. Writing 1 to the field does not affect device operation.
R/W1O	**Read/Write 1 Only** Writing 1 to this field sets the field to 1. Writing 0 to the field does not affect device operation.

Access Type	Definition
R/W1C	**Read/Write 1 to Clear** Writing 1 to this field clears this field to 0. Writing 0 to the field does not affect device operation.
R/W0S	**Read/Write 0 to Set** Writing 0 to this field sets this field to 1. Writing 1 to the field does not affect device operation.

1.2.3 Register Lists

Each peripheral includes a table listing all of the peripheral's registers. The register table includes the offset, register name, and description of each register. The offset shown in the table must be added to the peripheral's base address in *Table 3-3* to get the register's absolute address.

Table 1-2: Example Registers

Offset	Register Name	Description
[0x0000]	REG_NAME0	Name 0 Register

1.2.4 Register Detail Tables

Each register in a peripheral includes a detailed register table, as shown in *Table 1-3*. The first row of the register detail table includes the register's description, the register's name, and the register's offset from the base peripheral address. The second row of the table is the header for the bit fields represented in the register. The third and subsequent rows of the table include the bit or bit range, the field name, the bit's or field's access, the reset value, and a description of the field. All registers are 32-bits unless specified otherwise. Reserved bits and fields are shown as **Reserved** in the description column. See *Table 1-1* for a list of all access types for each bit and field.

Table 1-3: Example Name 0 Register

<table>
<tr><th colspan="4">Name 0</th><th colspan="1">REG_NAME0 [0x0000]</th></tr>
<tr><th>Bits</th><th>Field</th><th>Access</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:16</td><td>-</td><td>RO</td><td>-</td><td><b>Reserved</b></td></tr>
<tr><td>15:0</td><td>field_name</td><td>R/W</td><td>0</td><td><b>Field name description</b>
Description of <i>field_name</i>.</td></tr>
</table>

2. Overview

Artificial intelligence (AI) requires extreme computational horsepower, but Analog Devices is cutting the power cord from AI insights. The MAX78002 is a new breed of AI microcontroller built to enable neural networks to execute at ultra-low power and live at the edge of the IoT. This product combines the most energy-efficient AI processing with Analog Devices' proven ultra-low-power microcontrollers. The hardware-based convolutional neural network (CNN) accelerator enables battery-powered applications to execute AI inferences while spending only microjoules of energy.

The MAX78002 is an advanced system-on-chip featuring an Arm® Cortex®-M4 with FPU CPU for efficient system control with an ultra-low-power deep neural-network accelerator. The CNN engine has a weight storage memory of 2MB, and can support 1-, 2-, 4-, and 8-bit weights (supporting networks of up to 16 million weights). The CNN weight memory is SRAM-based so that AI network updates can be made on the fly. The CNN engine also has 1.3MB of data memory. The CNN architecture is highly flexible, allowing networks to be trained in conventional toolsets like PyTorch and TensorFlow®, then converted for execution on the MAX78002 using tools provided by Maxim.

In addition to the memory in the CNN engine, the MAX78002 has large on-chip system memory for the microcontroller core, with 2.5MB flash and up to 384KB SRAM. Multiple high-speed and low-power communications interfaces are supported, including I^{2}S, MIPI® CSI-2 serial camera, parallel camera (PCIF), and SD3.0/SDIO3.0/eMMC4.51 secure digital. For information on the Arm Cortex-M4 with FPU core, please refer to the *Arm Cortex-M4 Processor Technical Reference Manual*.

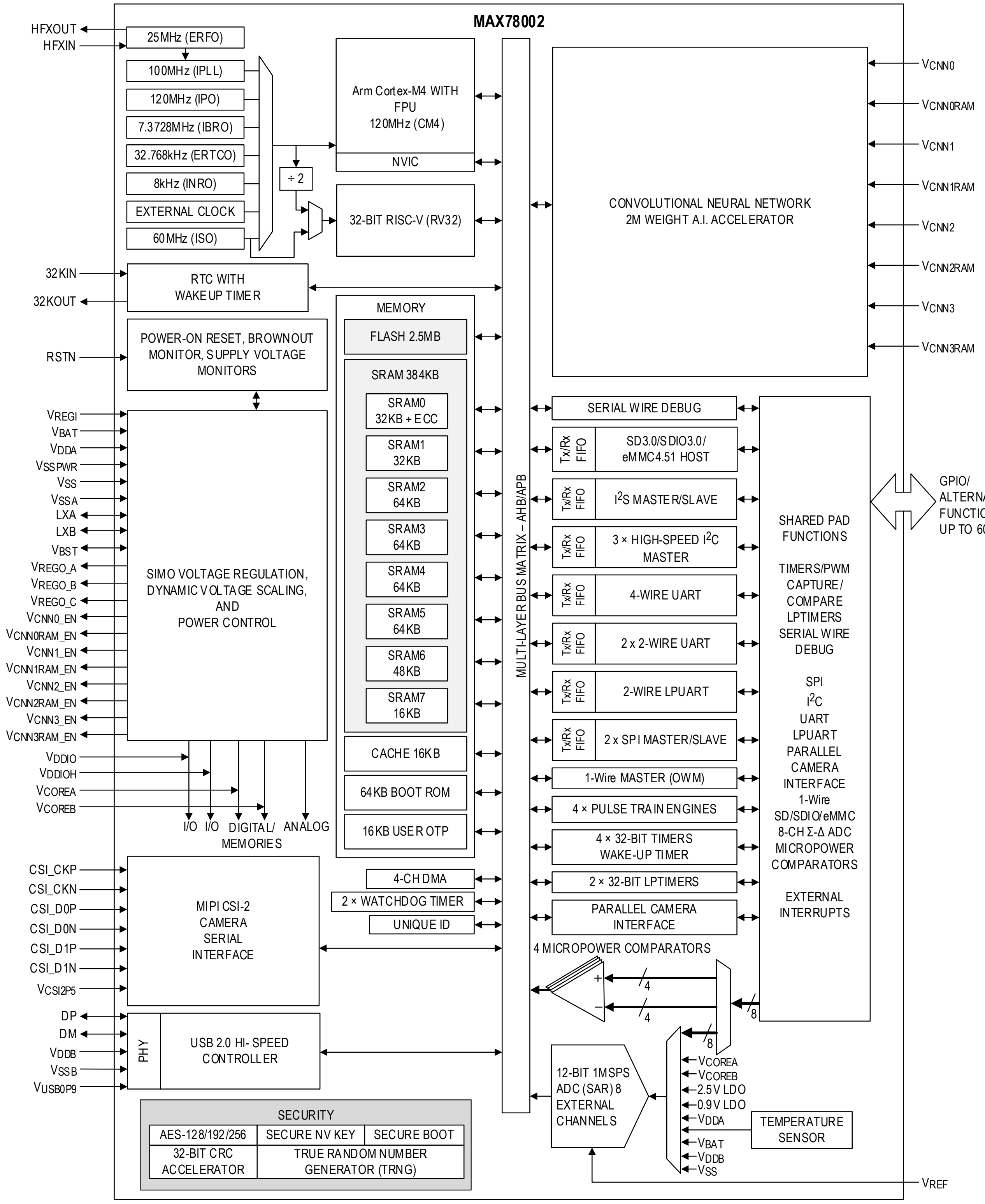
The high-level block diagram for the MAX78002 is shown in *Figure 2-1*.

Arm is a registered trademark and registered service mark of Arm Limited.

Cortex is a registered trademark of Arm Limited.

2.1 Block Diagram

Figure 2-1: MAX78002 Block Diagram



3. Memory, Register Mapping, and Access

3.1 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.

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

Figure 3-1: CM4 Code Memory Mapping

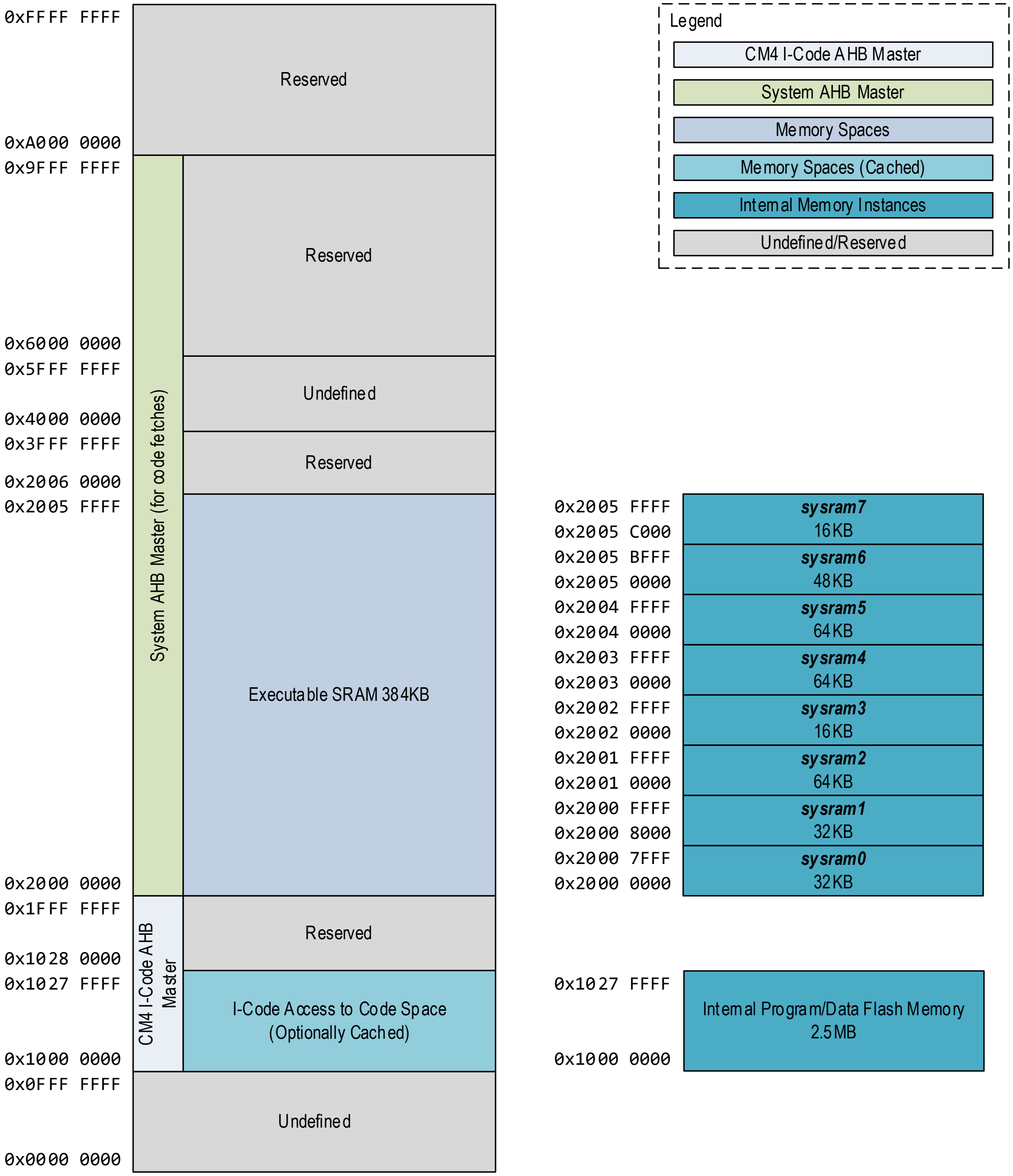


Figure 3-2: RISC-V IBUS Code Memory Mapping

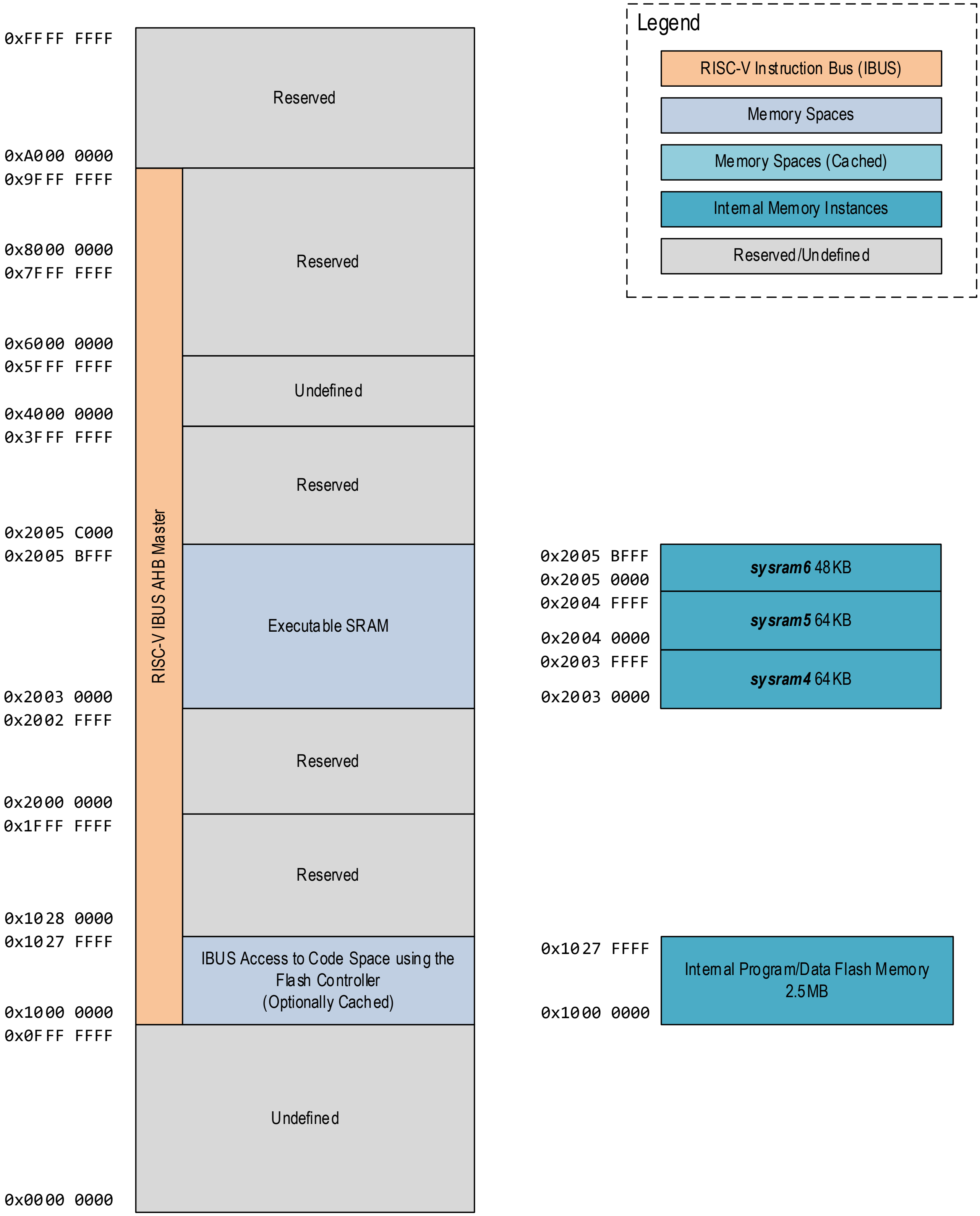
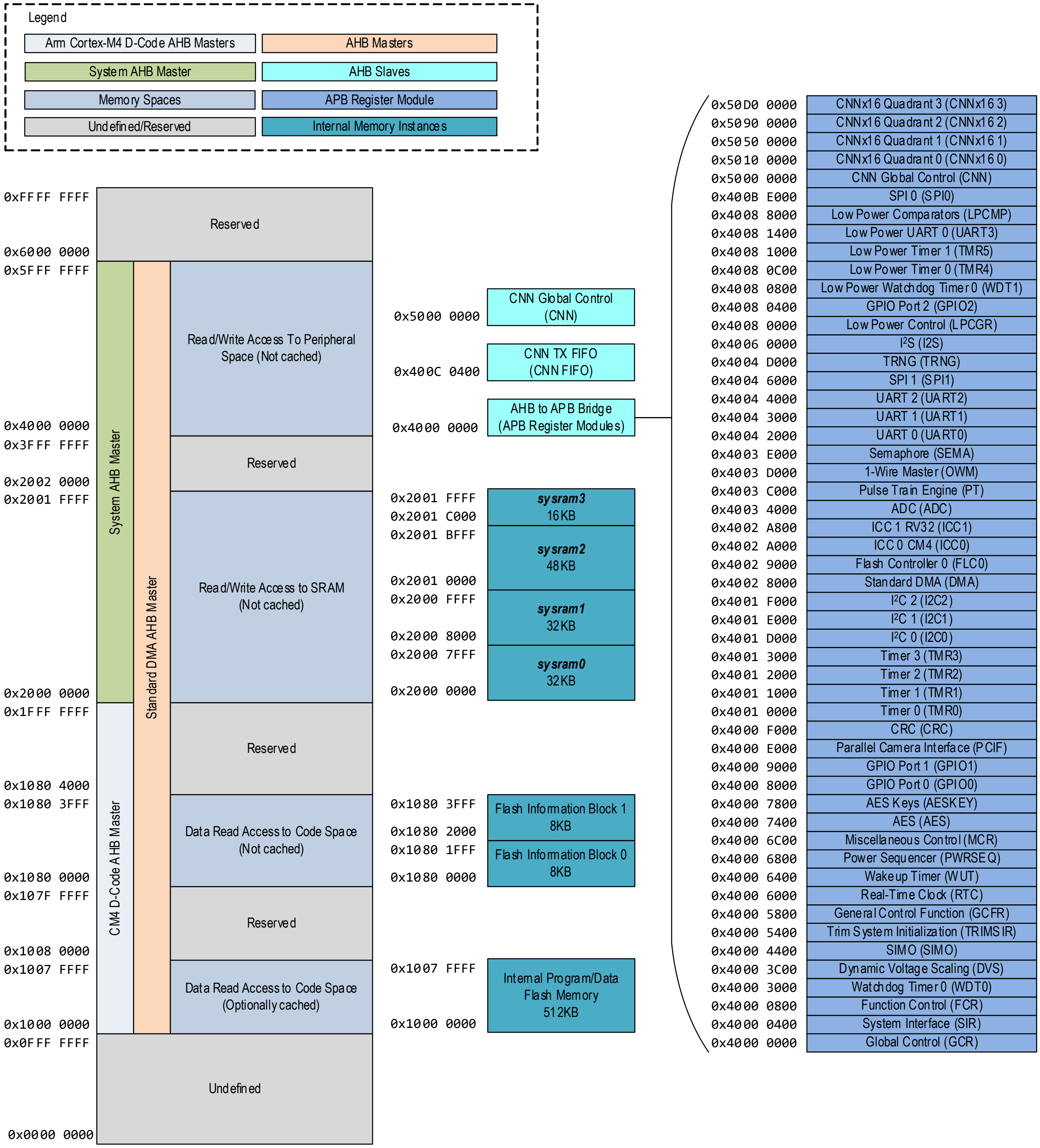
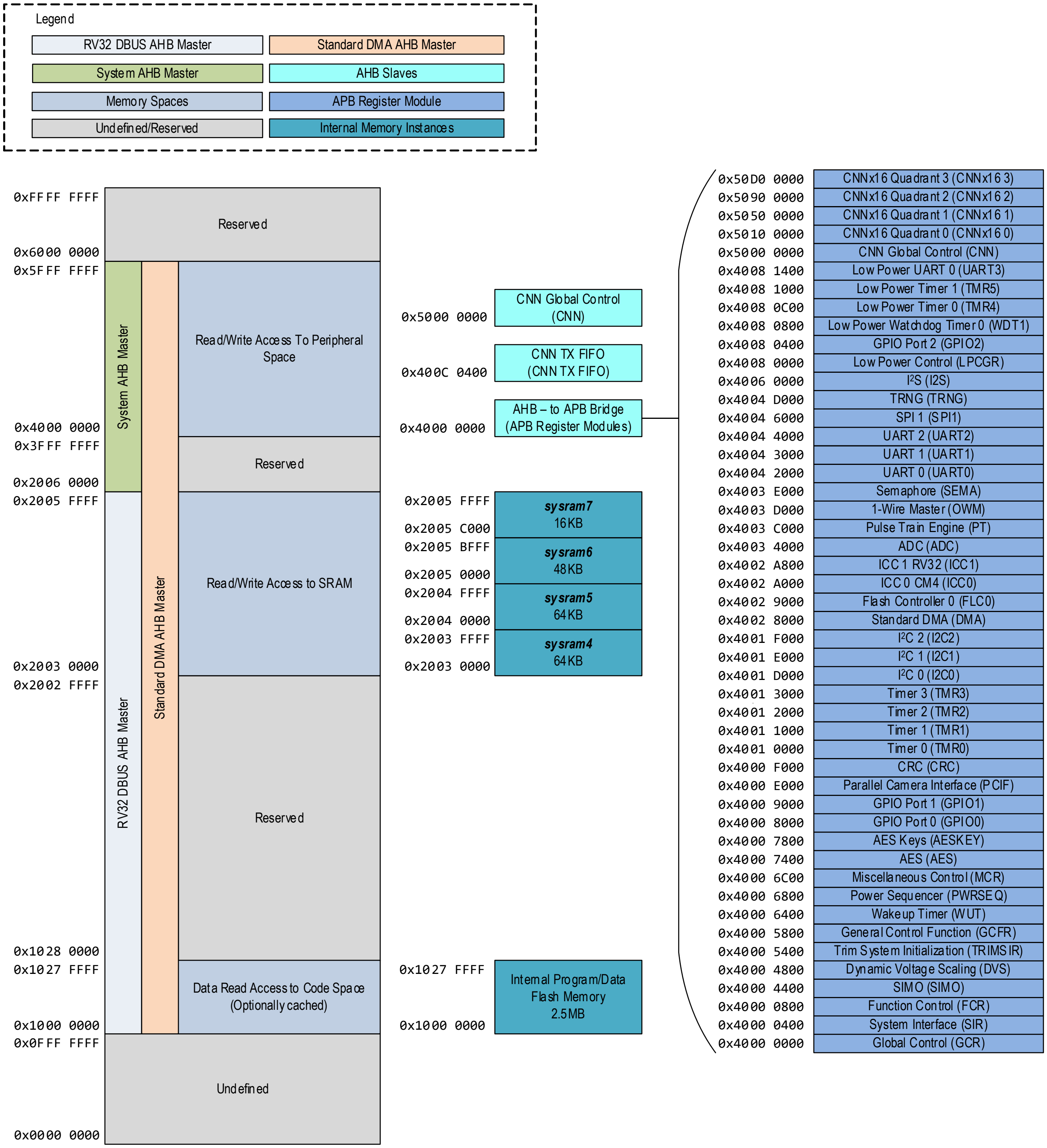


Figure 3-3: CM4 Peripheral and Data Memory Mapping



0x50D0 0000	CNNx16 Quadrant 3 (CNNx16 3)
0x5090 0000	CNNx16 Quadrant 2 (CNNx16 2)
0x5050 0000	CNNx16 Quadrant 1 (CNNx16 1)
0x5010 0000	CNNx16 Quadrant 0 (CNNx16 0)
0x5000 0000	CNN Global Control (CNN)
0x400B E000	SPI0 (SPI0)
0x4008 8000	Low Power Comparators (LPCMP)
0x4008 1400	Low Power UART 0 (UART3)
0x4008 1000	Low Power Timer 1 (TMR5)
0x4008 0C00	Low Power Timer 0 (TMR4)
0x4008 0800	Low Power Watchdog Timer 0 (WDT1)
0x4008 0400	GPIO Port 2 (GPIO2)
0x4008 0000	Low Power Control (LPCGR)
0x4006 0000	I^2S (I2S)
0x4004 D000	TRNG (TRNG)
0x4004 6000	SPI1 (SPI1)
0x4004 4000	UART 2 (UART2)
0x4004 3000	UART 1 (UART1)
0x4004 2000	UART 0 (UART0)
0x4003 E000	Semaphore (SEMA)
0x4003 D000	1-Wire Master (OWM)
0x4003 C000	Pulse Train Engine (PT)
0x4003 4000	ADC (ADC)
0x4002 A800	ICC1 RV32 (ICC1)
0x4002 A000	ICC0 CM4 (ICC0)
0x4002 9000	Flash Controller 0 (FLC0)
0x4002 8000	Standard DMA (DMA)
0x4001 F000	I^2C 2 (I2C2)
0x4001 E000	I^2C 1 (I2C1)
0x4001 D000	I^2C 0 (I2C0)
0x4001 3000	Timer 3 (TMR3)
0x4001 2000	Timer 2 (TMR2)
0x4001 1000	Timer 1 (TMR1)
0x4001 0000	Timer 0 (TMR0)
0x4000 F000	CRC (CRC)
0x4000 E000	Parallel Camera Interface (PCIF)
0x4000 9000	GPIO Port 1 (GPIO1)
0x4000 8000	GPIO Port 0 (GPIO0)
0x4000 7800	AES Keys (AESKEY)
0x4000 7400	AES (AES)
0x4000 6C00	Miscellaneous Control (MCR)
0x4000 6800	Power Sequencer (PWRSEQ)
0x4000 6400	Wakeup Timer (WUT)
0x4000 6000	Real-Time Clock (RTC)
0x4000 5800	General Control Function (GCFR)
0x4000 5400	Trim System Initialization (TRIMSIR)
0x4000 4400	SIMO (SIMO)
0x4000 3C00	Dynamic Voltage Scaling (DVS)
0x4000 3000	Watchdog Timer 0 (WDT0)
0x4000 0800	Function Control (FCR)
0x4000 0400	System Interface (SIR)
0x4000 0000	Global Control (GCR)

Figure 3-4: RV32 Peripheral and Data Memory Mapping


Legend
RV32 DBUS AHB Master
Standard DMA AHB Master
System AHB Master
AHB Slaves
Memory Spaces
APB Register Module
Undefined/Reserved
Internal Memory Instances
0xFFFF FFFF
Reserved
0x6000 0000
0x5FFF FFFF
System AHB Master
Read/Write Access To Peripheral Space
0x4000 0000
0x3FFF FFFF
Reserved
0x2006 0000
0x2005 FFFF
Standard DMA AHB Master
Read/Write Access to SRAM
0x2003 0000
0x2002 FFFF
RV32 DBUS AHB Master
Reserved
0x1028 0000
0x1027 FFFF
Data Read Access to Code Space (Optionally cached)
0x1000 0000
0x0FFF FFFF
Undefined
0x0000 0000
0x5000 0000
CNN Global Control (CNN)
0x400C 0400
CNN TX FIFO (CNN TX FIFO)
0x4000 0000
AHB – to APB Bridge (APB Register Modules)
0x2005 FFFF
sysram7
16KB
0x2005 C000
0x2005 BFFF
sysram6
48KB
0x2005 0000
0x2004 FFFF
sysram5
64KB
0x2004 0000
0x2003 FFFF
sysram4
64KB
0x2003 0000
0x1027 FFFF
Internal Program/Data Flash Memory
2.5MB
0x1000 0000
0x50D0 0000 CNNx16 Quadrant 3 (CNNx16 3)
0x5090 0000 CNNx16 Quadrant 2 (CNNx16 2)
0x5050 0000 CNNx16 Quadrant 1 (CNNx16 1)
0x5010 0000 CNNx16 Quadrant 0 (CNNx16 0)
0x5000 0000 CNN Global Control (CNN)
0x4008 1400 Low Power UART 0 (UART3)
0x4008 1000 Low Power Timer 1 (TMR5)
0x4008 0C00 Low Power Timer 0 (TMR4)
0x4008 0800 Low Power Watchdog Timer 0 (WDT1)
0x4008 0400 GPIO Port 2 (GPIO2)
0x4008 0000 Low Power Control (LPCGR)
0x4006 0000 I²S (I2S)
0x4004 D000 TRNG (TRNG)
0x4004 6000 SPI 1 (SPI1)
0x4004 4000 UART 2 (UART2)
0x4004 3000 UART 1 (UART1)
0x4004 2000 UART 0 (UART0)
0x4003 E000 Semaphore (SEMA)
0x4003 D000 1-Wire Master (OWM)
0x4003 C000 Pulse Train Engine (PT)
0x4003 4000 ADC (ADC)
0x4002 A800 ICC 1 RV32 (ICC1)
0x4002 A000 ICC 0 CM4 (ICC0)
0x4002 9000 Flash Controller 0 (FLC0)
0x4002 8000 Standard DMA (DMA)
0x4001 F000 I²C 2 (I2C2)
0x4001 E000 I²C 1 (I2C1)
0x4001 D000 I²C 0 (I2C0)
0x4001 3000 Timer 3 (TMR3)
0x4001 2000 Timer 2 (TMR2)
0x4001 1000 Timer 1 (TMR1)
0x4001 0000 Timer 0 (TMR0)
0x4000 F000 CRC (CRC)
0x4000 E000 Parallel Camera Interface (PCIF)
0x4000 9000 GPIO Port 1 (GPIO1)
0x4000 8000 GPIO Port 0 (GPIO0)
0x4000 7800 AES Keys (AESKEY)
0x4000 7400 AES (AES)
0x4000 6C00 Miscellaneous Control (MCR)
0x4000 6800 Power Sequencer (PWRSEQ)
0x4000 6400 Wakeup Timer (WUT)
0x4000 5800 General Control Function (GCFR)
0x4000 5400 Trim System Initialization (TRIMSIR)
0x4000 4800 Dynamic Voltage Scaling (DVS)
0x4000 4400 SIMO (SIMO)
0x4000 0800 Function Control (FCR)
0x4000 0400 System Interface (SIR)
0x4000 0000 Global Control (GCR)

3.2 Standard Memory Regions

Several standard memory regions are defined for the Arm Cortex-M4 and RISC-V (CPU1) architectures; many of these are optional for the system integrator. At a minimum, the MAX78002 must contain code and data memory for software, stack, and variable space for the CM4.

3.2.1 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). The Cortex-M4 core and Arm debugger use two different standard core bus masters 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 MAX78002 code memory mapping is illustrated in *Figure 3-1* and *Figure 3-2*. The code space memory area contains the main internal flash memory, which holds the software executed on the device. The internal flash memory is mapped into both code and data space from 0x1000 0000 to 0x1027 FFFF. The main program flash memory is 2.5MB.

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 that transfers execution to user code at address 0x1000 0000.

The code space memory on the MAX78002 also contains the mapping for the flash information block, from 0x1080 0000 to 0x1080 3FFF. However, this mapping is only present during 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. See *Information Block Flash Memory* for additional details.

3.2.2 Internal Cache Memory

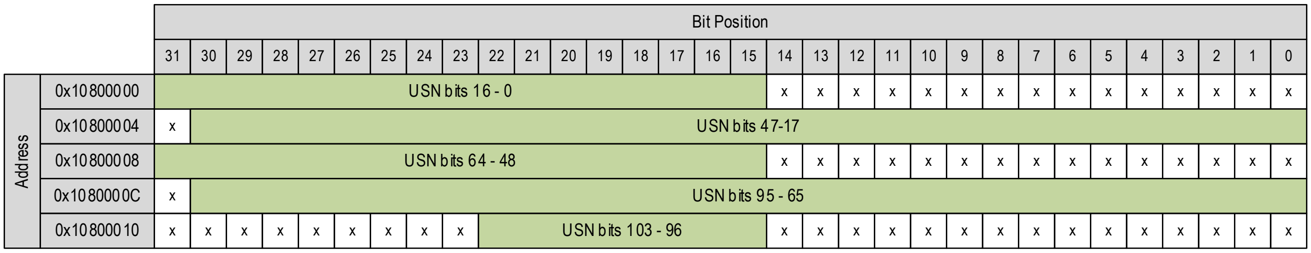
The MAX78002 includes a dedicated unified internal cache controller with 16,384 bytes of internal cache memory (ICC0) for the CM4 core. Optionally, *sysram7* can be used as a unified internal cache controller (ICC1) for the RV32.

The unified internal cache memory is used to cache data and instructions fetched through the I-Code bus for the CM4 or the IBUS for the RV32 from the internal flash memory. See section *Unified Internal Cache Controller* for detailed instructions on enabling the unified internal cache controllers.

3.2.3 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) and other nonvolatile device-specific information. The information block also contains the device's Unique Serial Number (USN). The USN is a 104-bit field.

Figure 3-5: 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
	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	x	x	x	x	x

3.2.4 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, the CM4 stack, and the RV32 stack.

The MAX78002 CM4's data memory mapping is illustrated in *Figure 3-1*. The MAX78002 RV32's data memory mapping is illustrated in *Figure 3-4*.

The system SRAM configuration is defined in *Table 3-1*.

The SRAM area contains the main system RAM. The size of the internal general-purpose data SRAM is 384KB. The SRAM is divided into four blocks and consists of the contiguous address range from 0x2000 0000 to 0x2001 FFFF. The SRAM area on the MAX78002 can be used for data storage and code execution by the CM4. The RV32 is limited to *sysram2* and *sysram3* for code and data storage.

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 sysram4, sysram5, and sysram6.

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 on the device capable of this function.

Table 3-1: System SRAM Configuration

System RAM Block #	Size (KB)	Start Address	End Address	CM4 Accessible	RV32 Accessible
sysram0	32	0x2000 0000	0x2000 7FFF	✓	No
sysram1	32	0x2000 8000	0x2000 FFFF	✓	No
sysram2	64	0x2001 0000	0x2001 FFFF	✓	No
sysram3	64	0x2002 0000	0x2002 FFFF	✓	No
sysram4	64	0x2003 0000	0x2003 FFFF	Configurable	✓
sysram5	64	0x2004 0000	0x2004 FFFF	Configurable	✓
sysram6	48	0x2005 0000	0x2005 BFFF	Configurable	✓
sysram7	16	0x2005 C000	0x2005 FFFF	Configurable	✓ (Optional ICC1)

The MAX78002 specific AHB Bus Masters can access the SRAM to use as general storage or working space.

The entirety of the SRAM space on the MAX78002 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 384KB 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. Bit-banding is a core function (i.e., not a function of the SRAM interface layer or the AHB bus layer) and thus is only applicable to accesses generated by the core. Reads and writes to the bit-banding alias area by other (non-Arm-core) bus masters does not trigger a bit-banding operation and instead results in an AHB bus error.

3.2.5 Peripheral Space

The peripheral space area of memory is intended to map control registers, internal buffers, and other features needed for the software control of non-core peripherals. It is defined from byte address range 0x4000 0000 to 0x5FFF FFFF (0.5GB maximum). On the MAX78002, all device-specific module registers are mapped to this memory area and any local memory buffers or FIFOs that 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) 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 does not occur. In this case, the bit-banding alias region appears to be a non-implemented memory area (causing an AHB bus error).

On the MAX78002, access to the region containing most peripheral registers (0x4000 0000 to 0x400F FFFF) goes from the AHB bus through an AHB-to-APB bridge enabling 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.

3.2.6 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 software access.

3.2.7 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.

In addition to being restricted to the core, application software can only access this area when running in privileged execution mode (instead of 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 access this area.

Core functions controlled by registers mapped to this area include the SysTick timer, debug and tracing functions, the nested vector interrupt controller (NVIC), and the flash breakpoint controller.

3.3 AHB Interfaces

The following sections detail memory accessibility on the AHB and the organization of AHB master and slave instances.

3.3.1 Arm Core AHB Interfaces

3.3.1.1 I-Code

The Arm core uses the I-Code AHB master 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 cache, which in turn triggers a cache line fill cycle to fetch instructions from the internal flash memory when a cache miss occurs.

3.3.1.2 D-Code

The Arm core uses the D-Code AHB master for data fetches from memory instances 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.

3.3.1.3 System

The Arm core uses the system AHB master 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.

3.3.1.4 USB

The USB peripheral is an AHB master with access to all internal system RAM (*sysram0 - sysram7*).

3.3.1.5 SDHC

The SDHC peripheral is an AHB master with access to all internal system RAM (*sysram0 - sysram7*).

3.3.1.6 CSI-2

The CSI-2 peripheral is an AHB master with access to all internal system RAM (*sysram0 - sysram7*).

3.3.2 AHB Slaves

3.3.2.1 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.

3.3.2.2 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.

3.3.2.3 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.

3.3.3 AHB Slave Base Address Map

Table 3-2 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 3-2: 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_FIFO_	0x400C 0400	0x400C 04FF
MIPI CSI-2 FIFO	CSI2_FIFO	0x400C 0800	0x400C 0BFF

3.4 Peripheral Register Map

3.4.1 APB Peripheral Base Address Map

Table 3-3 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 3-3: 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	SIR_	0x4000 0400	0x4000 07FF
Function Control	FCR_	0x4000 0800	0x4000 0BFF
Watchdog Timer 0	WDT0_	0x4000 3000	0x4000 33FF
Single Input Multiple Output	SIMO_	0x4000 4400	0x4000 47FF
Dynamic Voltage Scaling Controller	DVS_	0x4000 4800	0x4000 4BFF
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	AESKEY_	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	FLC_	0x4002 9000	0x4002 93FF
Instruction-Cache Controller 0 (CM4)	ICC0_	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
1-Wire Master	OWM_	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
I^{2}S	I2S_	0x4006 0000	0x4006 0FFF
MIPI Camera Serial Interface 2	CSI2_	0x4006 2000	0x4006 2FFF
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

Peripheral Register Name	Register Prefix	APB Base Address	APB End Address
Low Power Timer 0 (TMR4)	TMR4_	0x4008 0C00	0x4008 0FFF
Low Power Timer 1 (TMR5)	TMR5_	0x4008 1000	0x4008 13FF
Low Power UART 0 (UART3)	UART3_	0x4008 1400	0x4008 17FF
Low Power Comparator 0	LPCMP_	0x4008 8000	0x4008 83FF
USB	USBHS_	0x400B 1000	0x400B 1FFF
SDHC	SDHC_	0x400B 6000	0x400B 6FFF
SPI 0	SPI0_	0x400B E000	0x400B EFFF
Trim System Initialization	TRIMSIR_	0x4010 5400	0x4010 54FF
CNN	CNN_	0x5000 0000	0x5FFF FFFF

3.5 Error Correction Coding (ECC) Module

This device features an Error Correction Coding (ECC) module that helps ensure data integrity by detecting and correcting bit corruption of the system RAM0 (*sysram0*) memory array. More specifically, the ECC module is a single error-correcting, double error detecting (SEC-DED). It corrects any single bit flip, detects two 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 is detected but not corrected.

If only one bit is determined to be corrupt, reads contain the "corrected" value. Reading memory does not correct the error value stored at the read memory location. It is up to the software to determine the appropriate time and method to write the correct data to memory. It is strongly recommended that the software 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 occurs only during a read operation, it is recommended that the application periodically reads critical memory so that errors can be identified and corrected.

3.5.1 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 unavailable for application usage.

3.5.2 Limitations

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

4. System, Power, Clocks, Reset

Different peripherals and subsystems use several clocks. These clocks are highly configurable by software, 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.

4.1 Oscillator Sources

4.1.1 120MHz Internal Primary Oscillator (IPO)

The MAX78002 includes a 120MHz internal high-speed oscillator, referred to in this document as the internal primary oscillator (IPO). The IPO is the highest 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. Use the IPO as the SYS_OSC by performing the following steps:

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.

4.1.2 60MHz Internal Secondary Oscillator (ISO)

The ISO is a low-power internal secondary oscillator that is the power-on reset default SYS_OSC. The ISO 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.

4.1.3 8kHz-30kHz Internal Nano-Ring Oscillator (INRO)

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

The frequency of this oscillator is configurable to 8kHz, 16kHz, or 30kHz. Use the *TRIMSIR_INRO.lpclksel* 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.

4.1.4 7.3728MHz Internal Baud Rate Oscillator (IBRO)

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

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.

4.1.5 100MHz/200MHz Internal Phase Lock Loop Oscillator (IPLL)

The IPLL is a very high speed internal PLL that operates from the external 25MHz crystal. The IPLL provides a 100MHz oscillator that can be used as the system clock as well as a 200MHz clock that can optionally be used for the CNN clock. The following steps show how to enable the IPLL and select it as the SYS_OSC.

1. Enable the IPLL by setting *GCR_IPLL_CTRL.en* to 1.
2. Wait until the *GCR_IPLL_CTRL.rdy* field reads 1, indicating the IPLL is operating.
3. Set *GCR_CLKCTRL.sysclk_sel* to 1.
4. Wait until the *GCR_CLKCTRL.sysclk_rdy* field reads 1. The IPLL is now operating as the SYS_OSC.

Note: Enabling the IPLL automatically enables the 25MHz external base oscillator.

4.1.6 25MHz External Base Oscillator (EBO)

The 25MHz EBO is an available external oscillator that can be selected as the SYS_OSC and is used for the IPLL. Additionally, the EBO can be used for the ADC clock.
The following steps show how to enable the EBO and select it as the SYS_OSC.

1. Enable the EBO by setting *GCR_CLKCTRL.ebo_en* to 1.
2. Wait until the *GCR_CLKCTRL.ebo_en* field reads 1, indicating the EBO is operating.
3. Set *GCR_CLKCTRL.sysclk_sel* to 2 to select EBO as the system clock.
4. Wait until the *GCR_CLKCTRL.sysclk_rdy* field reads 1. The EBO is now operating as the SYS_OSC.

4.1.7 32.768kHz External Real-Time Clock Oscillator (ERTCO)

The ERTCO 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. The ERTCO is disabled on a POR 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.

4.1.8 External Clock (EXT_CLK)

An external clock can be used as the SYS_OSC. The external clock supports clock frequencies up to 80MHz.

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

1. Set device pin P0.3 for AF1 mode:
   a. GPIO0_EN0.[3] = 0
   b. GPIO0_EN1.[3] = 0
   c. GPIO0_EN2.[3] = 0
2. Ensure the external clock is operating.
3. Set GCR_CLKCTRL.*sysclk_sel* to 7.
4. Wait until the GCR_CLKCTRL.*sysclk_rdy* field reads 1. The EXT_CLK is now operating as the SYS_OSC.

4.2 System Oscillator (SYS_OSC)

The MAX78002 supports multiple clock sources as the SYS_OSC. The selected SYS_OSC is the clock source for most internal blocks. Each oscillator, description, and nominal frequency are shown in *Table 4-1*. An external clock source, EXT_CLK, is supported on P0.3, alternate function 1. Each of the oscillators/clocks is described in detail in section *Oscillator Sources*.

Table 4-1: Available System Oscillators

Oscillator/Clock	Description	Nominal Frequency
IPO	*Internal Primary Oscillator*	120MHz
ISO	*Internal Secondary Oscillator*	60MHz
INRO	*Internal Nano-Ring Oscillator*	Configurable 8kHz, 16kHz, or 30kHz
IBRO	*Internal Baud Rate Oscillator*	7.3728MHz
ERTCO	*External Real-Time Clock Oscillator*	32.768kHz
IPLL	*Internal Phase Lock Loop*	100MHz
EBO	*External Base Oscillator*	25MHz
EXT_CLK	*External Clock*	Up to 80MHz

4.2.1 System Oscillator Selection

Set the system oscillator using the GCR_CLKCTRL.*sysclk_sel* field. Before selecting an oscillator as the system oscillator, the oscillator source must first be enabled and ready. See each 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. The application software must verify that the switchover is complete before continuing operation.

CAUTION: When switching SYS_OSC or modifying the SYS_OSC prescaler, any device peripherals using APB or AHB clock must be reconfigured for the new clock frequency.

4.2.2 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 4-1*.

Equation 4-1: System Clock Scaling

$$SYS\_CLK = \frac{SYS\_OSC}{2^{sysclk\_div}}$$

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 core, the RV32 core, and all AHB masters in the system. SYS_CLK generates the following internal clocks as shown below:

- AHB Clock
    - $HCLK = SYS\_CLK$
- APB Clock
- $PCLK = \frac{SYS\_CLK}{2}$

The RTC uses the ERTCO for its clock source. Optionally, the RTC can run using an internal dedicated 8kHz nano-ring oscillator. See 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 4-2 shows each oscillator's enabled state for each type of reset source in the MAX78002.

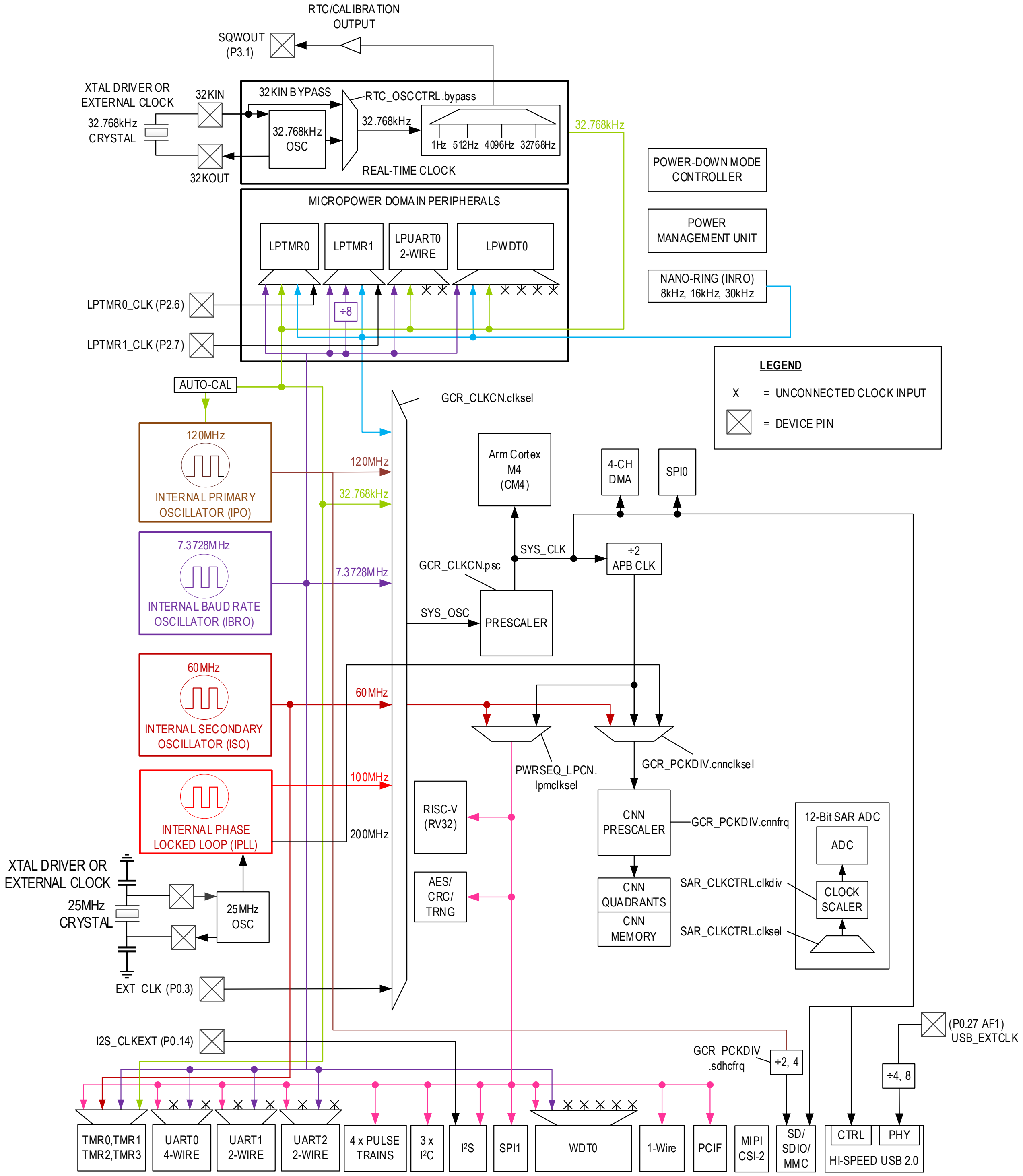
Note: A Watchdog Timer Reset performs a System Reset.

Table 4-2: Reset Sources and Effect on Oscillator and System Clock

	Reset Source			
Oscillator	**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
PLL	Disabled	Disabled	Retains State	Retains State
System Clock (SYS_OSC) Source	ISO	ISO	Retains State	Retains State

Figure 4-1 shows a high-level diagram of the MAX78002 clock tree.

Figure 4-1: MAX78002 Clock Block Diagram



4.3 Operating Modes

The MAX78002 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*)

4.3.1 ACTIVE Mode

In this mode, both the CM4 and the RV32 cores can execute software, 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 can be optionally 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.

4.3.2 Low-Power Modes

4.3.2.1 SLEEP

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 is individually configurable for power down
- Standard DMA is available for use
- All peripherals are on unless explicitly disabled before entering *SLEEP*

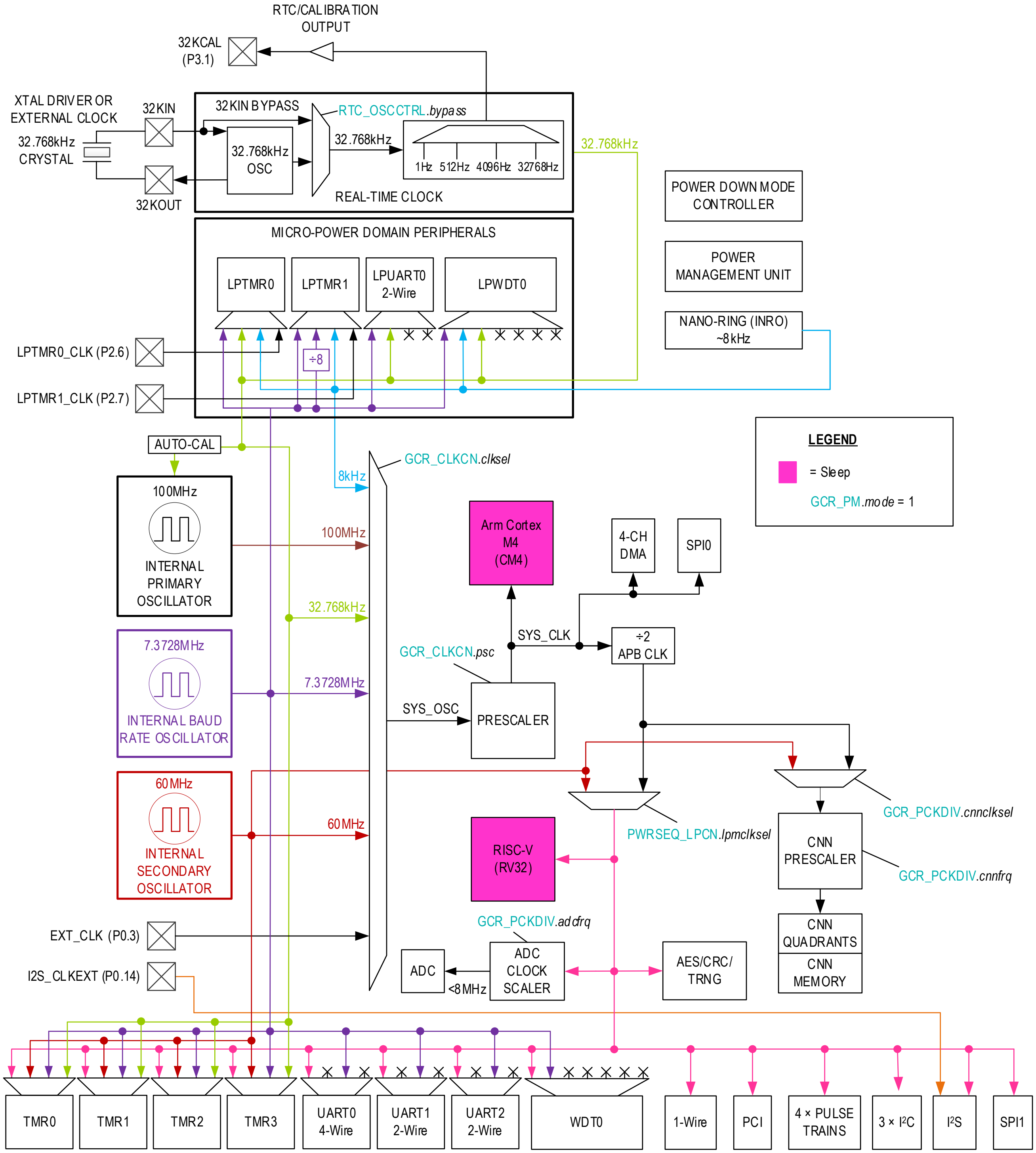
4.3.2.1.1 Entering SLEEP

Entering *SLEEP* requires both the CM4 and RV32 to cooperate to enter *SLEEP*. 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 confirmation through 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, followed by two NOP instructions.

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 through *Multiprocessor Communications* and performs a WFI instruction followed by two NOP instructions. 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 4-2: SLEEP Mode Clock Control



4.3.2.2 LPM

This mode is suitable for running the RV32 processor to collect and move data from enabled peripherals. The device status is a 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, UARTS, Timers, I^{2}C, 1-Wire, Timers, Pulse Train Engine, I^{2}S, 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_GP0* and *PWRSEQ_GP1* registers retain state
- Choose the system PCLK or ISO as the clock source for the RV32 and all peripherals
  - *PWRSEQ_LPCN*.*isoclk_select* defaults to use ISO during LPM. Setting this field to 1 uses the PCLK
- The following oscillators are powered down by default, but can be configured by software to remain active:
  - ISO
  - IPO
  - ERTCO
  - INRO
- The following oscillator is enabled:
  - IBRO

4.3.2.2.1 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*.

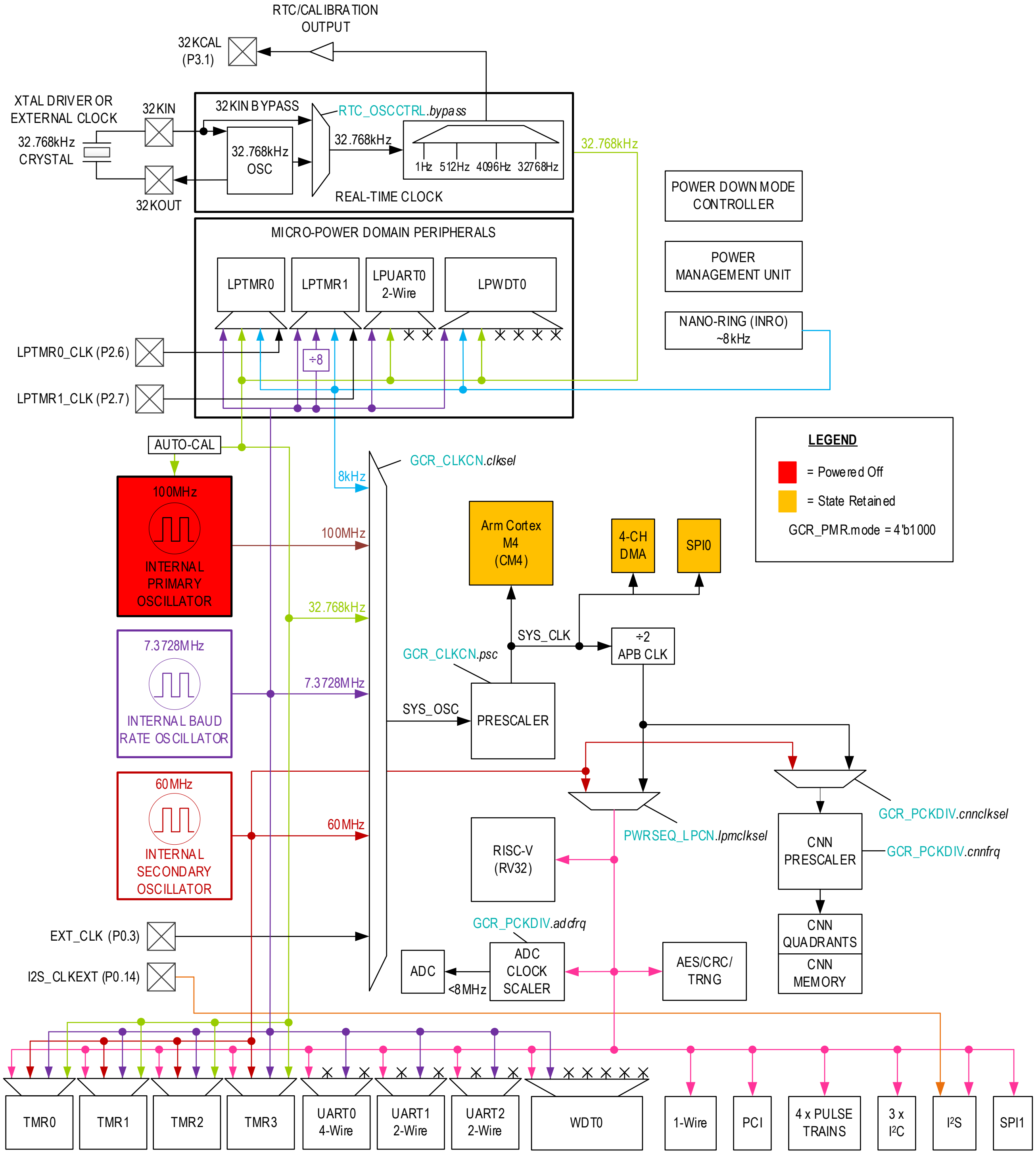
When the CM4 puts itself into *deep sleep*, the device automatically enters *LPM,* and hardware sets the *GCR_PM*.*mode* to *LPM*. To place the CM4 in *LPM* mode in software, perform the following instructions.

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

If the RV32 requests the CM4 to enter *LPM* mode through *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 4-3: LPM Clock and State Retention Diagram



4.3.2.3 UPM

This mode is used for extremely low power consumption while using a minimal set of peripherals to provide wake-up 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, firmware configurable
  - INRO, firmware configurable
- The following *UPM* peripherals are available for use to wake the device:
  - LPUART0
  - LPTMR0
  - LPTMR1
  - LPWDT0
  - LPCMP0-LPCMP3
  - GPIO

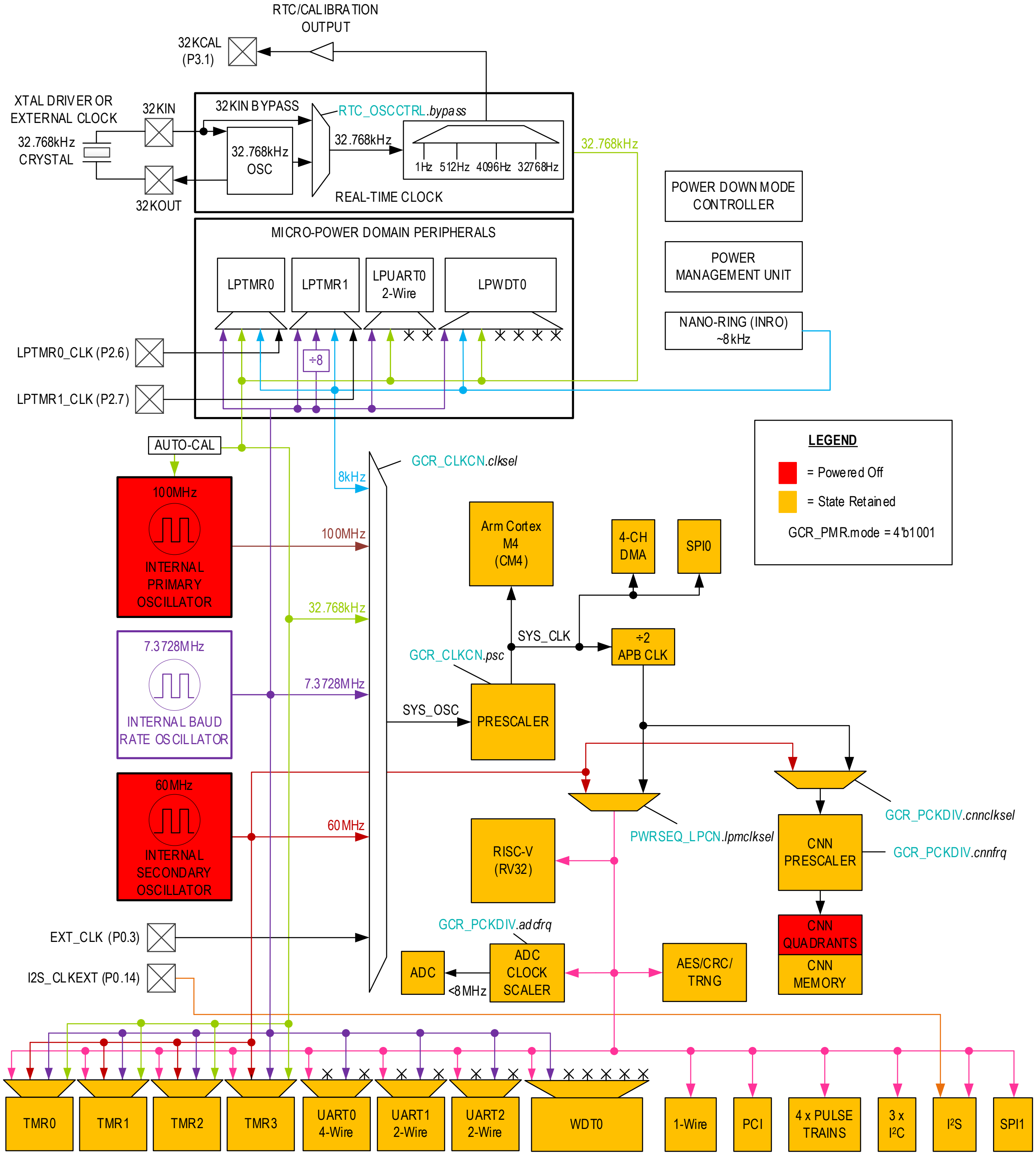
4.3.2.3.1 Entering UPM

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 through 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*, followed by two NOP instructions, and the device immediately enters *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 through *Multiprocessor Communications* and performs a WFI instruction, followed by two NOP instructions. The CM4 then sets the *GCR_PM*.*mode* to *UPM,* and the device immediately enters *UPM*.

Figure 4-4: UPM Clock and State Retention Block Diagram



4.3.2.4 STANDBY

This mode is used to maintain the 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
  - IBRO
- The following oscillators are enabled:
  - ERTCO, firmware configurable
  - INRO

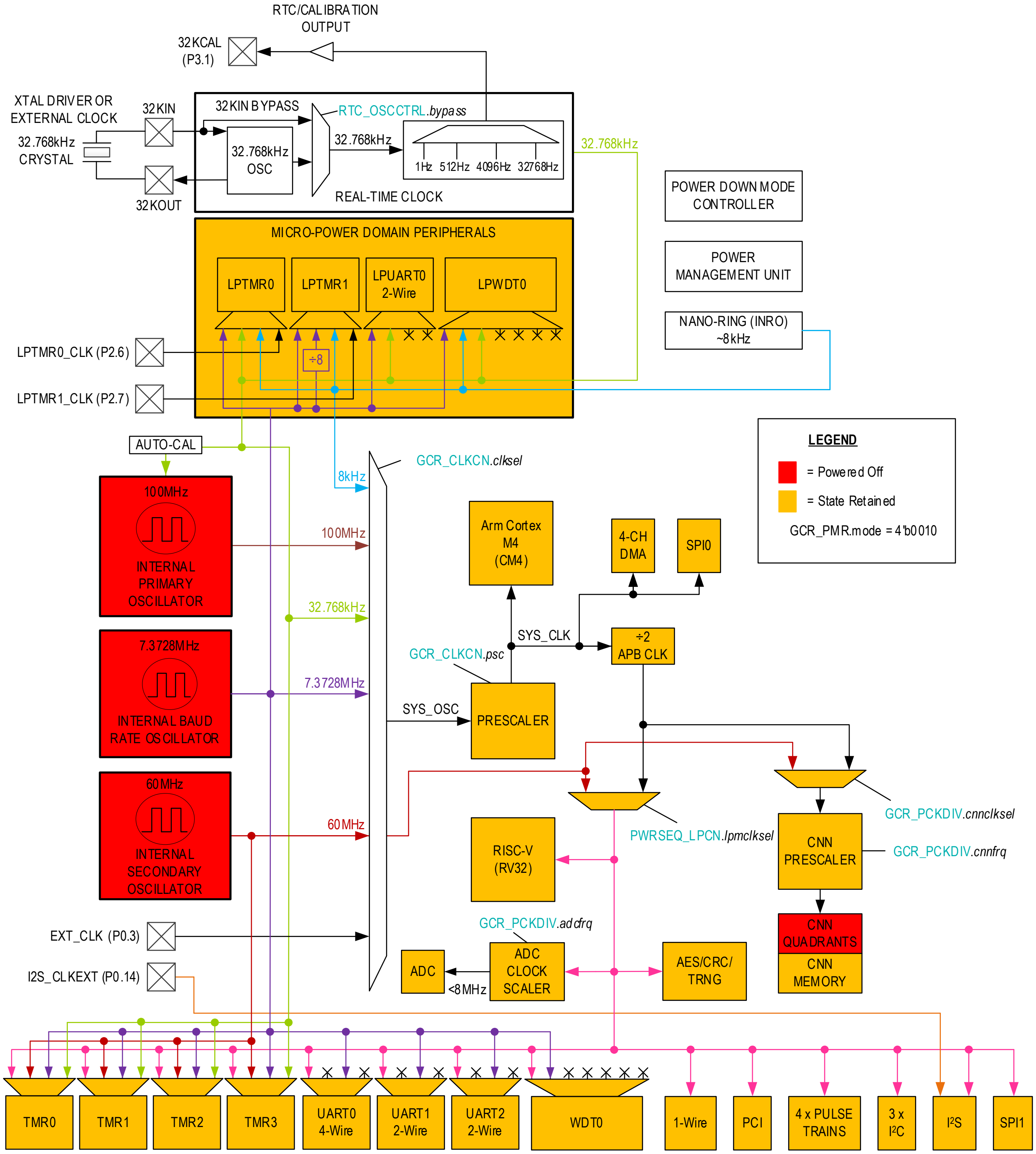
4.3.2.4.1 Entering STANDBY

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 through 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*, followed by two NOP instructions, 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 through *Multiprocessor Communications* and performs a WFI instruction followed by two NOP instructions. The CM4 then sets the *GCR_PM*.*mode* to *STANDBY,* and the device immediately enters *STANDBY*.

Figure 4-5: STANDBY Mode Clock and State Retention Block Diagram



4.3.2.5 BACKUP

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 for state retention, as shown in *Table 4-3*.
- 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 4-3 System RAM Retention in BACKUP Mode

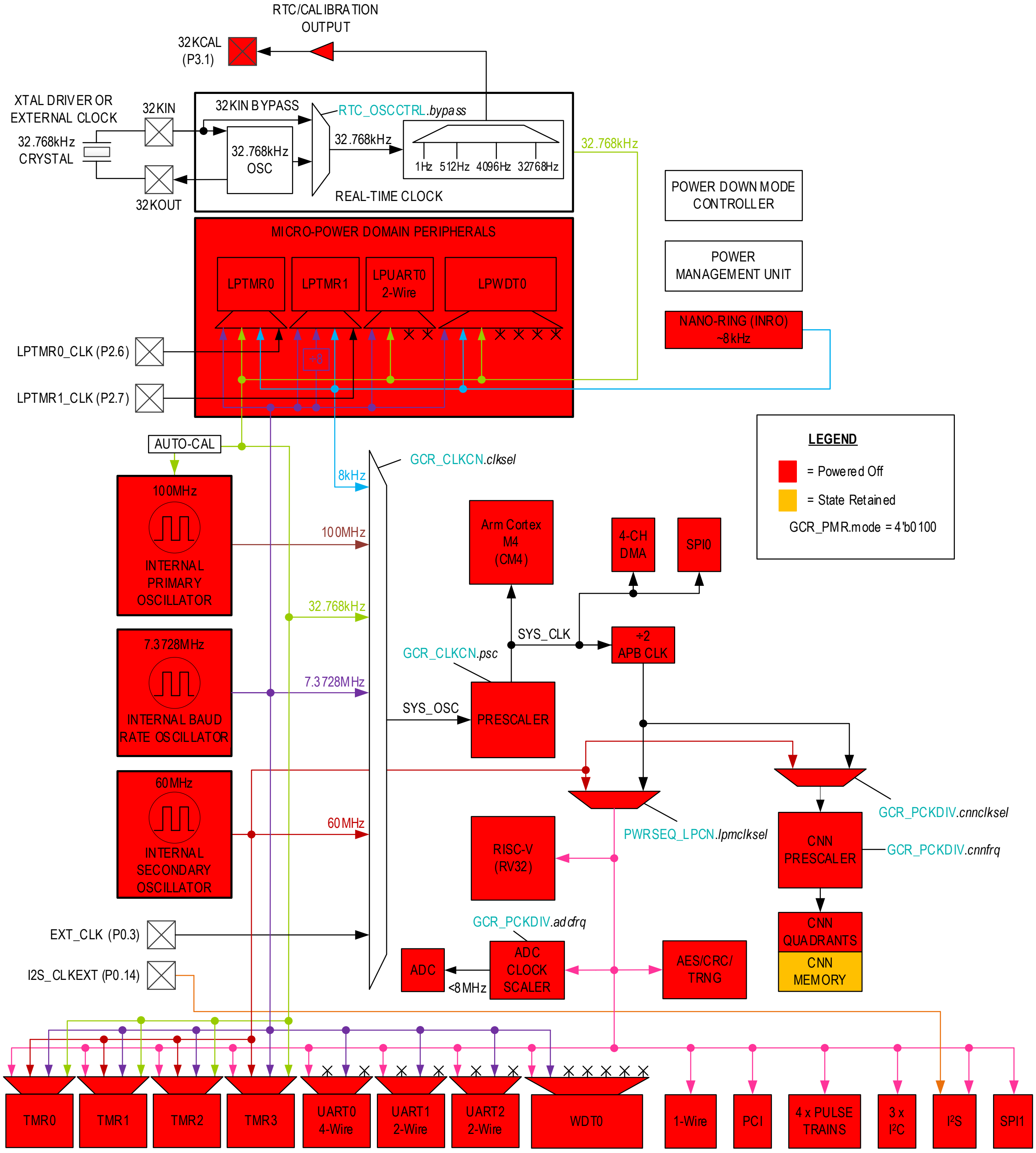
RAM Block #	Size	State Retention Control
sysram0	*32KB + ECC if enabled*	*PWRSEQ_LPCN.ramret0*
sysram1	*32KB*	*PWRSEQ_LPCN.ramret1*
sysram2	*48KB*	*PWRSEQ_LPCN.ramret2*
sysram3	*16KB*	*PWRSEQ_LPCN.ramret3*

4.3.2.5.1 Entering BACKUP

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

Either core can set *GCR_PM*.*mode* to *BACKUP,* and the device immediately enters *BACKUP*.

Figure 4-6: BACKUP Mode Clock and State Retention Block Diagram



4.3.2.6 PDM

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

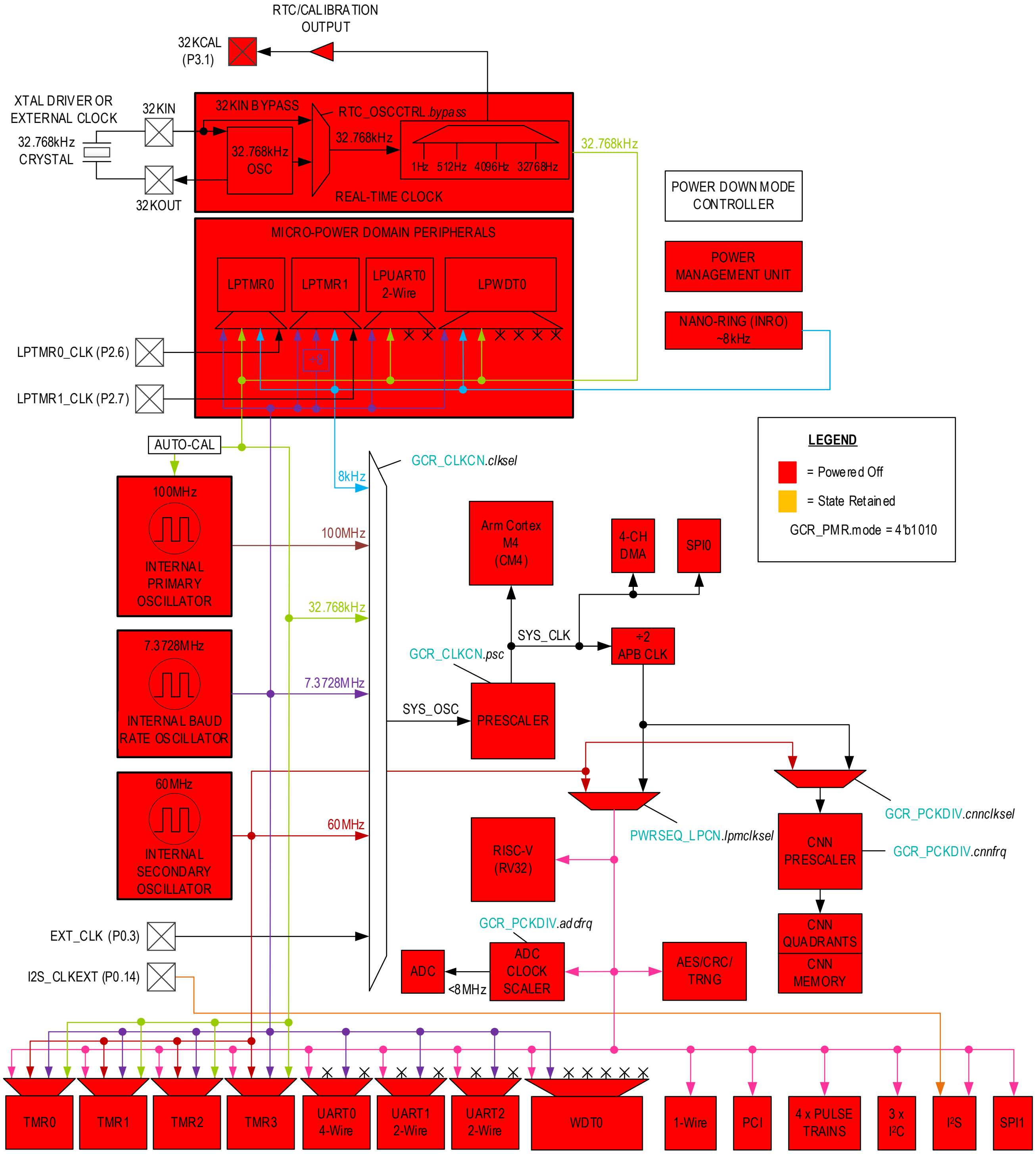
- The CM4 and RV32 are powered off
- All peripherals and all RAMs are powered down
- All oscillators are powered down
- There is no data retention in this mode, but values in the flash are preserved
- V_{REGI} POR voltage monitor is operational.
- Exit from PDM is possible through an external reset (RSTN)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

4.3.2.6.1 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 4-7: PDM Clock and State Retention Block Diagram



4.4 Wake-Up Sources for Each Operating Mode

In all operating modes other than *ACTIVE*, wake-up sources are required to re-enter *ACTIVE* operation. *Table 4-4* shows available wake-up sources for each operating mode of the MAX78002.

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

Table 4-4: Wake-Up Sources for Each Operating Mode in the MAX78002

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	LPCMP3	LPCMP2	LPCMP1	LPCMP0	RTC	WUT	GPIO3	GPIO2	GPIO1	GPIO0
SLEEP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
LPM		✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
UPM		✓										✓				✓	✓					✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
STANDBY		✓																									✓	✓	✓	✓	✓	✓	✓
BACKUP		✓																									✓	✓	✓	✓	✓	✓	✓
PDM		✓																												✓			

1: The CNN and CNN FIFO cannot wake the CM4 from LPM.

4.5 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*. Program execution begins at the reset vector address.

The contents of the always-on domain (AoD) are reset only on power-cycling V_{COREA}, V_{COREB}, V_{DDA}, V_{DDIOH}, or V_{REGI}.

The on-chip peripherals can also be reset to their POR default state using the two reset registers, *GCR_RST0* and *GCR_RST1*.

Table 4-5 shows the effects of each reset type on each of the operating modes.

Table 4-5: Reset and Low-Power Mode Effects

	Peripheral Reset[4]	Soft Reset[4]	System Reset[4]	POR	ACTIVE	SLEEP	LPM	UPM	BACKUP[3]	PDM
IPO	-	-	Off	Off	R	-	FW	Off	Off	Off
ISO	-	-	On	Off	R	-	FW	Off	-	-
ERTCO	-	-	-	Off	FW	FW	FW	FW	FW	FW
IBRO	-	-	Off	Off	R	-	FW	FW	Off	Off
ERFO	-	-	Off	Off	R	-	Off	Off	Off	Off
INRO	On	On	On	On	On	On	On	On	On	On
SYS_CLK	On	On	On[2]	On[2]	On	On	Off	Off	Off	Off
CPU Clock	On	On	On	On	On	Off	Off	Off	Off	Off
RTC				Reset	FW	FW	FW	FW	FW	FW
WDT0, WDT1	-	Reset	Reset	Reset	FW	Off	Off	Off	Off	Off
GPIO0 - GPIO3	-	Reset	Reset	Reset	R	-	-	-	-	-
All Other Peripherals	Reset	Reset	Reset	Reset	R	-	R	R	Off	Off
Always-On Domain	-	-	-	Reset	-	-	-	-	-	-
RAM Retention	-	-	-	Reset	-	-	On	On	FW	Off

Table key:

FW = Controlled by firmware
On = Enabled by hardware (Cannot be disabled)
Off = Disabled by hardware (Cannot be enabled)
- = No Effect
R = Restored to previous *ACTIVE* setting when exiting *LPM* and *UPM*, restored to system reset state when exiting *BACKUP* or *STORAGE*.

[1] The always-on domain (AoD) is only reset on power-cycling V_{COREA}, V_{COREB}, V_{DDA}, V_{DDIOH}, or V_{REGI}

[2] On a system reset or POR, the ISO is automatically set as the SYS_OSC.

[3] A system reset occurs when returning from *BACKUP* or *PDM*.

[4] Peripheral, soft, and system resets are initiated by software though the *GCR_RST0* register. System reset can also be triggered by the RSTN device pin or a watchdog reset.

4.5.1 Peripheral Reset

Peripheral reset 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* to 1. The reset is completed immediately upon setting *GCR_RST0.periph* to 1.

4.5.2 Soft Reset

A soft reset is the same as a peripheral reset except that it also resets the GPIO to its POR state.

To perform a soft reset, set *GCR_RST0.soft* to 1. The reset occurs immediately upon setting *GCR_RST0.soft* to 1.

4.5.3 System Reset

A system reset is the same as a soft reset, except it also resets all GCR, resetting the clocks to their POR 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, set *GCR_RST0.sys* to 1.

4.5.4 *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 *MAX78002 data sheet* for details of the reset voltage levels.

4.6 Unified Internal Cache Controllers

The MAX78002 includes two unified internal cache controllers. ICC0 is the cache controller used for the CM4. ICC1, if enabled, 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.

4.6.1 *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: Before enabling ICC1 as a cache controller, sysram3 should be zeroized.

Perform the following steps to enable each ICC:

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.

4.6.2 *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.

4.6.3 *Invalidating the ICC Cache and 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.

4.6.4 *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.iflushen* to 1 to immediately flush the contents of the 16KB cache and tag RAM.

4.6.5 *ICC Registers*

See *Table 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 4-6: Instruction Cache Controller Register Summary

Offset	Register	Name
[0x0000]	*ICCn_INFO*	*Cache ID Register*
[0x0004]	*ICCn_SZ*	*Cache Memory Size Register*
[0x0100]	*ICCn_CTRL*	*Instruction Cache Control Register*
[0x0700]	*ICCn_INVALIDATE*	*Instruction Cache Controller Invalidate Register*

4.6.6 ICC Register Details

Table 4-7: ICC0 Cache Information Register

ICC0 Cache Information			ICCn_INFO	[0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**
31:16	-	RO	0	**Reserved**
15:10	id	R	-	**Cache ID** This field returns the ID for the cache instance.
9:6	partnum	R	-	**Cache Part Number** This field returns the part number indicator for the cache instance.
5:0	relnum	R	-	**Cache Release Number** This field returns the release number for the cache instance.

Table 4-8: ICC0 Memory Size Register

ICC0 Memory Size			ICCn_SZ	[0x0004]
Bits	**Field**	**Access**	**Reset**	**Description**
31:16	mem	R	-	**Addressable Memory Size** This field indicates the size of addressable memory by the cache controller instance in 128KB units.
15:0	cch	R	-	**Cache Size** This field returns the size of the cache RAM in 1KB units. 16: 16KB Cache RAM

Table 4-9: ICC0 Cache Control Register

ICC0 Cache Control			ICCn_CTRL	[0x0100]
Bits	**Field**	**Access**	**Reset**	**Description**
31:17	-	R/W	-	**Reserved**
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. *Note: While this field reads 0, the cache is bypassed, and reads come directly from the line fill buffer.*
15:1	-	R/W	-	**Reserved**

ICC0 Cache Control				ICCn_CTRL [0x0100]
Bits	**Field**	**Access**	**Reset**	**Description**
0	en	R/W	0	**Cache Enable** Set this field to 1 to enable the cache. Setting this field to 0 invalidates the cache contents, and the line fill buffer handles all reads. 0: Disable 1: Enable

Table 4-10: ICC0 Invalidate Register

ICC0 Invalidate				ICCn_INVALIDATE [0x0700]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	invalid	W	-	**Invalidate** Writing any value to this register invalidates the cache.

4.7 RAM Memory Management

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

4.7.1 On-Chip Cache Management

The MAX78002 includes two unified internal cache controllers for code and data fetches from the flash memory. The caches can be enabled, disabled, zeroized, and flushed. See section *Unified Internal Cache Controller* for details.

4.7.2 RAM Zeroization

The GCR memory zeroize register, *GCR_MEMZ*, allows clearing memory for software 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*

4.8 Miscellaneous Control Registers (MCR)

See *Table 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 4-11: Miscellaneous Control Register Summary

Offset	Register Name	Description
[0x0000]	*MCR_ECCEN*	Error Correction Coding Enable Register
[0x0004]	*MCR_IPO_MTRIM*	IPO Manual Trim Register
[0x0008]	*MCR_OUTEN*	Miscellaneous Output Enable Register
[0x000C]	*MCR_CMP_CTRL*	Comparator Control Register
[0x0010]	*MCR_CTRL*	Miscellaneous Control Register
[0x0020]	*MCR_GPIO3_CTRL*	GPIO3 Pin Control Register
[0x0040]	*MCR_CWD0*	Code Word 0 Register
[0x0044]	*MCR_CWD1*	Code Word 1 Register
[0x0050]	*MCR_ADCCFG0*	ADC Configuration 0 Register
[0x0054]	*MCR_ADCCFG1*	ADC Configuration 1 Register
[0x0058]	*MCR_ADCCFG2*	ADC Configuration 2 Register
[0x0060]	*MCR_LDOCTRL*	LDO Control Register

4.8.1 Miscellaneous Control Register Details

Table 4-12: Error Correction Coding Enable Register

Error Correction Coding Enable			**MCR_ECCEN**	**[0x0000]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
0	ram0	R/W	0	**System RAM 0 ECC Enable** Set this field to 1 to enable ECC for *sysram0*. 0: Disabled 1: Enabled

Table 4-13: 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 the *MCR_IPO_MTRIM.mtrim* field must be greater than the trim setting in the *TRIMSIR_IPOLO.ipo_limitlo* field. If this bit is set to 0, the value loaded into the *MCR_IPO_MTRIM.mtrim* field must be less than the trim setting in the *TRIMSIR_CTRL.ipo_limithi* field. 0: *MCR_IPO_MTRIM.mtrim* < *TRIMSIR_IPOLO.ipo_limitlo* 1: *MCR_IPO_MTRIM.mtrim* > *TRIMSIR_CTRL.ipo_limithi*
7:0	mtrim	R/W	4	**Manual Trim Value** Set this value to the desired manual trim based on the value set in *MCR_IPO_MTRIM.trim_range*. If *MCR_IPO_MTRIM.trim_range* is 0, the value in this field must be less than the value in *TRIMSIR_IPOLO.ipo_limitlo*. If *MCR_IPO_MTRIM.trim_range* is 1, the value in this field must be greater than the value in *TRIMSIR_CTRL.ipo_limithi*.

Table 4-14: Output Enable Register

Output Enable		**MCR_OUTEN**		**[0x0008]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:2	-	RO	0	**Reserved**
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). PDOWN is active in *BACKUP* and *STANDBY*. 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 4-15: Comparator 0 Control Register

Comparator 0 Control		**MCR_CMP_CTRL**		**[0x000C]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	RO	0	**Reserved**
15	int_fl	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 *MCR_CMP_CTRL.pol* 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**
6	int_en	R/W	0	**Comparator 0 Interrupt Enable** Set this field to 1 to enable the interrupt 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**
0	en	R/W	0	**Comparator 0 Enable** Set this field to 1 to enable the comparator 0: Comparator disabled 1: Comparator enable

Table 4-16: Miscellaneous Control Register

Miscellaneous Control			MCR_CTRL	[0x0010]
Bits	**Name**	**Access**	**Reset**	**Description**
31:10	-	RO	0	**Reserved**
9	simo_rstd	R/W	0	**SIMO System Reset Disable** If this field is set, the SIMO is only reset by a POR. When this bit is set, the VSET* stays 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.
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
7:5	-	DNM	0	**Reserved**
4	ibro_en	R/W	1	**IBRO Enable for *UPM*** Set this field to 1 to enable IBRO during *UPM*. 0: Disabled 1: Enabled
3	ertco_en	R/W	0	**ERTCO Enable for *UPM, STANDBY, and BACKUP*** Set this field to 1 to enable the ERTCO in *UPM, STANDBY* and *BACKUP*. 0: Disabled 1: Enabled
2	inro_en	R/W	0	**INRO Enable for RTC** Set this field to 1 to enable the INRO as the clock source for the RTC. 0: Disabled 1: Enabled
1:0	cmphyst	RO	0	**Comparator Hysteresis**

4.8.1.1 GPIO 3 Control

Table 4-17: GPIO3 Pin Control Register

GPIO3 Pin Control			MCR_GPIO3_CTRL	[0x0020]
Bits	**Name**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
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 pullup resistor for P3.1 0: Pull-up Disabled 1: Pull-up Enabled

GPIO3 Pin Control			MCR_GPIO3_CTRL	[0x0020]
Bits	**Name**	**Access**	**Reset**	**Description**
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.
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 pullup 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.

Table 4-18: Code Word 0 Register

Code Word 0			MCR_CWD0	[0x0040]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	data	R/W	0	**Data** This register maintains the contents written to it as long as V_{REGI} supply is valid.

Table 4-19: Code Word 1 Register

Code Word 1			MCR_CWD1	[0x0044]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	data	R/W	0	**Data** This register maintains the contents written to it as long as V_{REGI} supply is valid.

Table 4-20: ADC Configuration Register 0

ADC Configuration 0			MCR_ADCCFG0	[0x0038]
Bits	**Field**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved**

ADC Configuration 0			MCR_ADCCFG0	[0x0038]
Bits	**Field**	**Access**	**Reset**	**Description**
3	ref_sel	R/W	0	**ADC Reference Select** This field selects either the 1.25V or 2.048V internal reference when the internal reference is selected (*MCR_ADCCFG0.ext_ref* = 0). 0: 1.25V 1: 2.048V
2	ext_ref	R/W	0	**ADC External Reference Select** This field selects between the internal and external references. 0: Internal reference 1: External reference
1:0	-	RO	0	**Reserved**

Table 4-21: ADC Configuration Register 1

ADC Configuration 1			MCR_ADCCFG1	[0x003C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	RO	0	**Reserved**

Table 4-22: ADC Configuration Register 2

ADC Configuration 2			MCR_ADCCFG2	[0x0040]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	RO	0	**Reserved**

Table 4-23: LDO Control Register

LDO Control			MCR_LDOCTRL	[0x0060]
Bits	**Name**	**Access**	**Reset**	**Description**
31:2	-	RO	0	**Reserved**
1	2p5en	R/W	0	**LDO 2.5V Enable** Set this field to 1 to enable the 2.5V LDO for the MIPI CSI-2 interface. 0: Disabled. 1: Enabled.
0	0p9en	R/W	0	**LDO 0.9V Enable**

4.9 Single Inductor Multiple Output Power Supply (SIMO)

The SIMO switch mode power supply allows the device to operate autonomously from a single lithium cell. The SIMO provides three buck switching regulators (V_{REGO_A} thru V_{REGO_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 shown in *Table 4-24*.

4.9.1 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 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}) AoD 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 device data sheet electrical characteristics for the trigger threshold values and power fail reset voltages.

Table 4-24: 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
-	-	V_{DDIO} Power On	GPIO pad held in reset until the voltage rises above its threshold
-	-	V_{DDIOH} Power On	GPIO pad held in reset until the voltage rises above its threshold
-	-	V_{DDIO}	GPIO pad logic enters POR
-	-	V_{DDIOH}	GPIO pad logic enters POR

4.9.2 Single Inductor Multiple Output Registers (SIMO)

See *Table 3-3* for the SIMO Controller Peripheral Base Address.

Table 4-25: 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.*

Offset	Register	Access	Name
[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.*

4.9.3 Single Inductor Multiple Output (SIMO) Registers Details

Table 4-26: SIMO Buck Voltage Regulator A Control Register

SIMO Buck Voltage Regulator A Control		**SIMO_VREGO_A**		**[0x0004]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	-	**Reserved**
7	rangea	R/W	1	**Regulator Output A Range** This field selects the regulator output range for V_{REGO_A}. 0: 0.5V to 1.77V 1: 0.6V to 1.87V
6:0	vseta	R/W	0x78	**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. *SIMO_VREGO_A.rangea* = 1: $\text{Output Voltage} = 0.6\text{V} + (10\text{mV} \times \text{vseta})$ *SIMO_VREGO_A.rangea* = 0: $\text{Output Voltage} = 0.5\text{V} + (10\text{mV} \times \text{vseta})$ Default: 0x78 = *SIMO_VREGO_A.rangea* = 0, *Output Voltage* = 1.7V; *SIMO_VREGO_A.rangea* = 1, *Output Voltage* = 1.8V *Warning: When this regulator is connected as shown in Table 4-24: SIMO Power Supply Device Pin Connectivity, the following apply:* 1. The maximum setting for this regulator must be followed for V_{DDA} as indicated in the device data sheet. 2. Setting the regulator to a voltage below the power-fail reset voltage for V_{DDA} initiates the power monitor reset action.

Table 4-27: SIMO Buck Voltage Regulator B Control Register

SIMO Buck Voltage Regulator B Control		**SIMO_VREGO_B**		**[0x0008]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	-	**Reserved**
7	rangeb	R/W	1	**Regulator Output B Range** This field selects the regulator output range for V_{REGO_B}. 0: 0.5V to 1.77V 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	0x32	**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. *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 *SIMO_VREGO_B.rangeb* selected (1.77V or 1.87V) Default: 0x32 = *SIMO_VREGO_B.rangeb* = 0, *Output Voltage* = 1.0V; *SIMO_VREGO_B.rangeb* = 1, *Output Voltage* = 1.1V ***Warning**: When this regulator is connected as shown in Table 4-24: SIMO Power Supply Device Pin Connectivity, the following apply:* 1. The maximum setting for this regulator must be followed for V_{COREB} as indicated in the device data sheet. 2. Setting the regulator to a voltage below the power-fail reset voltage for V_{COREB} initiates the power monitor reset action.

Table 4-28: SIMO Buck Voltage Regulator C Control Register

SIMO Buck Voltage Regulator C Control		SIMO_VREGO_C		[0x000C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	-	**Reserved**
7	rangec	R/W	1	**Regulator Output Range** This field elects the regulator output range for V_{REGO_C}. 0: 0.5V to 1.77V 1: 0.6V to 1.87V
6:0	vsetc	R/W	0x32	**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 *SIMO_VREGO_C.rangec* selected (1.77V or 1.87V) Default: 0x32 = *SIMO_VREGO_C.rangec* = 0, *Output Voltage* = 1.0V; *SIMO_VREGO_C.rangec* = 1, *Output Voltage* = 1.1V ***Warning**: When this regulator is connected as shown in Table 4-24: SIMO Power Supply Device Pin Connectivity, the following apply:* 1. The maximum setting for this regulator must be followed for V_{COREA} as indicated in the device data sheet. 2. Setting the regulator to a voltage below the power-fail reset voltage for V_{COREA} initiates the power monitor reset action.

Table 4-29: SIMO High Side FET Peak Current V_{REGO_A} V_{REGO_B} Register

SIMO High Side FET Peak Current V_{REGO_A} V_{REGO_B}		SIMO_IPKA		[0x0014]
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	-	**Reserved**
7:4	ipksetb	RO	8	**Reserved**

SIMO High Side FET Peak Current V_{REGO_A} V_{REGO_B}			SIMO_IPKA	[0x0014]
Bits	**Field**	**Access**	**Reset**	**Description**
3:0	ipkseta	RO	8	**Reserved**

Table 4-30: SIMO High Side FET Peak Current V_{REGO_C} Register

SIMO High Side FET Peak Current V_{REGO_C} V_{REGO_D}			SIMO_IPKB	[0x0018]
Bits	**Field**	**Access**	**Reset**	**Description**
31:4	-	RO	-	**Reserved**
3:0	ipksetc	RO	8	**Reserved**

Table 4-31: SIMO Maximum High Side FET Time On Register

SIMO Maximum High Side FET On Time			SIMO_MAXTON	[0x001C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved**
3:0	tonset	RO	0x8	**Reserved**

Table 4-32: SIMO Buck Cycle Count V_{REGO_A} Register

SIMO Buck Cycle Count VREGO_A			SIMO_ILOAD_A	[0x0020]
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
7:0	iloada	RO	0	**Reserved**

Table 4-33: SIMO Buck Cycle Count V_{REGO_B} Register

SIMO Buck Cycle Count VREGO_B			SIMO_ILOAD_B	[0x0024]
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
7:0	iloadb	RO	0	**Reserved**

Table 4-34: SIMO Buck Cycle Count V_{REGO_C} Register

SIMO Buck Cycle Count VREGO_C			SIMO_ILOAD_C	[0x0028]
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
7:0	iloadc	RO	0	**Reserved**

Table 4-35: SIMO Buck Cycle Count Alert V_{REGO_A} Register

SIMO Buck Cycle Count Alert VREGO_A			SIMO_BUCK_ALERT_THR_A	[0x0030]
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
7:0	buckthra	RO	0	**Reserved**

Table 4-36: SIMO Buck Cycle Count Alert V_{REGO_B} Register

SIMO Buck Cycle Count Alert VREGO_A			**SIMO_BUCK_ALERT_THR_B**	**[0x0034]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
7:0	buckthrb	RO	0	**Reserved**

Table 4-37: SIMO Buck Cycle Count Alert V_{REGO_C} Register

SIMO Buck Cycle Count Alert VREGO_A			**SIMO_BUCK_ALERT_THR_C**	**[0x0038]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
7:0	buckthrc	RO	0	**Reserved**

Table 4-38: SIMO Buck Regulator Output Ready Register

SIMO Buck Regulator Output Ready			**SIMO_BUCK_OUT_READY**	**[0x0040]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved**
3	buckoutrdya	RO	0	**V_{REGO_A} Output Ready** When *SIMO_VREGO_A.vseta* changes, this bit is set when the output voltage has reached its regulated value. It is not cleared if the output voltage drops below its set value. 0: Not ready 1: Ready
2	buckoutrdyb	RO	0	**V_{REGO_B} Output Ready** When *SIMO_VREGO_B.vsetb* changes, this bit is set when the output voltage has reached its regulated value. It is not cleared if the output voltage drops below its set value. 0: Not ready 1: Ready
1	buckoutrdyc	R/W	0	**V_{REGO_C} Output Ready** When *SIMO_VREGO_C.vsetc* changes, this bit is set when the output voltage has reached its regulated value. It is not cleared if the output voltage drops below its set value. 0: Not ready 1: Ready
0	-	RO	0	**Reserved**

Table 4-39: SIMO Zero Cross Calibration V_{REGO_A} Register

SIMO Zero Cross Calibration V_{REGO_A}			**SIMO_ZERO_CROSS_CAL_A**	**[0x0044]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:5	-	RO	0	**Reserved**
4:0	zxcala	RO	0	**Reserved**

Table 4-40: SIMO Zero Cross Calibration V_{REGO_B} Register

SIMO Zero Cross Calibration V_{REGO_B}		SIMO_ZERO_CROSS_CAL_B		[0x0048]
Bits	**Field**	**Access**	**Reset**	**Description**
31:5	-	RO	0	**Reserved**
4:0	zxcalb	RO	0	**Reserved**

Table 4-41: SIMO Zero Cross Calibration V_{REGO_C} Register

SIMO Zero Cross Calibration V_{REGO_C}		SIMO_ZERO_CROSS_CAL_C		[0x004C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:5	-	RO	0	**Reserved**
4:0	zxcalc	RO	0	**Reserved**
4:0	zxcald	RO	0	**Reserved**

4.10 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 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 4-42: Low-Power Control Register Summary

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

4.10.1 Low-Power General Control Registers Details

Table 4-43: Reset Control Register

Low-Power Reset Control		LPGCR_RST		[0x0004]
Bits	**Field**	**Access**	**Reset**	**Description**
31:7	-	RO	0	**Reserved**
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**
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.

Low-Power Reset Control			**LPGCR_RST**	**[0x0004]**
Bits	**Field**	**Access**	**Reset**	**Description**
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 4-44: Clock Disable Register

Clock Disable			**LPGCR_PCLKDIS**	**[0x0008]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:7	-	RO	0	**Reserved**
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. *Note: This field disables clocks to LPCOMP1, LPCOMP2, and LPCOMP3.* 0: Enabled 1: Disabled
5	-	RO	0	**Reserved**
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

Clock Disable		LPGCR_PCLKDIS		[0x0008]
Bits	**Field**	**Access**	**Reset**	**Description**
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

4.11 Power Sequencer Registers (PWRSEQ)

See *Table 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 4-45: Power Sequencer Register Summary

Offset	Register	Name
[0x0000]	*PWRSEQ_LPCN*	*Low Power Control Register*
[0x0004]	*PWRSEQ_LPWKST0*	*Low Power GPIO0 Wakeup Status Flags*
[0x0008]	*PWRSEQ_LPWKEN0*	*Low Power GPIO0 Wakeup Enable Register*
[0x000C]	*PWRSEQ_LPWKST1*	*Low Power GPIO1 Wakeup Status Flags*
[0x0010]	*PWRSEQ_LPWKEN1*	*Low Power GPIO1 Wakeup Enable Register*
[0x0014]	*PWRSEQ_LPWKST2*	*Low Power GPIO2 Wakeup Status Flags*
[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_GP0*	*General Purpose Register 0*
[0x004C]	*PWRSEQ_GP1*	*General Purpose Register 1*

4.11.1 Power Sequencer Register Details

Table 4-46: Low Power Control Register

Low Power Control		PWRSEQ_LPCN		[0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**
31	wkrst	R/W1O	0	**Low Power Wakeup Status Register Clear** Write 1 to this field to clear the Low Power Wakeup Status registers: • *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**

Low Power Control			PWRSEQ_LPCN	[0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**
11	bgoff	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	-	RO	0	**Reserved**
9	fast_entry_dis	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	isoclk_select	R/W	1	**Low Power Mode APB Clock Select** This field selects the clock source 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	ramret7	R/W	0	**System RAM 7 Data Retention Enable for *BACKUP*** Set this field to 1 to enable data retention for *sysram7*. See *SRAM Space* for the system RAM configuration. 0: Disable data retention for *sysram7* address space in *BACKUP*. 1: Enable data retention for *sysram7* address space in *BACKUP*.
6	ramret6	R/W	0	**System RAM 6 Data Retention Enable for *BACKUP*** Set this field to 1 to enable data retention for *sysram6*. See *SRAM Space* for the system RAM configuration. 0: Disable data retention for *sysram6* address space in *BACKUP*. 1: Enable data retention for *sysram6* address space in *BACKUP*.
5	ramret5	R/W	0	**System RAM 5 Data Retention Enable for *BACKUP*** Set this field to 1 to enable data retention for *sysram5*. See *SRAM Space* for the system RAM configuration. 0: Disable data retention for *sysram5* address space in *BACKUP*. 1: Enable data retention for *sysram5* address space in *BACKUP*.
4	ramret4	R/W	0	**System RAM 4 Data Retention Enable for *BACKUP*** Set this field to 1 to enable data retention for *sysram4*. See *SRAM Space* for the system RAM configuration. 0: Disable data retention for *sysram4* address space in *BACKUP*. 1: Enable data retention for *sysram4* address space in *BACKUP*.
3	ramret3	R/W	0	**System RAM 3 Data Retention Enable for *BACKUP*** Set this field to 1 to enable data retention for *sysram3*. See *SRAM Space* for the system RAM configuration. 0: Disable data retention for *sysram3* address space in *BACKUP*. 1: Enable data retention for *sysram3* address space in *BACKUP*.
2	ramret2	R/W	0	**System RAM 2 Data Retention Enable for *BACKUP*** Set this field to 1 to enable data retention for *sysram2*. See *SRAM Space* for the system RAM configuration. 0: Disable data retention for *sysram2* address space in *BACKUP*. 1: Enable data retention for *sysram2* address space in *BACKUP*.

Low Power Control			PWRSEQ_LPCN	[0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**
1	ramret1	R/W	0	**System RAM 1 Data Retention Enable for *BACKUP*** Set this field to 1 to enable data retention for *sysram1*. See *SRAM Space* for the system RAM configuration. 0: Disable data retention for *sysram1* address space in *BACKUP*. 1: Enable data retention for *sysram1* address space in *BACKUP*.
0	ramret0	R/W	0	**System RAM 0 Data Retention Enable for *BACKUP*** Set this field to 1 to enable data retention for *sysram0*. See *SRAM Space* for the system RAM configuration. 0: Disable data retention for *sysram0* address space in *BACKUP*. 1: Enable data retention for *sysram0* address space in *BACKUP*.

Table 4-47: GPIO0 Low Power Wakeup Status Flags

GPIO0 Low Power Wakeup Status Flags			PWRSEQ_LPWKST0	[0x0004]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	wakest	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 transitions from a low-power mode to *ACTIVE* if the corresponding GPIO pin's interrupt enable bit is set in the *PWRSEQ_LPWKEN0* register. *Note: Clear this register before entering any low-power mode.*

Table 4-48: GPIO0 Low Power Wakeup Enable Registers

GPIO0 Low Power Wakeup Enable			PWRSEQ_LPWKEN0	[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 causes an interrupt to be generated to wake up the device from any low-power mode to *ACTIVE*. A wake-up event sets the corresponding GPIO0 bit in the *PWRSEQ_LPWKST0* register, enabling the determination of which GPIO0 pin triggered the wake-up event. Bits corresponding to unimplemented GPIO are ignored. *Note: To enable the MAX78002 to wake up from a low-power mode on a GPIO pin transition, first set the GPIO wake-up enable register bit GCR_PM.gpio_we to 1.*

Table 4-49: GPIO1 Low Power Wakeup Status Flags

GPIO1 Low Power Wakeup Status Flags			PWRSEQ_LPWKST1	[0x000C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:10	-	RO	0	**Reserved** Bits corresponding to unimplemented GPIO are ignored.

GPIO1 Low Power Wakeup Status Flags			PWRSEQ_LPWKST1	[0x000C]
Bits	**Field**	**Access**	**Reset**	**Description**
9:0	wakest	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 wakes from a low-power mode to *ACTIVE* if the corresponding interrupt enable bit is set in *PWRSEQ_LPWKEN1*. *Note: Clear this register before entering any low-power mode.*

Table 4-50: 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 causes an interrupt to be generated that wakes up the device from any low-power mode to *ACTIVE*. A wake-up event sets the corresponding GPIO1 bit in the *PWRSEQ_LPWKST1* register, enabling the determination of which GPIO1 pin triggered the wake-up event. Bits corresponding to unimplemented GPIO are ignored. *Note: To enable the MAX78002 to wake up from a low-power mode on a GPIO pin transition, first set the GPIO wake-up enable register bit GCR_PM.gpio_we to 1.*

Table 4-51: 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	wakest	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 pin's corresponding bit in this register is set. The device wakes from a low-power mode to *ACTIVE* if the corresponding interrupt enable bit is set in *PWRSEQ_LPWKEN2*. *Note: Clear this register before entering any low-power mode.*

Table 4-52: 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.

GPIO2 Low Power Wakeup Enable		PWRSEQ_LPWKEN2		[0x0018]
Bits	**Field**	**Access**	**Reset**	**Description**
7:0	en	R/W	0	**GPIO2 Pin Wakeup Interrupt Enable** Setting a GPIO2 pin's bit in this register causes an interrupt to be generated that wakes up the device from any low-power mode to *ACTIVE*. A wake-up event sets the corresponding GPIO2 bit in the *PWRSEQ_LPWKST2* register, enabling the determination of which GPIO2 pin triggered the wake-up event. *Note: To enable the MAX78002 to wake up from a low-power mode on a GPIO pin transition, first set the GPIO wake-up enable register bit GCR_PM.gpio_we to 1.*

Table 4-53: 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**
1:0	wakest	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. The device wakes from a low-power mode to *ACTIVE* if the corresponding interrupt enable bit is set in *PWRSEQ_LPWKEN3*. *Note: Clear this register before entering any low-power mode.*

Table 4-54: GPIO3 Low Power Wakeup Enable Registers

GPIO3 Low Power Wakeup Enable		PWRSEQ_LPWKEN3		[0x0020]
Bits	**Field**	**Access**	**Reset**	**Description**
31:2		RO	0	**Reserved**
1:0	en	R/W	0	**GPIO3 Pin Wakeup Interrupt Enable** Setting a GPIO3 pin's bit in this register causes an interrupt to be generated that wakes up the device from any low-power mode to *ACTIVE*. A wake-up event sets the corresponding GPIO3 bit in the *PWRSEQ_LPWKST3* register, enabling the determination of which GPIO3 pin triggered the wake-up event. Bits corresponding to unimplemented GPIO are ignored. *Note: To enable the MAX78002 to wake up from a low-power mode on a GPIO pin transition, first set the GPIO wake-up enable register bit GCR_PM.gpio_we = 1.*

Table 4-55: 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**
17	reset	R/W1C	0	**Reset Detected Wakeup Flag** This field is set when an external reset caused the wake-up event.
16	backup	R/W1C	0	**BACKUP Mode Wakeup Flag** This field is set when the device wakes up from *BACKUP*.

Low Power Peripheral Wakeup Status Flags		PWRSEQ_LPPWST		[0x0030]
Bits	**Field**	**Access**	**Reset**	**Description**
15:5	-	RO	0	**Reserved**
4	aincomp0	R/W1C	0	**Comparator 0 Wakeup Flag** This field is set if the wake-up event was the result of a comparator 0 trigger event.
3:0	-	RO	0	**Reserved**

Table 4-56: Low Power Peripheral Wakeup Enable Registers

Low Power Peripheral Wakeup Enable		PWRSEQ_LPPWEN		[0x0034]
Bits	**Field**	**Access**	**Reset**	**Description**
31:27		RO	0	**Reserved**
26	lpcmp	R/W	0	**Low Power Comparator Interrupt Wakeup Enable** Set this field to 1 to enable wake-up events from the LPCMP_IRQn interrupt. 0: Disable wake-up on interrupt. 1: Enable wake-up on interrupt.
25	spi1	R/W	0	**SPI1 Interrupt Wakeup Enable** Set this field to 1 to enable wake-up events from the SPI1 interrupt. 0: Disable wake-up on interrupt. 1: Enable wake-up on interrupt.
24	i2s	R/W	0	**I^{2}S Interrupt Wakeup Enable** Set this field to 1 to enable wake-up events from the I2S interrupt. 0: Disable wake-up on interrupt. 1: Enable wake-up on interrupt.
23	i2c2	R/W	0	**I2C2 Interrupt Wakeup Enable** Set this field to 1 to enable wake-up events from the I2C2 interrupt. 0: Disable wake-up on interrupt. 1: Enable wake-up on interrupt.
22	i2c1	R/W	0	**I2C1 Interrupt Wakeup Enable** Set this field to 1 to enable wake-up events from the I2C1 interrupt. 0: Disable wake-up on interrupt. 1: Enable wake-up on interrupt.
21	i2c0	R/W	0	**I2C0 Interrupt Wakeup Enable** Set this field to 1 to enable wake-up events from the I2C0 interrupt. 0: Disable wake-up on interrupt. 1: Enable wake-up on interrupt.
20	uart3	R/W	0	**LPUART0 (UART3) Interrupt Wakeup Enable** Set this field to 1 to enable wake-up events from LPUART0 (UART3) interrupt. 0: Disable wake-up on interrupt. 1: Enable wake-up on interrupt.
19	uart2	R/W	0	**UART2 Interrupt Wakeup Enable** Set this field to 1 to enable wake-up events from the UART2 interrupt. 0: Disable wake-up on interrupt. 1: Enable wake-up on interrupt.

Low Power Peripheral Wakeup Enable			PWRSEQ_LPPWEN	[0x0034]
Bits	**Field**	**Access**	**Reset**	**Description**
18	uart1	R/W	0	**UART1 Interrupt Wakeup Enable** Set this field to 1 to enable wake-up events from the UART1 interrupt. 0: Disable wake-up on interrupt. 1: Enable wake-up on interrupt.
17	uart0	R/W	0	**UART0 Interrupt Wakeup Enable** Set this field to 1 to enable wake-up events from the UART0 interrupt. 0: Disable wake-up on interrupt. 1: Enable wake-up on interrupt.
16	tmr5	R/W	0	**LPTMR1 (TMR5) Interrupt Wakeup Enable** Set this field to 1 to enable wake-up events from the LPTMR1 (TMR5) interrupt. 0: Disable wake-up on interrupt. 1: Enable wake-up on interrupt.
15	tmr4	R/W	0	**LPTMR0 (TMR4) Interrupt Wakeup Enable** Set this field to 1 to enable wake-up events from the LPTMR0 (TMR4) interrupt. 0: Disable wake-up on interrupt. 1: Enable wake-up on interrupt.
14	tmr3	R/W	0	**TMR3 Interrupt Wakeup Enable** Set this field to 1 to enable wake-up events from the TMR3 interrupt. 0: Disable wake-up on interrupt. 1: Enable wake-up on interrupt.
13	tmr2	R/W	0	**TMR2 Interrupt Wakeup Enable** Set this field to 1 to enable wake-up events from the TMR2 interrupt. 0: Disable wake-up on interrupt. 1: Enable wake-up on interrupt.
12	tmr1	R/W	0	**TMR1 Interrupt Wakeup Enable** Set this field to 1 to enable wake-up events from the TMR1 interrupt. 0: Disable wake-up on interrupt. 1: Enable wake-up on interrupt.
11	tmr0	R/W	0	**TMR0 Interrupt Wakeup Enable** Set this field to 1 to enable wake-up events from the TMR0 interrupt. 0: Disable wake-up on interrupt. 1: Enable wake-up on interrupt.
10	cpu1	R/W	0	**CPU1 (RV32) Interrupt Wakeup Enable** Set this field to 1 to enable wake-up events from the RV32 interrupt. 0: Disable wake-up on interrupt. 1: Enable wake-up on interrupt.
9	wdt1	R/W	0	**WDT1 (LPWDT0) Interrupt Wakeup Enable** Set this field to 1 to enable wake-up events from the WDT1 (LPWDT0) interrupt. 0: Disable wake-up on interrupt. 1: Enable wake-up on interrupt.
8	wdt0	R/W	0	**WDT0 Interrupt Wakeup Enable** Set this field to 1 to enable wake-up events from the WDT0 interrupt. 0: Disable wake-up on interrupt. 1: Enable wake-up on interrupt.

Low Power Peripheral Wakeup Enable			PWRSEQ_LPPWEN	[0x0034]
Bits	**Field**	**Access**	**Reset**	**Description**
7:5	-	RO	0	**Reserved**
4	lpcmp	R/W	0	**Comparator 0 Wakeup Enable** Set this field to 1 to enable wake-up events from Comparator 0. Comparator 0 can wake the device up from *SLEEP*, *LPM*, *UPM*, *STANDBY,* and *BACKUP*. 0: Disable wake-up on interrupt 1: Enable wake-up on interrupt
3:0	-	RO	0	**Reserved**

Table 4-57: Low Power General Purpose 0 Register

Low Power General Purpose 0			PWRSEQ_GP0	[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*.

Table 4-58: Low Power General Purpose 1 Register

Low Power General Purpose 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*.

4.12 Trim System Initialization Registers (TRIMSIR)

See *Table 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 POR. System reset, soft reset, and peripheral reset do not affect the TRIMSIR register values.

Table 4-59: Trim System Initialization Register Summary

Offset	Register Name	Description
[0x0008]	*TRIMSIR_RTC*	*RTC Trim System Initialization Register*
[0x0034]	*TRIMSIR_SIMO*	*System Initialization Register*
[0x003C]	*TRIMSIR_IPOLO*	*System initialization Function Status Register*
[0x0040]	*TRIMSIR_CTRL*	Control Trim System Initialization Register
[0x0044]	*TRIMSIR_INRO*	*INRO Trim System Initialization Register*

4.12.1 TRIM System Initialization Register Details

Table 4-60: RTC Trim System Initialization Register

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

Table 4-61: SIMO Trim System Initialization Register

SIMO System Initialization		TRIMSIR_SIMO		[0x0034]
Bits	**Name**	**Access**	**Reset**	**Description**
31:3	-	RO	0	**Reserved**
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. 3: $\frac{INRO}{32}$. 4: Reserved. 5: $\frac{INRO}{64}$. 6: Reserved. 7: $\frac{INRO}{128}$.

Table 4-62: IPO Low Trim System Initialization Register

IPO Trim Low System Initialization		TRIMSIR_IPOLO		[0x003C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
7:0	ipo_limitlo	RO	See Description	**IPO Low Trim Limit** This field contains the low trim limit for the IPO.

Table 4-63: 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**
25:24	inro_sel	R/W	2	**INRO Clock Select** This field selects the INRO frequency. 0: 8kHz. 1: 16kHz. 2: 30kHz. 3: Reserved.
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	**V_{DDA} High Trim Limit** This field is the high trim limit for V_{DDA}.
7	-	RO	0	**Reserved**
6:0	vdda_limitlo	R/W	0x64	**V_{DDA} Low Trim Limit** This field is the low trim limit for V_{DDA}.

Table 4-64: INRO Trim System Initialization Register

INRO System Initialization		**TRIMSIR_INRO**		**[0x0044]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
7:6	lpclksel	R/W	2	**INRO Low Power Mode Clock Select** This field selects the INRO clock frequency for *LPM* operation. 0: 8kHz. 1: 16kHz. 2: 30kHz (POR default). 3: Reserved.
5:3	trim30k	R/W	0	**INRO 30kHz Trim** This field contains the trim for the INRO when set to 30kHz.
2:0	trim16k	R/W	0	**INRO 16kHz Trim** This field contains the trim for the INRO when set to 16kHz.

4.13 Global Control Registers (GCR)

See *Table 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 GCR are only reset on a system reset or POR. A soft reset or peripheral reset does not affect these registers.

Table 4-65: 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*
[0x0010]	*GCR_IPLL_CTRL*	*ITO PLL Control*
[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_SYSST*	*System Status Flags*
[0x0044]	*GCR_RST1*	*Reset Register 1*
[0x0048]	*GCR_PCLKDIS1*	*Peripheral Clocks Disable 1*
[0x004C]	*GCR_EVENTEN*	*Event Enable Register*
[0x0050]	*GCR_REVISION*	*Revision Register*
[0x0054]	*GCR_SYSIE*	*System Status Interrupt Enable*
[0x0064]	*GCR_ECCERR*	*Error Correction Coding Error Register*
[0x0068]	*GCR_ECCCED*	*Error Correction Coding Correctable Error Detected*
[0x006C]	*GCR_ECCIE*	*Error Correction Coding Interrupt Enable Register*
[0x0070]	*GCR_ECCADDR*	*Error Correction Coding Error Address Register*
[0x0080]	*GCR_GPR0*	*General Purpose Register 0*

4.13.1 Global Control Register Details (GCR)

Table 4-66: System Control Register

System Control		**GCR_SYSCTRL**		**[0x0000]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:18	-	RO	0	**Reserved**
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.

System Control			GCR_SYSCTRL	[0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**
15	chkres	R	0	**ROM Checksum Calculation Pass/Fail** This field is the result after setting the *GCR_SYSCTRL.cchk* bit. This bit is only valid after the ROM checksum is complete and *GCR_SYSCTRL.cchk* is cleared. 0: Pass. 1: Fail.
14	swd_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. *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.romdone bit is 1.*
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.
12	romdone	DNM	1	**ROM Start Code Status** Reserved, Do Not Modify.
11:7	-	RO	0	**Reserved**
6	icc0_flush	R/W	0	**ICC0 Cache Flush** Write 1 to flush the code cache and the instruction buffer for the CM4. 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: Normal operation. 1: Flush the contents of the ICC0 cache.
5	-	RO	0	**Reserved**
4	flash_page_flip	R/*	0	**Flash Page Flip Flag** This field flips the bottom and top halves of flash memory. *Note: Software should not change the state of this bit during normal operation. Any change to this bit also flushes both code and data caches.* 0: Physical layout matches the logical layout. 1: Top and bottom halves flipped.
3:1	-	RO	1	**Reserved**
0	bstapen	DNM	*	**Boundary Scan Tap Enable** This field's reset value matches *GCR_SYSST.icelock*. Do not modify.

Table 4-67: 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. 0: Normal operation. 1: Initiate reset.
30	periph	R/W	0	**Peripheral Reset** Write 1 to reset. This field is cleared by hardware when the reset is complete. 0: Normal operation. 1: Initiate reset. *Note: Watchdog timers, GPIO ports, the AoD, RAM retention, and the GCR are unaffected. See Table 4-5 for additional information.*
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. 0: Normal operation. 1: Initiate reset.
28	uart2	R/W	0	**UART2 Reset** Write 1 to reset. This field is cleared by hardware when the reset is complete. 0: Normal operation. 1: Initiate reset.
27	-	R/W	0	**Reserved**
26	adc	R/W	0	**ADC Reset** Write 1 to reset. This field is cleared by hardware when the reset is complete. 0: Normal operation. 1: Initiate reset.
25	cnn	R/W	0	**CNN Reset** Write 1 to reset. This field is cleared by hardware when the reset is complete. 0: Normal operation. 1: Initiate reset.
24	trng	R/W	0	**TRNG Reset** Write 1 to reset. This field is cleared by hardware when the reset is complete. 0: Normal operation. 1: Initiate reset.
23	-	R/W	0	**Reserved**
22	smphr	R/W	0	**Semaphore Reset** Write 1 to reset. This field is cleared by hardware when the reset is complete. 0: Normal operation. 1: Initiate reset.
21:18	-	R/W	0	**Reserved**
17	rtc	R/W	0	**RTC Reset** Write 1 to reset. This field is cleared by hardware when the reset is complete. 0: Normal operation. 1: Initiate reset.

Reset 0				**GCR_RST0** **[0x0004]**
Bits	**Field**	**Access**	**Reset**	**Description**
16	i2c0	R/W	0	**I2C0 Reset** Write 1 to reset. This field is cleared by hardware when the reset is complete. 0: Normal operation. 1: Initiate reset.
15:14	-	RO	0	**Reserved**
13	spi1	R/W	0	**SPI1 Reset** Write 1 to reset. This field is cleared by hardware when the reset is complete. 0: Normal operation. 1: Initiate reset.
12	uart1	R/W	0	**UART1 Reset** Write 1 to reset. This field is cleared by hardware when the reset is complete. 0: Normal operation. 1: Initiate reset.
11	uart0	R/W	0	**UART0 Reset** Write 1 to reset. This field is cleared by hardware when the reset is complete. 0: Normal operation. 1: Initiate reset.
10:9	-	R/W	0	**Reserved**
8	tmr3	R/W	0	**TMR3 Reset** Write 1 to reset. This field is cleared by hardware when the reset is complete. 0: Normal operation. 1: Initiate reset.
7	tmr2	R/W	0	**TMR2 Reset** Write 1 to reset. This field is cleared by hardware when the reset is complete. 0: Normal operation. 1: Initiate reset.
6	tmr1	R/W	0	**TMR1 Reset** Write 1 to reset. This field is cleared by hardware when the reset is complete. 0: Normal operation. 1: Initiate reset.
5	tmr0	R/W	0	**TMR0 Reset** Write 1 to reset. This field is cleared by hardware when the reset is complete. 0: Normal operation. 1: Initiate reset.
4	-	RO	-	**Reserved**
3	gpio1	R/W	0	**GPIO1 Reset** Write 1 to reset. This field is cleared by hardware when the reset is complete. 0: Normal operation. 1: Initiate reset.
2	gpio0	R/W	0	**GPIO0 Reset** Write 1 to reset. This field is cleared by hardware when the reset is complete. 0: Normal operation. 1: Initiate reset.

Reset 0			GCR_RST0	[0x0004]
Bits	**Field**	**Access**	**Reset**	**Description**
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: Normal operation. 1: Initiate reset.
0	dma	R/W	0	**DMA Access Block Reset** Write 1 to reset. This field is cleared by hardware when the reset is complete. 0: Normal operation. 1: Initiate reset.

Table 4-68: Clock Control Register

Clock Control			GCR_CLKCTRL	[0x0008]
Bits	**Field**	**Access**	**Reset**	**Description**
31:30	-	DNM	0b10	**Reserved, Do Not Modify**
29	inro_rdy		0	**8kHz INRO Ready Status** 0: Not ready or not enabled. 1: Ready.
28	ibro_rdy	R	0	**7.3728MHz IBRO Ready Status** 0: Not ready. 1: Ready.
27	ipo_rdy	R	0	**120MHz IPO Ready Status** 0: Not ready or not enabled. 1: Ready.
26	iso_rdy	R	0	**60MHz ISO Ready Status** 0: Not ready or not enabled. 1: Ready.
25	ertco_rdy	R	0	**32.768kHz ERTCO Ready Status** 0: Not ready or not enabled. 1: Ready.
24	ebo_rdy	R	0	**25MHz EBO Ready Status** 0: Not ready or not enabled. 1: Ready.
23:22	-	RO	0	**Reserved**
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	**120MHz IPO Enable** 0: Disabled. 1: Enabled and ready when *GCR_CLKCTRL.ipo_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	ebo_en	R/W	0	**25MHz EBO Enable** 0: Disabled. 1: Enabled. *Note: The EBO can be enabled directly by setting this field to 1 or by enabling the IPLL (GCR_IPLL_CTRL.en = 1).*
15:14	-	RO	0	**Reserved**
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 completes. 0: Switch to new clock source not yet complete. 1: SYS_OSC is the clock source selected in GCR_CLKCTRL.*sysclk_sel*.
12	-	RO	0	**Reserved**
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 clears GCR_CLKCTRL.*sysclk_rdy* immediately. 0: ISO (POR and system reset default). 1: IPLL. 2: EBO. 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: $SYS\_CLK = \frac{SYS\_OSC}{2^{sysclk\_div}}$ *Note: Valid values are from 0 to 7 for sysclk_div.*
5:0	-	RO	8	**Reserved**

Table 4-69: Power Management Register

Power Management			**GCR_PM**	**0x000C**
Bits	**Field**	**Access**	**Reset**	**Description**
31:21	-	RO	0	**Reserved**

Power Management			**GCR_PM**	**0x000C**
Bits	**Field**	**Access**	**Reset**	**Description**
20	ebo_bp	R/W	0	**EBO Crystal Bypass** This field disables the oscillator for the EBO and allows an external clock source to drive the HFXIN pin. 0: Disable bypass. EBO time base is an external 25MHz crystal. 1: Enable bypass. EBO time base is an external square wave driven on HFXIN.
19:18	-	RO	0	**Reserved**
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**
9	aincomp_we	R/W	0	**Analog Input Comparator Wakeup Enable** This bit enables the Analog Input Comparator interrupt to wake the device from *SLEEP*, *LPM*, or *BACKUP*.
8	-	RO	0	**Reserved**
7	wut_we	R/W	0	**Wake-Up Timer Enable** Set this field to 1 to enable the wake-up timer as a wake-up source. The wake-up timer wakes the device from *SLEEP*, *LPM,* or *BACKUP*. 0: Wake-up source disabled. 1: Wake-up source enabled.
6	usb_we	R/W	0	**USB Wake-Up Enable** Set this field to 1 to enable the USB to wake the device. The USB wakes the device from *SLEEP*, *LPM*, or *BACKUP*. 0: Wake-up source disabled. 1: Wake-up source enabled.
5	rtc_we	R/W	0	**RTC Alarm Wake-Up Enable** Set this field to 1 to enable an RTC alarm to wake the device. The RTC alarm wakes the device from *SLEEP*, *LPM,* or *BACKUP*. 0: Wake-up source disabled. 1: Wake-up source enabled.
4	gpio_we	R/W	0	**GPIO Wake-Up Enable** Set this field to 1 to enable all GPIO pins as potential wake-up sources. Any GPIO configured for wake-up wakes the device from *SLEEP*, *LPM,* or *BACKUP*. 0: Wake-up source disabled. 1: Wake-up source enabled.

Power Management				GCR_PM	0x000C
Bits	**Field**	**Access**	**Reset**	**Description**	
3:0	mode	R/W	0	**Operating Mode** This field controls the operating mode of the device. 0: *ACTIVE.* 1: *SLEEP.* 2: *STANDBY.* 3: Reserved. 4: *BACKUP.* 5-7: Reserved. 8: *LPM* (CM4 *deep sleep*). 9: *UPM.* 10: *PDM.* 11-15: Reserved.	

Table 4-70: PLL Control Register

PLL Control				GCR_IPLL_CTRL	[0x0010]
Bits	**Field**	**Access**	**Reset**	**Description**	
31:2	-	RO	-	**Reserved**	
1	rdy	R	0	**PLL Ready Flag** This field is set to 1 after the PLL is enabled (*GCR_IPLL_CTRL.en* = 1) and the PLL is warmed up and ready for use. 0: Not ready. 1: Ready.	
0	en	R/W	0	**PLL Enable** Set this field to enable the PLL. 0: Disabled. 1: Enabled.	

Table 4-71: Peripheral Clock Divisor Register

Peripheral Clocks Divisor				GCR_PCLKDIV	[0x0018]
Bits	**Field**	**Access**	**Reset**	**Description**	
31:19	-	RO	-	**Reserved**	
18:17	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. 2: PCLK. 3: IPLL x 2 (200MHz).	

Peripheral Clocks Divisor			GCR_PCLKDIV	[0x0018]
Bits	**Field**	**Access**	**Reset**	**Description**
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: $\frac{CNN\_Clock}{1}$.
13:0	-	RO	-	**Reserved**

Table 4-72: Peripheral Clock Disable Register 0

Peripheral Clocks Disable 0			GCR_PCLKDIS0	[0x0024]
Bits	**Field**	**Access**	**Reset**	**Description**
31:30	-	R/W	1	**Reserved**
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**
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**
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**
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.

Peripheral Clocks Disable 0			GCR_PCLKDIS0	[0x0024]
Bits	**Field**	**Access**	**Reset**	**Description**
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	**TMR0 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**
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**
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**
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	-	RO	1	**Reserved**

Peripheral Clocks Disable 0		GCR_PCLKDIS0		[0x0024]
Bits	**Field**	**Access**	**Reset**	**Description**
3	usb	R/W	1	**USB 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.
2	-	RO	1	**Reserved**
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 4-73: Memory Clock Control Register

Memory Clock Control		GCR_MEMCTRL		[0x0028]
Bits	**Field**	**Access**	**Reset**	**Description**
31:17	-	RO	0	**Reserved**
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.
15:3	-	RO	0	**Reserved**
2:0	fws	R/W	5	**Program Flash Wait States** This field sets the number of wait-state cycles per flash memory read access. 0 – 7: Number of flash code access wait states *Note: For the IPO and ISO clocks, the minimum wait state is 2.* *Note: For all other clock sources, the minimum wait state is 0.*

Table 4-74: Memory Zeroize Control Register

Memory Zeroize		GCR_MEMZ		[0x002C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:12	-	RO	-	**Reserved**
11	usb	R/W1O	0	**USB RAM 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**
10	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.
9	icc0	R/W1O	0	**ICC0 Zeroization** Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Normal operation 1: Initiate zeroization
8	sysram0ecc	R/W1O	0	***Sysram0* ECC Zeroization** Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Normal operation 1: Initiate zeroization
7	ram7	R/W1O	0	***Sysram7* Zeroization** Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Normal operation 1: Initiate zeroization
6	ram6	R/W1O	0	***Sysram6* Zeroization** Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Normal operation 1: Initiate zeroization
5	ram5	R/W1O	0	***Sysram5* Zeroization** Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Normal operation 1: Initiate zeroization
4	ram4	R/W1O	0	***Sysram4* Zeroization** Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Normal operation 1: Initiate zeroization
3	ram3	R/W1O	0	***Sysram3* Zeroization** Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Normal operation 1: Initiate zeroization
2	ram2	R/W1O	0	***Sysram2* Zeroization** Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Normal operation 1: Initiate zeroization

Memory Zeroize			GCR_MEMZ	[0x002C]
Bits	**Field**	**Access**	**Reset**	**Description**
1	ram1	R/W1O	0	***Sysram1* Zeroization** Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Normal operation 1: Initiate zeroization
0	ram0	R/W1O	0	***Sysram0* Zeroization** Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Normal operation 1: Initiate zeroization

Table 4-75: System Status Flag Register

System Status Flag			GCR_SYSST	[0x0040]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
0	icelock	R	0	**Arm ICE Lock Status Flag** 0: Arm ICE is unlocked (enabled) 1: Arm ICE is locked (disabled)

Table 4-76: 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: Normal operation 1: Initiate reset
30:28	-	RO	0	**Reserved**
27	csi2	R/W	0	**CSI2 Block Reset** Write 1 to initiate the reset operation. 0: Normal operation 1: Initiate reset
26	pcif	R/W	0	**PCIF Block Reset** Write 1 to initiate the reset operation. 0: Normal operation 1: Initiate reset
25	simo	R/W	0	**Single Inductor Multiple Output Block Reset** Write 1 to initiate the reset operation. 0: Normal operation 1: Initiate reset
24	dvs	R/W	0	**Dynamic Voltage Scaling Controller Reset** Write 1 to initiate the operation. 0: Normal operation 1: Initiate reset

Reset 1			GCR_RST1	[0x0044]
Bits	**Field**	**Access**	**Reset**	**Description**
23:21	-	RO	0	**Reserved**
20	i2c2	R/W	0	**I2C2 Reset** Write 1 to initiate the operation. 0: Normal operation 1: Initiate reset
19	i2s	R/W	0	**I^{2}S Interface Reset** Write 1 to initiate the operation. 0: Normal operation 1: Initiate reset
18:17	-	R/W	0	**Reserved**
16	smphr	R/W	0	**Semaphore Block Reset** Write 1 to initiate the operation. 0: Normal operation 1: Initiate reset
15	-	RO	0	**Reserved**
14	csi2phy	R/W	0	**CSI2 PHY Reset** Write 1 to initiate the operation. 0: Normal operation 1: Initiate reset
13:12	-	RO	0	**Reserved**
11	spi0	R/W	0	**SPI0 Reset** Write 1 to initiate the operation. 0: Normal operation 1: Initiate reset
10	aes	R/W	0	**AES Block Reset** Write 1 to initiate the operation. 0: Normal operation 1: Initiate reset
9	crc	R/W	0	**CRC Reset** Write 1 to initiate the operation. 0: Normal operation 1: Initiate reset
8	-	R/W	0	**Reserved**
7	owm	R/W	0	**1-Wire Reset** Write 1 to initiate the operation. 0: Normal operation 1: Initiate reset
6	sdhc	R/W	0	**SDHC Reset** Write 1 to initiate the operation. 0: Normal operation 1: Initiate reset
5:2	-	RO	0	**Reserved**

Reset 1		GCR_RST1		[0x0044]
Bits	**Field**	**Access**	**Reset**	**Description**
1	pt	R/W	0	**Pulse Train Reset** Write 1 to initiate the operation. 0: Normal operation 1: Initiate reset
0	i2c1	R/W	0	**I2C1 Reset** Write 1 to initiate the operation. 0: Normal operation 1: Initiate reset

Table 4-77: 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	csi2	R/W	1	**CSI2 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
29:28	-	R/W	1	**Reserved**
27	wdt0	R/W	1	**Watchdog Timer 0 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
26:25	-	R/W	1	**Reserved**
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	**I^{2}S 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
22:19	-	R/W	1	**Reserved**
18	pcif	R/W	1	**PCIF 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**
17	-	RO	1	**Reserved**
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.
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.
12:11	-	RO	1	**Reserved**
10	sdhc	R/W	1	**SDHC 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
9	smphr	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	-	RO	1	**Reserved**
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	-	RO	1	**Reserved**

Table 4-78: Event Enable Register

Event Enable				GCR_EVENTEN	[0x004C]
Bits	**Field**	**Access**	**Reset**	**Description**	
31:3	-	RO	0	**Reserved**	
2	tx	R/W	0	**CPU0 (CM4) TXEV Event Enable** A TXEV event wakes the CM4 from a low-power mode entered with a WFE instruction when this bit is set. 0: Disabled 1: Enabled	
1	-	RO	0	**Reserved**	
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 4-79: Revision Register

Revision				GCR_REVISION	[0x0050]
Bits	**Field**	**Access**	**Reset**	**Description**	
31:16	-	RO	0	**Reserved**	
15:0	revision	R	*	**Device Revision** This field returns the chip revision ID as packed BCD. For example, 0x00A1 would indicate the device is revision A1.	

Table 4-80: System Status Interrupt Enable Register

System Status Interrupt Enable				GCR_SYSIE	[0x0054]
Bits	**Field**	**Access**	**Reset**	**Description**	
31:1	-	RO	-	**Reserved**	
0	iceunlock	R/W	0	**Arm ICE Unlocked Interrupt Enable** Set this field to generate an interrupt if the *GCR_SYSST.icelock* is set. 0: Interrupt disabled 1: Interrupt enabled	

Table 4-81: Error Correction Coding Error Register

Error Correction Coding Error				GCR_ECCERR	[0x0064]
Bits	**Field**	**Access**	**Reset**	**Description**	
31:1	-	RO	0	**Reserved**	
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 4-82: 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**
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.ram* is set to 1. 1: Correctable error detected.

Table 4-83: Error Correction Coding Interrupt Enable Register

Error Correction Coding Interrupt Enable			GCR_ECCIE	[0x006C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
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 4-84: 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 the read that produced the error. If the error is in 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 the read that 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*, this field represents the bits of the AMBA address of the read that 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/Cache 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 the read that produced the error. If the error is from the cache, then this bit is set as shown below: 0: No error 1: Cache data RAM error.

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

Table 4-85: General Purpose 0 Register

General Purpose 0			GCR_GPR0	[0x0080]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	R/W	0	**General Purpose Register** This field is a general-purpose register usable by software.

4.14 System Initialization Registers (SIR)

See *Table 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 4-86: System Initialization Register Summary

Offset	Register Name	Description
[0x0000]	*SIR_SISTAT*	*System Initialization Status Register*
[0x0004]	*SIR_SIADDR*	*System Initialization Address Error Register*
[0x0100]	*SIR_FSTAT*	*System initialization Function Status Register*
[0x0104]	*SIR_SFSTAT*	*System initialization Security Function Status Register*

4.14.1 System Initialization Register Details

Table 4-87: System Initialization Status Register

System Initialization Status			SIR_SISTAT	[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
31:2	-	RO	0	**Reserved**
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_SIADDR* register. *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 the SIR_SIADDR.erraddr.*

System Initialization Status		SIR_SISTAT		[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
0	magic	RO	See Description	**Configuration Valid Flag** This field is set to 1 by hardware during reset if the device configuration is valid. 0: OTP is not configured correctly 1: OTP configuration valid *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.*

Table 4-88: System Initialization Address Error Register

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

Table 4-89: System Initialization Function Status Register

System Initialization Function Status		SIR_FSTAT		[0x0100]
Bits	**Name**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
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	sdio	RO	See Description	**SDIO Block** This field indicates if the device includes the SDIO block. 0: Block is not available. 1: Block is available.
5:3	--	RO	0	**Reserved**
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**
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 4-90: System Initialization Security Function Status Register

System Initialization Security Function Status		SIR_SFSTAT		[0x0104]
Bits	**Name**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved**

System Initialization Security Function Status		SIR_SFSTAT		[0x0104]
Bits	**Name**	**Access**	**Reset**	**Description**
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**

4.15 Function Control Registers (FCR)

See *Table 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 4-91: Function Control Register Summary

Offset	Register	Description
[0x0000]	*FCR_FCTRL0*	Function Control 0 Register (I2C Glitch Filter Control)
[0x0004]	*FCR_AUTOCAL0*	IPO Automatic Calibration 0 Register
[0x0008]	*FCR_AUTOCAL1*	IPO Automatic Calibration 1 Register
[0x000C]	*FCR_AUTOCAL2*	IPO Automatic Calibration 2 Register
[0x0010]	*FCR_URVBOOTADDR*	RV32 Boot Address Register
[0x0014]	*FCR_URVCTRL*	RV32 Control Register
[0x0020]	FCR_TS0	Temperature Sensor Gain Trim Register
[0x0024]	FCR_TS1	Temperature Sensor Offset Trim Register
[0x0028]	FCR_ADCREFTRIM0	ADC 1.25V Reference Trim Register
[0x002C]	FCR_ADCREFTRIM1	ADC 2.048V Reference Trim Register
[0x0030]	*FCR_ADCREFTRIM2*	ADC External Reference Trim Register

4.15.1 Function Control Register Details

Table 4-92: Function Control 0 Register

Function Control 0		FCR_FCTRL0		[0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**
31:26	-	RO	0	**Reserved**
25	i2c2dgen1	R/W	0	**I2C2 SCL Deglitch Enable** 0: Disabled 1: Enabled
24	i2c2dgen0	R/W	0	**I2C2 SDA Deglitch Enable** 0: Disabled 1: Enabled
23	i2c1dgen1	R/W	0	**I2C1 SCL Deglitch Enable** 0: Disabled 1: Enabled

Function Control 0			FCR_FCTRL0	[0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**
22	i2c1dgen0	R/W	0	**I2C1 SDA Deglitch Enable** 0: Disabled 1: Enabled
21	i2c0dgen1	R/W	0	**I2C0 SCL Deglitch Enable** 0: Disabled 1: Enabled
20	i2c0dgen0	R/W	0	**I2C0 SDA Deglitch Enable** 0: Disabled 1: Enabled
19:18	-	RO	0	**Reserved**
17:16	usbclksel	R/W	0	**USB Core Clock Select** Set this field to the desired core clock for the USB peripheral. *Note: The selected clock must be enabled and if using the external clock, the GPIO pin must be configured for the correct alternate function selection.* 0: $\frac{IPO}{8}$ (12.5MHz) 1: External clock (P0.27) 2: ERFO
15:0	-	RO	0	**Reserved**

Table 4-93: 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**
19:8	mu	R/W	0	**IPO Trim Adaptation Gain**
7:5	-	RO	0	**Reserved**
4	atomic	R/W1O	0	**IPO Trim Atomic Start** Set this bit to start an automatic atomic calibration of the IPO. The calibration runs for *FCR_AUTOCAL2.donecnt* milliseconds. This bit is automatically cleared by hardware when the calibration is complete.
3	gaininv	R/W	0	**IPO Trim Step Invert** 0: IPO trim step is not inverted 1: IPO trim step is inverted
2	ldtrm	R/*	0	**IPO Initial Trim Load** Set this bit to load the initial trim value for the IPO from *FCR_AUTOCAL1.inittrm*. This bit is cleared by hardware once the load is complete.
1	en	R/W	0	**IPO Automatic Calibration Continuous Mode Enable** 0: Disabled 1: Enabled
0	acen	R/W	0	**IPO Trim Select** 0: Use default trim 1: Use automatic calibration trim values

Table 4-94: IPO Automatic Calibration 1 Register

IPO Automatic Calibration 1			**FCR_AUTOCAL1**	**[0x0008]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:9	-	R/W	0	**Reserved, Do Not Modify**
8:0	inittrm	R/W	0	**IPO Trim Automatic Calibration Initial Trim** This field contains the initial trim setting for the IPO.

Table 4-95: IPO Automatic Calibration 2 Register

IPO Automatic Calibration 2			**FCR_AUTOCAL2**	**[0x000C]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:21	-	RO	0	**Reserved**
20:8	acdiv	R/W	0	**IPO Trim Automatic Calibration Divide Factor** Target trim frequency for the IPO: $f_{IPO} = acdiv \times 32768$ *Note: Setting div to 0 is equivalent to setting div to 1.*
7:0	donecnt	R/W	0	**IPO Trim Automatic Calibration Run Time** $Atomic\ Run\ Time = donecnt\ (mS)$

Table 4-96: RV32 Boot Address Register

RV32 Boot Address			**FCR_URVBOOTADDR**	**[0x0010]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	R/W	0x2003 0000	**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 4-97: RV32 Control Register

RV32 Boot Address			**FCR_URVCTRL**	**[0x0014]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:2	-	RO	0	**Reserved**
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

RV32 Boot Address			FCR_URVCTRL	[0x0014]
Bits	**Field**	**Access**	**Reset**	**Description**
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. *Note: The application software must ensure that no accesses are occurring in sysram2 or sysram3 before setting this field to 1. See section Multiprocessor Communications for information on using the semaphore peripheral for communication between the RV32 and CM4 cores.*

Table 4-98: Temperature Sensor Gain Register

Temperature Sensor Gain			FCR_TS0	[0x0020]
Bits	**Field**	**Access**	**Reset**	**Description**
31:12	-	R	0	**Reserved**
11:0	gain	R	*	**Temperature Sensor Gain** This field contains the unsigned gain for the temperature sensor normalization. See *Temperature Sensor* for details. $°\text{C} = (ADC\ result \times FCR\_TS0.gain) + FCR\_TS1.offset$

Table 4-99: Temperature Sensor Offset Register

Temperature Sensor Offset			FCR_TS1	[0x0024]
Bits	**Field**	**Access**	**Reset**	**Description**
31:14	ts_offset_sign	R	*	**Sign Extension for Offset**
13:0	offset	R	*	**Temperature Sensor Offset** This field contains the signed offset for the temperature sensor normalization. See *Temperature Sensor* for details. $°\text{C} = (ADC\ result \times FCR\_TS0.gain) + FCR\_TS1.offset$

Table 4-100: ADC 1.25V Reference Trim Register

ADC 1.25V Reference Trim			FCR_ADCREFTRIM0	[0x0028]
Bits	**Field**	**Access**	**Reset**	**Description**
31:30	-	R	0	**Reserved**
29:24	vx2_tune		*	**Tuning Capacitor In-Line DAC** See *1.25V Internal Reference Trim* for details on this field.
23:18	-	RO	0	**Reserved**
17:16	vcm	R	*	**Trim Code for V_{CM} Output of Reference Buffer** See *1.25V Internal Reference Trim* for details on this field.
15	-	RO	0	**Reserved**

ADC 1.25V Reference Trim			FCR_ADCREFTRIM0	[0x0028]
Bits	**Field**	**Access**	**Reset**	**Description**
14:8	vrefm	R	*	**Trim Code for V_{REFM} Output of Reference Buffer** See *1.25V Internal Reference Trim* for details on this field.
7	-	RO	0	**Reserved**
6:0	vrefp	R	*	**Trim Code for V_{REFP} Output of Reference Buffer** See *1.25V Internal Reference Trim* for details on this field.

Table 4-101: ADC 2.048V Reference Trim Register

ADC 2.048V Reference Trim			FCR_ADCREFTRIM1	[0x002C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:30	-	R	0	**Reserved**
29:24	vx2_tune		*	**Tuning Capacitor In-Line DAC** See *2.048V Internal Reference Trim* for details on this field.
23:18	-	RO	0	**Reserved**
17:16	vcm	R	*	**Trim Code for V_{CM} Output of Reference Buffer** See *2.048V Internal Reference Trim* for details on this field.
15	-	RO	0	**Reserved**
14:8	vrefm	R	*	**Trim Code for V_{REFM} Output of Reference Buffer** See *2.048V Internal Reference Trim* for details on this field.
7	-	RO	0	**Reserved**
6:0	vrefp	R	*	**Trim Code for V_{REFP} Output of Reference Buffer** See *2.048V Internal Reference Trim* for details on this field.

Table 4-102: ADC External Reference Trim Register

ADC External Reference Trim			FCR_ADCREFTRIM2	[0x0030]
Bits	**Field**	**Access**	**Reset**	**Description**
31:30	-	RO	0	**Reserved**
29:24	vx2_tune	R	*	**Tuning Capacitor In-Line DAC** See *External Reference Trim* for details on this field.
23:18	-	RO	0	**Reserved**
17:16	vcm	R	*	**Trim Code for V_{CM} Output of Reference Buffer** See *External Reference Trim* for details on this field.
15	-	RO	0	**Reserved**
12	iboost_2p048	R	0	**Extra Drive Current Enable for 2.048V Reference** See *2.048V Internal Reference Trim* for details on this field.
11:8	idrv_2p048	R	*	**Trim Code for 2.048V Reference Buffer Drive Strength** See *2.048V Internal Reference Trim* for details on this field.
7:5	-	RO	0	**Reserved**

ADC External Reference Trim		FCR_ADCREFTRIM2		[0x0030]
Bits	**Field**	**Access**	**Reset**	**Description**
4	iboost_1p25	R	0	**Extra Drive Current Enable for 1.25V Reference** See *1.25V Internal Reference Trim* for details on this field.
3:0	idrv_1p25	R	*	**Trim Code for 1.25V Reference Buffer Drive Strength** See *1.25V Internal Reference Trim* for details on this field.

4.16 General Control Function Registers (GCFR)

See *Table 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 4-103: 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*

4.16.1 General Control Function Register Details

Table 4-104: General Control Function Register 0

General Control Function 0		GCFR_REG0		[0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved**
3	cnnx16_3_pwr_en	R/W	0	**V_{CNN3_EN} Power Domain Enable** 0: Disabled 1: Enabled *Note: This field controls the power domain enable pin (V_{CNN3_EN}) for ACTIVE, SLEEP, and LPM. During UPM, STANDBY, and BACKUP use the GCFR_REG2.cnnx16_3_data_ret_en field.*
2	cnnx16_2_pwr_en	R/W	0	**V_{CNN2_EN} Power Domain Enable** 0: Disabled 1: Enabled *Note: This field controls the power domain enable pin (V_{CNN2_EN}) for ACTIVE, SLEEP, and LPM. During UPM, STANDBY, and BACKUP use the GCFR_REG2.cnnx16_2_data_ret_en field.*
1	cnnx16_1_pwr_en	R/W	0	**V_{CCN1_EN} Power Domain Enable** 0: Disabled 1: Enabled *Note: This field controls the power domain pin (V_{CNN1_EN}) enable for ACTIVE, SLEEP, and LPM. During UPM, STANDBY, and BACKUP use the GCFR_REG2.cnnx16_1_data_ret_en field.*

General Control Function 0			GCFR_REG0	[0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**
0	cnnx16_0_pwr_en	R/W	0	**V_{CNN0_EN} Power Domain Enable** 0: Disabled 1: Enabled *Note: This field controls the power domain pin (V_{CNN0_EN}) enable for ACTIVE, SLEEP, and LPM. During UPM, STANDBY, and BACKUP use the GCFR_REG2.cnnx16_0_data_ret_en field.*

Table 4-105: General Control Function Register 1

General Control Function Register 1			GCFR_REG1	[0x0004]
Bits	**Field**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved**
3	cnnx16_3_ram_en	R/W	0	**$V_{CNN3RAM\_EN}$ Pin Enable** 0: Disabled 1: Enabled *Note: This field controls the pin enable state for $V_{CNN3RAM\_EN}$ in ACTIVE, SLEEP, and LPM. During UPM, STANDBY, and BACKUP this field is set to 0 by hardware. Use the GCFR_REG2.cnnx16_3_ram_data_ret_en field to control the retention enable for the CNN3 RAM.*
2	cnnx16_2_ram_en	R/W	0	**$V_{CNN2RAM\_EN}$ Pin Enable** 0: Disabled 1: Enabled *Note: This field controls the pin enable state for $V_{CNN2RAM\_EN}$ in ACTIVE, SLEEP, and LPM. During UPM, STANDBY, and BACKUP this field is set to 0 by hardware. Use the GCFR_REG2.cnnx16_2_ram_data_ret_en field to control the retention enable for the CNN2 RAM.*
1	cnnx16_1_ram_en	R/W	0	**$V_{CNN1RAM\_EN}$ Pin Enable** 0: Disabled 1: Enabled *Note: This field controls the pin enable state for $V_{CNN1RAM\_EN}$ in ACTIVE, SLEEP, and LPM. During UPM, STANDBY, and BACKUP this field is set to 0 by hardware. Use the GCFR_REG2.cnnx16_1_ram_data_ret_en field to control the retention enable for the CNN1 RAM.*
0	cnnx16_0_ram_en	R/W	0	**$V_{CNN0RAM\_EN}$ Pin Enable** 0: Disabled 1: Enabled *Note: This field controls the pin enable state for $V_{CNN0RAM\_EN}$ in ACTIVE, SLEEP, and LPM. During UPM, STANDBY, and BACKUP this field is set to 0 by hardware. Use the GCFR_REG2.cnnx16_0_ram_data_ret_en field to control the retention enable for the CNN0 RAM.*

Table 4-106: General Control Function Register 2

General Control Function Register 2			GCFR_REG2	[0x0008]
Bits	**Field**	**Access**	**Reset**	**Description**
31:24	-	RO	0	**Reserved**

General Control Function Register 2		GCFR_REG2		[0x0008]
Bits	**Field**	**Access**	**Reset**	**Description**
23	cnnx16_3_ram_data_ret_en	R/W	0	**$V_{CNN3RAM\_EN}$ Data Retention Enable for UPM, STANDBY, and BACKUP** Set this field to 1 to enable data retention for CNN3's RAM during *UPM*, *STANDBY*, and *BACKUP*. 0: Disabled 1: Enabled
22	cnnx16_2_ram_data_ret_en	R/W	0	**$V_{CNN2RAM\_EN}$ Data Retention Enable for UPM, STANDBY, and BACKUP** Set this field to 1 to enable data retention for CNN2's RAM during *UPM*, *STANDBY*, and *BACKUP*. 0: Disabled 1: Enabled
21	cnnx16_1_ram_data_ret_en	R/W	0	**$V_{CNN1RAM\_EN}$ Data Retention Enable for UPM, STANDBY, and BACKUP** Set this field to 1 to enable data retention for CNN1's RAM during *UPM*, *STANDBY*, and *BACKUP*. 0: Disabled 1: Enabled
20	cnnx16_0_ram_data_ret_en	R/W	0	**$V_{CNN0RAM\_EN}$ Data Retention Enable for UPM, STANDBY, and BACKUP** Set this field to 1 to enable data retention for CNN0's RAM during *UPM*, *STANDBY*, and *BACKUP*. 0: Disabled 1: Enabled
19	cnnx16_3_data_ret_en	R/W	0	**V_{CNN3_EN} Power Switch Enable for UPM, STANDBY, and BACKUP** Set this field to 1 to enable data retention for CNN3 during *UPM*, *STANDBY*, and *BACKUP*. 0: Disabled 1: Enabled
18	cnnx16_2_data_ret_en	R/W	0	**V_{CNN2_EN} Power Switch Enable for UPM, STANDBY, and BACKUP** Set this field to 1 to enable data retention for CNN2 during *UPM*, *STANDBY*, and *BACKUP*. 0: Disabled 1: Enabled
17	cnnx16_1_data_ret_en	R/W	0	**V_{CNN1_EN} Power Switch Enable for UPM, STANDBY, and BACKUP** Set this field to 1 to enable data retention for CNN1 during *UPM*, *STANDBY*, and *BACKUP*. 0: Disabled 1: Enabled
16	cnnx16_0_data_ret_en	R/W	0	**V_{CNN0_EN} Power Switch Enable for UPM, STANDBY, and BACKUP** Set this field to 1 to enable data retention for CNN0 during *UPM*, *STANDBY*, and *BACKUP*. 0: Disabled 1: Enabled
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

General Control Function Register 2		GCFR_REG2		[0x0008]
Bits	**Field**	**Access**	**Reset**	**Description**
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 4-107: General Control Function Register 3

General Control Function Register 3		GCFR_REG3		[0x000C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved**
3	cnnx16_3_rst	R/W	0	**CNNx16_3 Power Domain Reset** Write this field to 1 to initiate a power domain reset for the CNNx16_3. 0: Normal operation 1: Initiate reset
2	cnnx16_2_rst	R/W	0	**CNNx16_2 Power Domain Reset** Write this field to 1 to initiate a power domain reset for the CNNx16_2. 0: Normal operation 1: Initiate reset
1	cnnx16_1_rst	R/W	0	**CNNx16_1 Power Domain Reset** Write this field to 1 to initiate a power domain reset for the CNNx16_1. 0: Normal operation 1: Initiate reset
0	cnnx16_0_rst	R/W	0	**CNNx16_0 Power Domain Reset** Write this field to 1 to initiate a power domain reset for the CNNx16_0. 0: Normal operation 1: Initiate reset

5. Interrupts and Exceptions

Interrupts and exceptions are managed by either the Arm Cortex-M4 with FPU NVIC or the RV32 interrupt controller. The NVIC manages the interrupts, exceptions, priorities, and masking. *Table 5-1* and *Table 5-2* detail the MAX78002's interrupt vector tables for the CM4 and RV32 processors, respectively, and describe each exception and interrupt.

5.1 CM4 Interrupt and Exception Features

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

5.2 CM4 Interrupt Vector Table

Table 5-1 lists the interrupt and exception table for the MAX78002's CM4 core. There are 105 interrupt entries for the MAX78002, including reserved interrupt placeholders. Including the 15 system exceptions for the Arm Cortex-M4 with FPU, the total number of entries is 120.

Table 5-1: MAX78002 CM4 Interrupt Vector Table

Exception (Interrupt) Number	Offset	Name	Description
1	[0x0004]	Reset_IRQn	Reset
2	[0x0008]	NonMaskableInt_IRQn	Non-Maskable Interrupt
3	[0x000C]	HardFault_IRQn	Hard Fault
4	[0x0010]	MemoryManagement_IRQn	Memory Management Fault
5	[0x0014]	BusFault_IRQn	Bus Fault
6	[0x0018]	UsageFault_IRQn	Usage Fault
7:10	[0x001C]-[0x0028]	-	Reserved
11	[0x002C]	SVCall_IRQn	Supervisor Call Exception
12	[0x0030]	DebugMonitor_IRQn	Debug Monitor Exception
13	[0x0034]	-	Reserved
14	[0x0038]	PendSV_IRQn	Request Pending for System Service
15	[0x003C]	SysTick_IRQn	System Tick Timer
16	[0x0040]	PF_IRQn	Power Fail interrupt
17	[0x0044]	WDT0_IRQn	Windowed Watchdog Timer 0 Interrupt
18	[0x0048]	USB_IRQn	Reserved
19	[0x004C]	RTC_IRQn	Reserved
20	[0x0050]	TRNG_IRQn	True Random Number Generator Interrupt
21	[0x0054]	TMR0_IRQn	Timer 0 Interrupt
22	[0x0058]	TMR1_IRQn	Timer 1 Interrupt
23	[0x005C]	TMR2_IRQn	Timer 2 Interrupt
24	[0x0060]	TMR3_IRQn	Timer 3 Interrupt
25	[0x0064]	TMR4_IRQn	Timer 4 (LPTMR0) Interrupt
26	[0x0068]	TMR5_IRQn	Timer 5 (LPTMR1) Interrupt
27:28	[0x006C]:[0x0070]	-	Reserved
29	[0x0074]	I2C0_IRQn	I^{2}C Port 0 Interrupt
30	[0x0078]	UART0_IRQn	UART Port 0 Interrupt

Exception (Interrupt) Number	Offset	Name	Description
31	[0x007C]	UART1_IRQn	UART Port 1 Interrupt
32	[0x0080]	SPI1_IRQn	SPI Port 1 Interrupt
33:35	[0x0084]:[0x008C]	-	Reserved
36	[0x0090]	ADC_IRQn	ADC Interrupt
37:38	[0x0094]:[0x0098]	-	Reserved
39	[0x009C]	FLC0_IRQn	Flash Controller 0 Interrupt
40	[0x00A0]	GPIO0_IRQn	GPIO Port 0 Interrupt
41	[0x00A4]	GPIO1_IRQn	GPIO Port 1 Interrupt
42	[0x00A8]	GPIO2_IRQn	GPIO Port 2 Interrupt
43	[0x00AC]	-	Reserved
44	[0x00B0]	DMA0_IRQn	DMA0 Interrupt
45	[0x00B4]	DMA1_IRQn	DMA1 Interrupt
46	[0x00B8]	DMA2_IRQn	DMA2 Interrupt
47	[0x00BC]	DMA3_IRQn	DMA3 Interrupt
48:49	[0x00C0 : 0x00C4]	-	Reserved
50	[0x00C8]	UART2_IRQn	UART Port 2 Interrupt
51	[0x00CC]	-	Reserved
52	[0x00D0]	I2C1_IRQn	I^{2}C Port 1 Interrupt
53:68	[0x00D4]: [0x0110]	-	Reserved
69	[0x0114]	WUT_IRQn	Wakeup Timer Interrupt
70	[0x0118]	GPIOWAKE_IRQn	GPIO Wakeup Interrupt
71	[0x011C]	-	Reserved
72	[0x0120]	SPI0_IRQn	SPI Port 0 Interrupt
73	[0x0124]	WDT1_IRQn	Low Power Watchdog Timer 0 (WDT1) Interrupt
74	[0x0128]	-	Reserved
75	[0x012C]	PT_IRQn	Pulse Train Interrupt
76:77	[0x0130]:[0x0134]	-	Reserved
78	[0x0138]	I2C2_IRQn	I^{2}C Port 2 Interrupt
79	[0x013C]	RISCV_IRQn	CPU1 (RV32) Interrupt
80:81	[0x0140]:[0x0144]	-	Reserved
82	[0x0148]	SDHC_IRQn	SDHC Interrupt
83	[0x014C]	OWM_IRQn	1-Wire Master Interrupt
84:95	[0x0150]:[0x017C]	-	Reserved
96	[0x0180]	USBDMA_IRQn	USB DMA Interrupt
97	[0x0184]	-	Reserved
98	[0x0188]	ECC_IRQn	Error Correction Coding Block Interrupt
99	[0x018C]	DVS_IRQn	Digital Voltage Scaling Interrupt
100	[0x0190]	SIMO_IRQn	Single Input Multiple Output Interrupt
101:103	[0x0194]:[0x019C]	-	Reserved
104	[0x01A0}	UART3_IRQn	UART3 (LPUART0) Interrupt
105:106	[0x01A4]:[0x01A8]	-	Reserved
107	[0x01AC]	PCIF_IRQn	Parallel Camera Interface Interrupt

Exception (Interrupt) Number	Offset	Name	Description
108:112	[0x01B0]:[0x01C0]	-	Reserved
113	[0x01C4]	AES_IRQn	AES Interrupt
114	[0x01C8]	CRC_IRQn	CRC Interrupt
115	[0x01CC]	I2S_IRQn	I^{2}S Interrupt
116	[0x01D0]	CNN_FIFO_IRQn	CNN FIFO Interrupt
117	[0x01D4]	CNN_IRQn	CNN Interrupt
118	[0x01D8]	-	Reserved
119	[0x01DC]	LPCMP_IRQn	Low Power Comparator Interrupt
120	[0x01E0]	CSI2_IRQn	MIPI CSI 2 Interrupt

5.3 RV32 Interrupt Vector Table

Table 5-2 lists the interrupt and exception table for the MAX78002's RV32 core.

Table 5-2: MAX78002 RV32 Interrupt Vector Table

Exception (Interrupt) Number	Name	Description
4	PF_IRQn	System Fault interrupt
5	WDT0_IRQn	Windowed Watchdog Timer 0 Interrupt
6	GPIOWAKE_IRQn	GPIO Wakeup Interrupt
7	RTC_IRQn	RTC Interrupt
8	TMR0_IRQn	Timer 0 Interrupt
9	TMR1_IRQn	Timer 1 Interrupt
10	TMR2_IRQn	Timer 2 Interrupt
11	TMR3_IRQn	Timer 3 Interrupt
12	TMR4_IRQn	Timer 4 (LPTMR0) Interrupt
13	TMR5_IRQn	Timer 5 (LPTMR1) Interrupt
14	I2C0_IRQn	I^{2}C Port 0 Interrupt
15	UART0_IRQn	UART Port 0 Interrupt
16	-	Reserved
17	I2C1_IRQn	I^{2}C Port 1 Interrupt
18	UART1_IRQn	UART Port 1 Interrupt
19	UART2_IRQn	UART Port 2 Interrupt
20	I2C2_IRQn	I^{2}C Port 2 Interrupt
21	UART3_IRQn	UART3 (LPUART0) Interrupt
22	SPI1_IRQn	SPI Port 1 Interrupt
23	WUT_IRQn	Wakeup Timer Interrupt
24	FLC0_IRQn	Flash Controller 0 Interrupt
25	GPIO0_IRQn	GPIO Port 0 Interrupt
26	GPIO1_IRQn	GPIO Port 1 Interrupt
27	GPIO2_IRQn	GPIO Port 2 Interrupt
28	DMA0_IRQn	DMA0 Interrupt
29	DMA1_IRQn	DMA1 Interrupt
30	DMA2_IRQn	DMA2 Interrupt

Exception (Interrupt) Number	Name	Description
31	DMA3_IRQn	DMA3 Interrupt
32:45	-	Reserved
46	AES_IRQn	AES Interrupt
47	TRNG_IRQn	TRNG Interrupt
48	WDT1_IRQn	Watchdog Timer 1 (LPWDT0) Interrupt
49	DVS_IRQn	Digital Voltage Scaling Interrupt
50	SIMO_IRQn	Single Input Multiple Output Interrupt
51	CRC_IRQn	CRC Interrupt
52	PT_IRQn	Pulse Train Interrupt
53	ADC_IRQn	ADC Interrupt
54	OWM_IRQn	1-Wire Master Interrupt
55	I2S_IRQn	I^{2}S Interrupt
56	CNN_FIFO_IRQn	CNN TX FIFO Interrupt
57	CNN_IRQn	CNN Interrupt
58	-	Reserved
59	PCIF_IRQn	Parallel Camera Interface Interrupt

6. 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 that 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 through the *GPIOn_IN* register.

The electrical characteristics of a GPIO pin are identical whether the pin is configured as an I/O or as an 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: The register set used to control the GPIO are identical across multiple Maxim Integrated microcontrollers, however the behavior of several registers vary depending on the specific device. The behavior of the registers should not be assumed to be the same from one device to a different device. Specifically the registers GPIOn_PADCTRL0, GPIOn_PADCTRL1, GPIOn_HYSEN, GPIOn_SRSEL, GPIOn_DS0, GPIOn_DS1, and GPIOn_VSSEL are device dependent in their usage. GPIO3 is controlled differently and has different features than the other GPIO ports in the MAX78002. 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:
  - V_{DDIO}
  - V_{DDIOH}
- Selectable interrupt events:
  - Level triggered low
  - Level triggered high
  - Edge triggered rising edge
  - Edge triggered falling edge
  - Edge triggered rising and falling edge
- All GPIO pins default to input mode with weak-pullup during power-on-reset events.

6.1 Instances

Table 6-1 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 6-1: MAX78002 GPIO Pin Count

Package	GPIO	PINS
144-CSBGA	GPIO0[31:0]	31
	GPIO1[17:0]	10
	GPIO2[7:0]	8
	GPIO3[1:0][†]	2

Note: See Power Sequencer Registers (PWRSEQ) for details on using GPIO3.

Note: Refer to the MAX78002 device data sheet for a description of alternate functions for each GPIO port pin.

6.2 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.

6.2.1 Power-On-Reset Configuration

During a POR event, all I/O default to GPIO mode as high impedance inputs except the SWDIO and SWDCLK pins. The SWD is enabled by default after POR with AF1 selected by hardware. See the *Bootloader* chapter for exceptions.

Following a POR event, all GPIO except device pins that have the SWDIO and SWDCLK function, are configured with the following default settings:

- GPIO mode enabled
  - *GPIOn_EN0.[pin]* = 1
  - *GPIOn_EN1.[pin]* = 0
  - *GPIOn_EN2.[pin]* = 0
- Pullup/Pulldown disabled, I/O in Hi-Z mode
  - *GPIOn_PADCTRL0.[pin]* = 0
  - *GPIOn_PADCTRL1.[pin]*
- Output mode disabled
  - *GPIOn_OUTEN.[pin]* = 0
- Interrupt disabled
  - *GPIOn_INTEN.[pin]* = 0

6.2.2 Serial Wire Debug Configuration

Perform the following steps to configure the SWDIO and SWDCLK device pins for SWD mode:

1. Set the device pin P0.28 for AF1 mode:
   a. *GPIOn_EN0.[28]* = 0
   b. *GPIOn_EN1.[28]* = 0
   c. *GPIOn_EN2.[28]* = 0
2. Set device pin P0.29 for AF1 mode:
   a. *GPIOn_EN0.[28]* = 0
   b. *GPIOn_EN1.[29]* = 0
   c. *GPIOn_EN2.[29]* = 0

Note: To use the SWD pins in I/O mode, set the desired GPIO pins for SWD AF and set the SWD disable field to 1 (GCR_SYSCTRL.swd_dis = 1).

6.2.3 Pin Function Configuration

Table 6-2 depicts the bit settings for the *GPIOn_EN0*, *GPIOn_EN1*, and *GPIOn_EN2* registers to configure the function of the GPIO port pins. Each of the bits within these registers represents the configuration of a single pin on the GPIO port. For example, *GPIO0_EN0.[25]*, *GPIO0_EN1.[25]*, and *GPIO0_EN2.[25]* all represent configuration for device pin P0.25. See *Table 6-5* for a detailed example of how each of these bits applies to each of the GPIO device pins.

Table 6-2: MAX78002 GPIO Pin Function Configuration

MODE	*GPIOn_EN0.[pin]*	*GPIOn_EN1.[pin]*	*GPIOn_EN2.[pin]*
AF1	0	0	0
AF2	0	1	0
I/O (transition to AF1)	1	0	0
I/O (transition to AF2)	1	1	0

6.2.4 Input Mode Configuration

Table 6-3 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_PADCTRL1.[25]*, *GPIO0_PADCTRL0.[25]*, *GPIO0_PS.[25]*, and *GPIO0_VSSEL.[25]* all represent configuration for device pin P0.25. See *Table 6-8* for a detailed example of how each of these bits applies to each of the GPIO device pins. Refer to the MAX78002 data sheet for details of specific electrical characteristics.

Table 6-3: MAX78002 Input Mode Configuration

Input Mode	Mode Select		Pullup/Pulldown Strength	Power Supply
	GPIOn_PADCTRL1.[pin]	*GPIOn_PADCTRL0.[pin]*	*GPIOn_PS.[pin]*	*GPIOn_VSSEL.[pin]*
High-impedance	0	0	N/A	N/A
Weak Pullup to V_{DDIO} (1MΩ)	0	1	0	0
Strong Pullup to V_{DDIO} (25KΩ)	0	1	1	0

Input Mode	Mode Select		Pullup/Pulldown Strength	Power Supply
	GPIOn_PADCTRL1.[pin]	*GPIOn_PADCTRL0.[pin]*	*GPIOn_PS.[pin]*	*GPIOn_VSSEL.[pin]*
Weak Pulldown to V_{DDIOH} (1MΩ)	1	0	0	1
Strong Pulldown to V_{DDIOH} (25KΩ)	1	0	1	1
Reserved	1	1	N/A	N/A

6.2.5 Output Mode Configuration

Table 6-4 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, *GPIO2_DS0.[7], GPIO2_DS1.[7], and GPIO2_VSSEL.[7]* all represent configuration for device pin P2.7. See Table 6-8 for a detailed example of how each of these bits applies to each of the GPIO device pins. Refer to the MAX78002 data sheet for details of specific electrical characteristics.

Table 6-4: MAX78002 Output Mode Configuration

Input Mode	Drive Strength		Power Supply
	GPIOn_DS1.[pin]	*GPIOn_DS0.[pin]*	*GPIOn_VSSEL.[pin]*
Output Drive Strength 0, V_{DDIO} Supply	0	0	0
Output Drive Strength 1, V_{DDIO} Supply	0	1	0
Output Drive Strength 2, V_{DDIO} Supply	1	0	0
Output Drive Strength 3, V_{DDIO} Supply	1	1	0
Output Drive Strength 0, V_{DDIOH} Supply	0	0	1
Output Drive Strength 1, V_{DDIOH} Supply	0	1	1
Output Drive Strength 2, V_{DDIOH} Supply	1	0	1
Output Drive Strength 3, V_{DDIOH} Supply	1	1	1

Each GPIO port is assigned a dedicated interrupt vector as shown in Table 6-9.

6.3 Reference Tables

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

Table 6-5: MAX78002 GPIO Alternate Function Configuration Reference

Device Pin	Alternate Function Configuration Bits		
P0.0	*GPIO0_EN0.[0]*	*GPIO0_EN1.[0]*	*GPIO0_EN2.[0]*
P0.1	*GPIO0_EN0.[1]*	*GPIO0_EN1.[1]*	*GPIO0_EN2.[1]*
…	…	…	…
P0.30	*GPIO0_EN0.[30]*	*GPIO0_EN1.[30]*	*GPIO0_EN2.[30]*
P0.31	*GPIO0_EN0.[31]*	*GPIO0_EN1.[31]*	*GPIO0_EN2.[31]*

Table 6-6: MAX78002 GPIO Output/Input Configuration Reference

Device Pin	GPIO Output Enable	GPIO Output Write	GPIO Input Enable	GPIO Input Read
P0.0	*GPIO0_OUTEN.[0]*	*GPIO0_OUT.[0]*	*GPIO0_INEN.[0]*	*GPIO0_IN.[0]*
P0.1	*GPIO0_OUTEN.[1]*	*GPIO0_OUT.[1]*	*GPIO0_INEN.[1]*	*GPIO0_IN.[1]*
…	…	…	…	…
P0.30	*GPIO0_OUTEN.[30]*	*GPIO0_OUT.[30]*	*GPIO0_INEN.[30]*	*GPIO0_IN.[30]*
P0.31	*GPIO0_OUTEN.[31]*	*GPIO0_OUT.[31]*	*GPIO0_INEN.[31]*	*GPIO0_IN.[31]*

Table 6-7: MAX78002 GPIO Interrupt Configuration Reference

Device Pin	Enable	Status	Dual Edge	Polarity	Trigger	Wakeup
P0.0	*GPIO0_INTEN.[0]*	*GPIO0_INTFL.[0]*	*GPIO0_DUALEDGE.[0]*	*GPIO0_INTPOL.[0]*	*GPIO0_INTMODE.[0]*	*GPIO0_WKEN.[0]*
P0.1	*GPIO0_INTEN.[1]*	*GPIO0_INTFL.[1]*	*GPIO0_DUALEDGE.[1]*	*GPIO0_INTPOL.[1]*	*GPIO0_INTMODE.[1]*	*GPIO0_WKEN.[1]*
…	…	…	…	…	…	…
P0.30	*GPIO0_INTEN.[30]*	*GPIO0_INTFL.[30]*	*GPIO0_DUALEDGE.[30]*	*GPIO0_INTPOL.[30]*	*GPIO0_INTMODE.[30]*	*GPIO0_WKEN.[30]*
P0.31	*GPIO0_INTEN.[31]*	*GPIO0_INTFL.[31]*	*GPIO0_DUALEDGE.[31]*	*GPIO0_INTPOL.[31]*	*GPIO0_INTMODE.[31]*	*GPIO0_WKEN.[31]*

Table 6-8: MAX78002 GPIO Pullup/Pulldown/Drive Strength/Voltage Configuration Reference

Device Pin	Pullup/Pulldown/Strength Select			Drive Strength		Voltage
P0.0	*GPIO0_PADCTRL0.[0]*	*GPIO0_PADCTRL1.[0]*	*GPIO0_PS.[0]*	*GPIO0_DS0.[0]*	*GPIO0_DS1.[0]*	*GPIO0_VSSEL.[0]*
P0.1	*GPIO0_PADCTRL0.[1]*	*GPIO0_PADCTRL1.[1]*	*GPIO0_PS.[1]*	*GPIO0_DS0.[1]*	*GPIO0_DS1.[1]*	*GPIO0_VSSEL.[1]*
…	…	…	…	…	…	…
P0.30	*GPIO0_PADCTRL0.[30]*	*GPIO0_PADCTRL1.[30]*	*GPIO0_PS.[30]*	*GPIO0_DS0.[30]*	*GPIO0_DS1.[30]*	*GPIO0_VSSEL[30]*
P0.31	*GPIO0_PADCTRL0.[31]*	*GPIO0_PADCTRL1.[31]*	*GPIO0_PS.[31]*	*GPIO0_DS0.[31]*	*GPIO0_DS1.[31]*	*GPIO0_VSSEL.[31]*

6.4 Usage

6.4.1 *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 6-2* are set to I/O (transition to AF1) mode.
2. Input mode is enabled (*GPIOn_INEN*.*[pin]* = 1).
3. High impedance mode enabled (*GPIOn_PADCTRL1*.*[pin]* = 0, *GPIOn_PADCTRL0*.*[pin]* = 0), pullup and pulldown disabled.
4. Output mode disabled (*GPIOn_OUTEN*.*[pin]* = 0)
5. Interrupt disabled (*GPIOn_INTEN*.*[pin]* = 0)

6.4.2 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 6-2* to any one of the I/O mode settings.
2. Configure the electrical characteristics of the pin as desired as shown in *Table 6-3*.
3. Enable the input buffer connection to the GPIO pin by setting *GPIOn_INEN.[pin]* to 1.
4. Read the input state of the pin using the *GPIOn_IN.[pin]* field.

6.4.3 Output Mode Configuration

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

1. Set the GPIO configuration enable bits shown in *Table 6-2* to any one of the I/O mode settings.
2. Configure the electrical characteristics of the pin as desired as shown in *Table 6-4*.
3. Set the output logic high or logic low using the *GPIOn_OUT.[pin]* bit.
4. Enable the output buffer for the pin by setting *GPIOn_OUTEN.[pin]* to 1.

6.4.4 Alternate Function Configuration

Most GPIO support one or more alternate functions selected with the GPIO configuration enable bits shown in *Table 6-2*. The bits that select the AF must only be changed while the pin is in one of the I/O modes (*GPIOn_EN0.[pin]* = 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 6-2* 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 does not affect the state or electrical characteristics of the pin.
2. Configure the electrical characteristics of the pin. See *Table 6-3* if the assigned alternate function uses the pin as an input. See *Table 6-4* if the assigned alternate function uses the pin as an output.
3. Set the GPIO configuration enable bits shown in *Table 6-2* to the desired alternate function.

6.5 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_INTFL* register. A GPIO interrupt occurs when the status bit transitions from 0 to 1 if the corresponding bit is set in the corresponding *GPIOn_INTEN* register. Note that the interrupt status bit is always set when the current interrupt configuration event occurs, but an interrupt is only 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_INTEN*.*[pin]* field to 0. This prevents any new interrupts on the pin from triggering but does not clear previously triggered (pending) interrupts. The application can disable all interrupts for a GPIO port by writing 0 to the *GPIOn_INEN* register. To maintain previously enabled interrupts, read the *GPIOn_INEN* register and save the state prior to setting the register to 0.
2. Clear pending interrupts by writing 1 to the *GPIOn_INTFL_CLR*.*[pin]* bit.
3. Configure the pin for the desired interrupt event
4. Set *GPIOn_INTMODE*.*[pin]* to select the desired interrupt.
5. For level triggered interrupts, the interrupt triggers on an input high (*GPIOn_INTPOL*.*[pin]* = 0) or input low level.
6. For edge triggered interrupts, the interrupt triggers on a transition from low to high(*GPIOn_INTPOL*.*[pin]* = 0) or high to low (*GPIOn_INTPOL*.*[pin]* = 1).
7. Optionally, set *GPIOn_DUALEDGE*.*[pin]* to 1 to trigger on both the rising and falling edges of the input signal.
    a. Set *GPIOn_INTEN*.*[pin]* to 1 to enable the interrupt for the pin.

6.5.1 GPIO Interrupt Handling

Each GPIO port is assigned its own dedicated interrupt vector as shown in *Table 6-9*.

Table 6-9: MAX78002 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]	*GPIOn_INTFL*	40	25	GPIO0_IRQn
GPIO1[17:0]	*GPIOn_INTFL*	41	26	GPIO1_IRQn
GPIO2[7:0]	*GPIOn_INTFL*	42	27	GPIO2_IRQn

To handle GPIO interrupts in the interrupt vector handler, complete the following steps:

1. Read the *GPIOn_INTFL* 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_INTFL* register by writing a 1 to the *GPIOn_INTFL_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.

6.5.2 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_IRQn, is assigned for all pins of all GPIO ports. When the GPIO wakeup event occurs, the application software must interrogate each *GPIOn_INTFL* register to determine which external port pin caused the wakeup event.

Table 6-10: MAX78002 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	*GPIO0_INTFL*	70	6	GPIOWAKE_IRQn
GPIO1	*GPIO1_INTFL*	70	6	GPIOWAKE_IRQn
GPIO2	*GPIO2_INTFL*	70	6	GPIOWAKE_IRQn

To enable low-power mode wakeup (*SLEEP*, *DEEPSLEEP*, *LPM*, *UPM*, and *BACKUP*) using an external GPIO interrupt, complete the following steps:

1. Clear pending interrupt flags by writing a logic 1 to *GPIOn_INTFL_CLR*.*[pin]*.
2. Activate the GPIO wakeup function by writing a logic 1 to *GPIOn_WKEN*.*[pin]*.
3. Configure the power manager to use the GPIO as a wakeup source by *GCR_PM*.*gpio_we* field to 1.

6.6 Registers

See *Table 3-3* 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 6-11*. 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 1-1* 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 6-11: GPIO Register Summary

Offset	Register	Description
[0x0000]	*GPIOn_EN0*	*GPIO Port n Configuration Enable Bit 0 Register*
[0x0004]	*GPIOn_EN0_SET*	*GPIO Port n Configuration Enable Atomic Set Bit 0 Register*
[0x0008]	*GPIOn_EN0_CLR*	*GPIO Port n Configuration Enable Atomic Clear Bit 0 Register*
[0x000C]	*GPIOn_OUTEN*	*GPIO Port n Output Enable Register*
[0x0010]	*GPIOn_OUTEN_SET*	*GPIO Port n Output Enable Atomic Set Register*
[0x0014]	*GPIOn_OUTEN_CLR*	*GPIO Port n Output Enable Atomic Clear Register*
[0x0018]	*GPIOn_OUT*	*GPIO Port n Output Register*
[0x001C]	*GPIOn_OUT_SET*	*GPIO Port n Output Atomic Set Register*
[0x0020]	*GPIOn_OUT_CLR*	*GPIO Port n Output Atomic Clear Register*
[0x0024]	*GPIOn_IN*	*GPIO Port n Input Register*
[0x0028]	*GPIOn_INTMODE*	*GPIO Port n Interrupt Mode Register*
[0x002C]	*GPIOn_INTPOL*	*GPIO Port n Interrupt Polarity Register*
[0x0030]	*GPIOn_INEN*	*GPIO Port n Input Enable Register*
[0x0034]	*GPIOn_INTEN*	*GPIO Port n Interrupt Enable Register*
[0x0038]	*GPIOn_INTEN_SET*	*GPIO Port n Interrupt Enable Atomic Set Register*
[0x003C]	*GPIOn_INTEN_CLR*	*GPIO Port n Interrupt Enable Atomic Clear Register*
[0x0040]	*GPIOn_INTFL*	*GPIO Port n Interrupt Status Register*
[0x0048]	*GPIOn_INTFL_CLR*	*GPIO Port n Interrupt Clear Register*
[0x004C]	*GPIOn_WKEN*	*GPIO Port n Wakeup Enable Register*
[0x0050]	*GPIOn_WKEN_SET*	*GPIO Port n Wakeup Enable Atomic Set Register*
[0x0054]	*GPIOn_WKEN_CLR*	*GPIO Port n Wakeup Enable Atomic Clear Register*
[0x005C]	*GPIOn_DUALEDGE*	*GPIO Port n Interrupt Dual Edge Mode Register*
[0x0060]	*GPIOn_PADCTRL0*	*GPIO Port n Pad Configuration 1 Register*
[0x0064]	*GPIOn_PADCTRL1*	*GPIO Port n Pad Configuration 2 Register*
[0x0068]	*GPIOn_EN1*	*GPIO Port n Configuration Enable Bit 1 Register*
[0x006C]	*GPIOn_EN1_SET*	*GPIO Port n Configuration Enable Atomic Set Bit 1 Register*
[0x0070]	*GPIOn_EN1_CLR*	*GPIO Port n Configuration Enable Atomic Clear Bit 1 Register*
[0x0074]	*GPIOn_EN2*	*GPIO Port n Configuration Enable Bit 2 Register*

Offset	Register	Description
[0x0078]	*GPIOn_EN2_SET*	*GPIO Port n Configuration Enable Atomic Set Bit 2 Register*
[0x007C]	*GPIOn_EN2_CLR*	*GPIO Port n Configuration Enable Atomic Clear Bit 2 Register*
[0x00A8]	*GPIOn_HYSEN*	*GPIO Port n Hysteresis Enable Register*
[0x00AC]	*GPIOn_SRSEL*	*GPIO Port n Slew Rate Select Register*
[0x00B0]	*GPIOn_DS0*	*GPIO Port n Output Drive Strength Bit 0 Register*
[0x00B4]	*GPIOn_DS1*	*GPIO Port n Output Drive Strength Bit 1 Register*
[0x00B8]	*GPIOn_PS*	*GPIO Port n Pulldown/Pullup Strength Select Register*
[0x00C0]	*GPIOn_VSSEL*	*GPIO Port n Voltage Select Register*

6.6.1 GPIO Register Details

Table 6-12: GPIO Port n Configuration Enable Bit 0 Register

<table>
<tr><th colspan="3">GPIO Port n Configuration Enable Bit 0</th><th colspan="2">GPIOn_EN0</th><th>[0x0000]</th></tr>
<tr><th>Bits</th><th>Field</th><th>Access</th><th>Reset</th><th colspan="2">Description</th></tr>
<tr><td>31:0</td><td>-</td><td>R/W</td><td>1</td><td colspan="2"><b>GPIO Configuration Enable Bit 0</b>
These bits, in conjunction with bits in <i>Table 6-2</i> 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 <i>GPIOn_EN0_SET</i> or <i>GPIOn_EN0_CLR</i>.
<i>Table 6-5</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></td></tr>
</table>

Table 6-13: GPIO Port n Configuration Enable Atomic Set Bit 0 Register

<table>
<tr><th colspan="3">GPIO Port n Configuration Enable Atomic Set Bit 0</th><th colspan="2">GPIOn_EN0_SET</th><th>[0x0004]</th></tr>
<tr><th>Bits</th><th>Field</th><th>Access</th><th>Reset</th><th colspan="2">Description</th></tr>
<tr><td>31:0</td><td>-</td><td>R/W1O</td><td>0</td><td colspan="2"><b>GPIO Configuration Enable Atomic Set Bit 0</b>
Writing 1 to one or more bits sets the corresponding bits in the <i>GPIOn_EN0</i> register.
0: No effect.
1: Corresponding bits in <i>GPIOn_EN0</i> register set to 1.</td></tr>
</table>

Table 6-14: GPIO Port n Configuration Enable Atomic Clear Bit 0 Register

<table>
<tr><th colspan="3">GPIO Port n Configuration Enable Atomic Clear Bit 0</th><th colspan="2">GPIOn_EN0_CLR</th><th>[0x0008]</th></tr>
<tr><th>Bits</th><th>Field</th><th>Access</th><th>Reset</th><th colspan="2">Description</th></tr>
<tr><td>31:0</td><td>-</td><td>R/W1O</td><td>0</td><td colspan="2"><b>GPIO Configuration Enable Atomic Clear Bit 0</b>
Writing 1 to one or more bits clears the corresponding bits in the <i>GPIOn_EN0</i> register.
0: No effect.
1: Corresponding bits in <i>GPIOn_EN0</i> register cleared to 0.</td></tr>
</table>

Table 6-15: GPIO Port n Output Enable Register

GPIO Port n Output Enable			GPIOn_OUTEN	[0x000C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	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_OUTEN_SET* or *GPIOn_OUTEN_CLR*. 0: Pin is set to input mode; output driver disabled. 1: Pin is set to output mode.

Table 6-16: GPIO Port n Output Enable Atomic Set Register

GPIO Port n Output Enable Atomic Set			GPIOn_OUTEN_SET	[0x0010]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	R/W1O	0	**GPIO Output Enable Atomic Set** Writing 1 to one or more bits sets the corresponding bits in the *GPIOn_OUTEN* register. 0: No effect. 1: Corresponding bits in *GPIOn_OUTEN* set to 1.

Table 6-17: GPIO Port n Output Enable Atomic Clear Register

GPIO Port n Output Enable Atomic Clear			GPIOn_OUTEN_CLR	[0x0014]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	R/W1O	0	**GPIO Output Enable Atomic Clear** Writing 1 to one or more bits sets the corresponding bits in the *GPIOn_OUTEN* register. 0: No effect. 1: Corresponding bits in *GPIOn_OUTEN* cleared to 0.

Table 6-18: GPIO Port n Output Register

GPIO Port n Output			GPIOn_OUT	[0x0018]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	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 6-19: GPIO Port n Output Atomic Set Register

GPIO Port n Output Atomic Set			GPIOn_OUT_SET	[0x001C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	R/W1O	0	**GPIO Output Atomic Set** Writing 1 to one or more bits sets the corresponding bits in the *GPIOn_OUT* register. 0: No effect. 1: Corresponding bits in *GPIOn_OUTEN* set to 1.

Table 6-20: GPIO Port n Output Atomic Clear Register

GPIO Port n Output Atomic Clear			**GPIOn_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 *GPIOn_OUT* register. 0: No effect. 1: Corresponding bits in *GPIOn_OUTEN* cleared to 0.

Table 6-21: GPIO Port n Input Register

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

Table 6-22: GPIO Port n Interrupt Mode Register

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

Table 6-23: GPIO Port n Interrupt Polarity Register

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

Table 6-24: GPIO Port n Input Enable Register

<table>
<tr><th colspan="4">GPIO Port n Input Enable</th><th>GPIOn_INEN [0x0030]</th></tr>
<tr><th>Bits</th><th>Field</th><th>Access</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:0</td><td>-</td><td>R/W</td><td>1</td><td><b>GPIO Input Enable</b>
Connects the corresponding input pad to the specified input pin for reading the pin state using the <i>GPIOn_IN</i> register.
0: Input not connected.
1: Input pin connected to the pad for reading through the <i>GPIOn_IN</i> register.</td></tr>
</table>

Table 6-25: GPIO Port n Interrupt Enable Register

<table>
<tr><th colspan="4">GPIO Port n Interrupt Enable</th><th>GPIOn_INTEN [0x0034]</th></tr>
<tr><th>Bits</th><th>Field</th><th>Access</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:0</td><td>-</td><td>R/W</td><td>0</td><td><b>GPIO Interrupt Enable</b>
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 GPIOn_INTFL_CLR register to clear pending interrupts.</i></td></tr>
</table>

Table 6-26: GPIO Port n Interrupt Enable Atomic Set Register

<table>
<tr><th colspan="4">GPIO Port Interrupt Enable Atomic Set</th><th>GPIOn_INTEN_SET [0x0038]</th></tr>
<tr><th>Bits</th><th>Field</th><th>Access</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:0</td><td>-</td><td>R/W1O</td><td>0</td><td><b>GPIO Interrupt Enable Atomic Set</b>
Writing 1 to one or more bits sets the corresponding bits in the <i>GPIOn_INTEN</i> register.
0: No effect.
1: Corresponding bits in <i>GPIOn_INTEN</i> register set to 1.</td></tr>
</table>

Table 6-27: GPIO Port n Interrupt Enable Atomic Clear Register

<table>
<tr><th colspan="4">GPIO Port Interrupt Enable Atomic Clear</th><th>GPIOn_INTEN_CLR [0x003C]</th></tr>
<tr><th>Bits</th><th>Field</th><th>Access</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:0</td><td>-</td><td>R/W1O</td><td>0</td><td><b>GPIO Interrupt Enable Atomic Clear</b>
Writing 1 to one or more bits clears the corresponding bits in the <i>GPIOn_INTEN</i> register.
0: No effect.
1: Corresponding bits in <i>GPIOn_INTEN</i> register cleared to 0.</td></tr>
</table>

Table 6-28: GPIO Port n Interrupt Status Register

<table>
<tr><th colspan="4">GPIO Port Interrupt Status</th><th>GPIOn_INTFL [0x0040]</th></tr>
<tr><th>Bits</th><th>Field</th><th>Access</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:0</td><td>-</td><td>RO</td><td>0</td><td><b>GPIO Interrupt Status</b>
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 GPIOn_INTFL_CLR register to clear the interrupt pending status flag.</i></td></tr>
</table>

Table 6-29: GPIO Port n Interrupt Clear Register

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

Table 6-30: GPIO Port n Wakeup Enable Register

GPIO Port n Wakeup Enable			GPIOn_WKEN	[0x004C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	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 6-31: GPIO Port n Wakeup Enable Atomic Set Register

GPIO Port Wakeup Enable Atomic Set			GPIOn_WKEN_SET	[0x0050]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	R/W1O	0	**GPIO Wakeup Enable Atomic Set** Writing 1 to one or more bits sets the corresponding bits in the *GPIOn_WKEN* register. 0: No effect. 1: Corresponding bits in *GPIOn_WKEN* register set to 1.

Table 6-32: GPIO Port n Wakeup Enable Atomic Clear Register

GPIO Port Wakeup Enable Atomic Clear			GPIOn_WKEN_CLR	[0x0054]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	R/W1O	0	**GPIO Wakeup Enable Atomic Clear** Writing 1 to one or more bits clears the corresponding bits in the *GPIOn_WKEN* register. 0: No effect. 1: Corresponding bits in *GPIOn_WKEN* register cleared to 0.

Table 6-33: GPIO Port n Interrupt Dual Edge Mode Register

GPIO Port n Interrupt Dual Edge Mode			GPIOn_DUALEDGE	[0x005C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	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 *GPIOn_INTMODE* bit is set to edge triggered. The associated polarity (*GPIOn_INTPOL*) setting has no effect when this bit is set. 0: Disabled 1: Enabled

Table 6-34: GPIO Port n Pad Configuration 1 Register

GPIO Port n Pad Configuration 1			GPIOn_PADCTRL0	[0x0060]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	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 *Table 6-3*.

Table 6-35: GPIO Port n Pad Configuration 2 Register

GPIO Port n Pad Configuration 2			GPIOn_PADCTRL1	[0x0064]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	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 6-3*.

Table 6-36: GPIO Port n Configuration Enable Bit 1 Register

GPIO Port n Configuration Enable Bit 1			GPIOn_EN1	[0x0068]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	R/W	0	**GPIO Configuration Enable Bit 1** These bits, in conjunction with bits in *Table 6-2* 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_EN1_SET* or *GPIOn_EN1_CLR*. *Table 6-5* depicts a detailed example of how each of these bits applies to each of the GPIO device pins *Note: Some GPIO are not implemented in all devices. The bits associated with unimplemented GPIO should not be changed from their default value.* *Note: This register setting does not affect input and interrupt functionality of the associated pin.*

Table 6-37: GPIO Port n Configuration Enable Atomic Set Bit 1 Register

GPIO Port n Configuration Enable Atomic Set Bit 1			GPIOn_EN1_SET	[0x006C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	R/W1O	0	**GPIO Configuration Enable Atomic Set Bit 1** Writing 1 to one or more bits sets the corresponding bits in the *GPIOn_EN1* register. 0: No effect. 1: Corresponding bits in *GPIOn_EN1* register set to 1.

Table 6-38: GPIO Port n Configuration Enable Atomic Clear Bit 1 Register

GPIO Port n Configuration Enable Atomic Clear Bit 1			GPIOn_EN1_CLR	[0x0070]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	R/W1O	0	**GPIO Configuration Enable Atomic Clear Bit 1** Writing 1 to one or more bits clears the corresponding bits in the *GPIOn_EN1* register. 0: No effect. 1: Corresponding bits in *GPIOn_EN1* register cleared to 0.

Table 6-39: 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	-	R/W	0	**GPIO Configuration Enable Bit 2** These bits, in conjunction with bits in These bits, in conjunction with bits in *Table 6-2* 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*. *Table 6-5* depicts a detailed example of how each of these bits applies to each of the GPIO device pins *Note: Some GPIO are not implemented in all devices. The bits associated with unimplemented GPIO should not be changed from their default value.* *Note: This register setting does not affect input and interrupt functionality of the associated pin.*

Table 6-40: 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/W1O	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 6-41: 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/W1O	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 6-42: GPIO Port n Hysteresis Enable Register

GPIO Port n Hysteresis Enable			**GPIOn_HYSEN**	**[0x00A8]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	RO	0	**Reserved**

Table 6-43: GPIO Port n Output Drive Strength Bit 0 Register

GPIO Port n Output Drive Strength Bit 0			**GPIOn_SRSEL**	**[0x00AC]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	R/W	0	**Reserved**

Table 6-44: GPIO Port n Output Drive Strength Bit 0 Register

GPIO Port n Output Drive Strength Bit 0			**GPIOn_DS0**	**[0x00B0]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	R/W	0	**GPIO Output Drive Strength Selection 0** See *Table 6-4* for details on how to set the GPIO output drive strength and other electrical characteristics.

Table 6-45: 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	-	R/W	0	**GPIO Output Drive Strength Selection 1** See *Table 6-4* for details on how to set the GPIO output drive strength and other electrical characteristics.

Table 6-46: 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	-	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. *Note: Refer to the data sheet for specific electrical characteristics of the pulldown/pullup resistances.*

Table 6-47: GPIO Port n Voltage Select Register

GPIO Port n Voltage Select			GPIOn_VSSEL	[0x00C0]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	R/W	0	**GPIO Supply Voltage Select** Selects the voltage rail used for the pin. 0: V_{DDIO} 1: V_{DDIOH}

7. Flash Controller (FLC)

The MAX78002 flash controller manages read, write, and erase accesses to the internal flash and provides the following features:

- Up to 2.5MB total internal flash memory
- 160 pages
- 16,384 bytes per page
    - 4,096 words by 128 bits per page
- 128-bit data reads and writes
- Page erase and mass erase support
- Each page supports write/erase protection and AHB read protection

7.1 Instances

The device includes one instance of the FLC. The 2.5MB of internal flash memory is programmable through the serial wire debug interface (in-system) or directly with software (in-application).

The flash is organized as an array of 4,096 words by 128 bits, or 16,384 bytes per page. *Table 7-1* shows the page start address and page end address of the internal flash memory.

Table 7-1: MAX78002 Internal Flash Memory Organization

Instance Number	Page Number	Page Size (Bytes)	Start Address	End Address
FLC0	0	16,384	0x1000 0000	0x1000 3FFF
	1	16,384	0x1000 4000	0x1000 7FFF
	2	16,384	0x1000 8000	0x1000 BFFF
	3	16,384	0x1000 C000	0x1000 FFFF
	…	…	…	…
	158	16,384	0x1027 8000	0x1027 BFFF
	159	16,384	0x1027 C000	0x1027 FFFF

7.2 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.

7.2.1 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 7-1*. If using the IPO as the system clock, the *FLC_CLKDIV*.*clkdiv* should be set to 100 (0x64).

Equation 7-1: FLC Clock Frequency

$$f_{FLCn\_CLK} = \frac{f_{SYS\_CLK}}{FLCn\_CLKDIV.clkdiv} = 1MHz$$

7.2.2 Lock Protection

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

1. The flash instance remaining locked,

   or,

2. The flash instance is 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, FLC_INTR.af, to indicate an access violation occurred.

7.2.3 Flash Write Width

The FLC supports write widths of 128-bits only. The target address bits *FLC_ADDR[3:0]* are ignored, resulting in 128-bit address alignment.

Table 7-2: Valid Addresses Flash Writes

	FLC_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	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0

7.2.4 Flash Write

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

1. If desired, enable the flash controller interrupts by setting the *FLC_INTR.afie* and *FLC_INTR.doneie* bits.
2. Read the *FLC_CTRL.pend* bit until it returns 0.
3. Configure the *FLC_CLKDIV.clkdiv* field to achieve a 1MHz frequency based on the selected SYS_CLK frequency.
4. Set the *FLC_ADDR* register to a valid target page address offset. See *Table 7-2* for details.
5. Set *FLC_DATA[3]*, *FLC_DATA[2]*, *FLC_DATA[1]*, and *FLC_DATA[0]* to the data to write.
   a. *FLC_DATA[3]* is the most significant word, and *FLC_DATA[0]* is the least significant word.
      i. 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 the *FLC_CTRL.unlock* field to 2 to unlock the flash.
7. Set the *FLC_CTRL.wr* field to 1.
   a. The hardware automatically clears this field when the write operation is complete.
8. The *FLC_INTR.done* field is set to 1 by hardware when the write completes.
   a. An interrupt is generated if the *FLC_INTR.doneie* field is set to 1.
9. If an error occurred, the *FLC_INTR.af* field is set to 1 by hardware. An interrupt is generated if the *FLC_INTR.afie* field is set to 1.
10. Set the *FLC_CTRL.unlock* field to any value other than 2 to re-lock the flash.

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

Note: If the ICC is enabled, either disable the ICC before writing to the flash or flush the ICC after writing to the flash.

7.2.5 Page Erase

CAUTION: Care must be taken not to erase the page from which the application software 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 *FLC_INTR.afie* and *FLC_INTR.doneie bits*.
2. Read the *FLC_CTRL.pend* bit until it returns 0.
3. Configure *FLC_CLKDIV.clkdiv* to match the SYS_CLK frequency.
4. Set the *FLC_ADDR* register to an address offset within the target page to be erased. *FLC_ADDR[14:0]* is ignored by the FLC to ensure the address is page-aligned.
5. Set *FLC_CTRL.unlock* to 2 to unlock the flash instance.
6. Set *FLC_CTRL.erase_code* to 0x55 for page erase.
7. Set *FLC_CTRL.pge* to 1 to start the page erase operation.
8. The *FLC_CTRL.pend* bit is set by the flash controller while the page erase is in progress, and the *FLC_CTRL.pge* and *FLC_CTRL*.pend are cleared by the flash controller when the page erase is complete.
9. *FLC_INTR.done* is set by hardware when the page erase completes, and if an error occurred, the *FLC_INTR.af* flag is set. These bits generate a flash interrupt if the interrupt enable bits are set.
10. Set *FLC_CTRL.unlock* to any value other than 2 to re-lock the flash instance.

7.2.6 Mass Erase

CAUTION: Care must be taken not to erase the flash from which application software 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 *FLC_CTRL.pend* bit until it returns 0.
2. Configure *FLC_CLKDIV.clkdiv* to match the SYS_CLK frequency.
3. Set *FLC_CTRL.unlock* to 2 to unlock the internal flash.
4. Set *FLC_CTRL.erase_code* to 0xAA for mass erase.
5. Set *FLC_CTRL.me* to 1 to start the mass erase operation.
6. The *FLC_CTRL.pend* bit is set by the flash controller while the mass erase is in progress, and the *FLC_CTRL.me* and *FLC_CTRL.pend* are cleared by the flash controller when the mass erase is complete.
7. *FLC_INTR.done* is set by the flash controller when the mass erase completes, and if an error occurred, the *FLC_INTR.af* flag is set. These bits generate a flash interrupt if the interrupt enable bits are set.
8. Set *FLC_CTRL.unlock* to any value other than 2 to re-lock the flash instance.

7.3 Registers

See *Table 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 7-3: Flash Controller Register Summary

Offset	Register Name	Access	Description
[0x0000]	*FLC_ADDR*	R/W	Flash Controller Address Pointer Register
[0x0004]	*FLC_CLKDIV*	R/W	Flash Controller Clock Divisor Register
[0x0008]	*FLC_CTRL*	R/W	Flash Controller Control Register
[0x0024]	*FLC_INTR*	R/W	Flash Controller Interrupt Register
[0x0030]	*FLC_DATA[0]*	R/W	Flash Controller Data Register 0
[0x0034]	*FLC_DATA[1]*	R/W	Flash Controller Data Register 1
[0x0038]	*FLC_DATA[2]*	R/W	Flash Controller Data Register 2
[0x003C]	*FLC_DATA[3]*	R/W	Flash Controller Data Register 3
[0x0040]	*FLC_ACTRL*	R/W	Flash Controller Access Control Register
[0x0080]	*FLC_WELR0*	R/W	Flash Write/Erase Lock 0 Register
[0x0088]	*FLC_WELR1*	R/W	Flash Write/Erase Lock 1 Register
[0x0090]	*FLC_WELR2*	R/W	Flash Write/Erase Lock 2 Register
[0x0098]	*FLC_WELR3*	R/W	Flash Write/Erase Lock 3 Register
[0x00A0]	*FLC_WELR4*	R/W	Flash Write/Erase Lock 4 Register
[0x0084]	*FLC_RLR0*	R/W	Flash Read Lock 0 Register
[0x008C]	*FLC_RLR1*	R/W	Flash Read Lock 1 Register
[0x0094]	*FLC_RLR2*	R/W	Flash Read Lock 2 Register
[0x009C]	*FLC_RLR3*	R/W	Flash Read Lock 3 Register
[0x00A4]	*FLC_RLR4*	R/W	Flash Read Lock 4 Register

7.3.1 Register Details

Table 7-4: Flash Controller Address Pointer Register

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

Table 7-5: Flash Controller Clock Divisor Register

Flash Controller Clock Divisor		**FLC_CLKDIV**		**[0x0004]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:8	-	RO	-	**Reserved**

Flash Controller Clock Divisor			FLC_CLKDIV	[0x0004]
Bits	**Name**	**Access**	**Reset**	**Description**
7:0	clkdiv	R/W	0x76	**Flash Controller Clock Divisor** The APB clock is divided by the value in this field to generate the FLCn peripheral clock, f_{FLC_CLK}. The FLC peripheral clock must equal 1MHz. The default on POR, system reset, and watchdog reset is 120, resulting in f_{FLC_CLK} = 1MHz when IPO is the system oscillator. The FLC peripheral clock is only used during erase and program functions and not during read functions. See *Clock Configuration* for additional details.

Table 7-6: Flash Controller Control Register

Flash Controller Control			FLC_CTRL	[0x0008]
Bits	**Name**	**Access**	**Reset**	**Description**
31:28	unlock	R/W	0	**Flash Unlock** Write the unlock code, 2, before 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**
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 *FLC_INTR* register, are ignored by the flash controller. This bit is cleared by hardware once the flash becomes accessible. *Note: If the flash controller is busy (FLC_CTRL.pend = 1), reads, writes, and erase operations are not allowed and result in an access failure (FLC_INTR.af = 1).* 0: Flash idle 1: Flash busy
23:16	-	RO	0	**Reserved**
15:8	erase_code	R/W	0	**Erase Code** Before 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 before setting the erase code. This field is automatically cleared after the erase operation is complete. 0x00: Erase disabled. 0x55: Page erase code. 0xAA: Mass erase code.
7:3	-	RO	0	**Reserved**
2	pge	R/W1O	0	**Page Erase** Write a 1 to this field to initiate a page erase at the address in *FLC_ADDR.addr*. The flash must be unlocked before attempting a page erase. See *FLC_CTRL.unlock* for details. The flash controller hardware clears this bit when a page erase operation is complete. 0: Normal operation 1: Write a 1 to initiate a page erase. If this field reads 1, a page erase operation is in progress.

Flash Controller Control			FLC_CTRL	[0x0008]
Bits	**Name**	**Access**	**Reset**	**Description**
1	me	R/W1O	0	**Mass Erase** Write a 1 to this field to initiate a mass erase of the internal flash memory. The flash must be unlocked before attempting a mass erase. See *FLC_CTRL*.*unlock* for details. The flash controller hardware clears this bit when the mass erase operation completes. 0: Normal operation 1: Initiate mass erase
0	wr	R/W1O	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 writes to the address set in the *FLC_ADDR* register. 0: Normal operation 1: Write 1 to initiate a write operation. If this field reads 1, a write operation is in progress. *Note: This field is protected and cannot be set to 0 by application software.*

Table 7-7: Flash Controller Interrupt Register

Flash Controller Interrupt			FLC_INTR	[0x0024]
Bits	**Name**	**Access**	**Reset**	**Description**
31:10	-	RO	0	**Reserved**
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
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**
1	af	R/W0C	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/W0C	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 7-8: Flash Controller Data 0 Register

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

Table 7-9: Flash Controller Data Register 1

Flash Controller Data 1		FLC_DATA[1]		[0x0034]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	data	R/W	0	**Flash Data 1** Flash data for bits 63:32.

Table 7-10: Flash Controller Data Register 2

Flash Controller Data 2		FLC_DATA[2]		[0x0038]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	data	R/W	0	**Flash Data 2** Flash data for bits 95:64.

Table 7-11: Flash Controller Data Register 3

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

Table 7-12: Flash Controller Access Control Register

Flash Controller Access Control		FLC_ACTRL		[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 7-13: Flash Write/Lock 0 Register

Flash Write/Lock 0		FLC_WELR0		[0x0080]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	welr0	R/W1C	0xFFFF FFFF	**Flash Write/Lock Bit** Each bit in this register maps to a page of the internal flash. *FLC_WELR0*[0] maps to page 0 of the flash, and *FLC_WELR0*[31] maps to page 31. Each flash page is 16,384 bytes. Write a 1 to clear 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 write protected. 1: The corresponding page of flash is *not* write protected.

Table 7-14: Flash Write/Lock 1 Register

Flash Write/Lock 1			FLC_WELR1	[0x0088]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	welr1	R/W1C	0xFFFF FFFF	**Flash Write/Lock Bit** Each bit in this register maps to a page of the internal flash. *FLC_WELR1*[0] maps to page 32 of the flash, and *FLC_WELR1*[31] maps to page 63 of flash. Each flash page is 16,384 bytes. Write a 1 to clear 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 write protected. 1: The corresponding flash page is *not* write protected.

Table 7-15: Flash Write/Lock 2 Register

Flash Write/Lock 2			FLC_WELR2	[0x0090]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	welr2	R/W1C	0xFFFF FFFF	**Flash Write/Lock Bit** Each bit in this register maps to a page of the internal flash. *FLC_WELR2*[0] maps to page 64 of the flash, and *FLC_WELR2*[31] maps to page 95 of flash. Each flash page is 16,384 bytes. Write a 1 to clear 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 write protected. 1: The corresponding flash page is *not* write protected.

Table 7-16: Flash Write/Lock 3 Register

Flash Write/Lock 3			FLC_WELR3	[0x0098]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	welr3	R/W1C	0xFFFF FFFF	**Flash Write/Lock Bit** Each bit in this register maps to a page of the internal flash. *FLC_WELR3*[0] maps to page 96 of the flash, and *FLC_WELR3*[31] maps to page 127 of flash. Each flash page is 16,384 bytes. Write a 1 to clear 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 write protected. 1: The corresponding flash page is *not* write protected.

Table 7-17: Flash Write/Lock 4 Register

Flash Write/Lock 4			FLC_WELR4	[0x00A0]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	welr4	R/W1C	0xFFFF FFFF	**Flash Write/Lock Bit** Each bit in this register maps to a page of the internal flash. *FLC_WELR4*[0] maps to page 128 of the flash, and *FLC_WELR4*[31] maps to page 159 of flash. Each flash page is 16,384 bytes. Write a 1 to clear 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 write protected. 1: The corresponding flash page is *not* write protected.

Table 7-18: Flash Read Lock 0 Register

Flash Read Lock 0			FLC_RLR0	[0x0084]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	rlr0	R/W1C	0xFFFF FFFF	**Read Lock Bit** Each bit in this register maps to a page of the internal flash. *FLC_RLR0*[0] maps to page 0 of the flash, and *FLC_RLR0*[31] maps to page 31 of flash. Each flash page is 16,384 bytes. Write a 1 to clear 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 read protected. 1: The corresponding flash page is *not* read protected.

Table 7-19: Flash Read Lock 1 Register

Flash Read Lock 1			FLC_RLR1	[0x008C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	rlr1	R/W1C	0xFFFF FFFF	**Read Lock Bit** Each bit in this register maps to a page of the internal flash. *FLC_RLR1*[0] maps to page 32 of the flash, and *FLC_RLR1*[31] maps to page 63 of flash. Each flash page is 16,384 bytes. Write a 1 to clear 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 read protected. 1: The corresponding flash page is *not* read protected.

Table 7-20: Flash Read Lock 2 Register

Flash Read Lock 2			FLC_RLR2	[0x0094]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	rlr2	R/W1C	0xFFFF FFFF	**Read Lock Bit** Each bit in this register maps to a page of the internal flash. *FLC_RLR2*[0] maps to page 64 of the flash, and *FLC_RLR2*[31] maps to page 95 of flash. Each flash page is 16,384 bytes. Write a 1 to clear 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 read protected. 1: The corresponding flash page is *not* read protected.

Table 7-21: Flash Read Lock 3 Register

Flash Read Lock 3			FLC_RLR3	[0x009C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	rlr3	R/W1C	0xFFFF FFFF	**Read Lock Bit** Each bit in this register maps to a page of the internal flash. *FLC_RLR3*[0] maps to page 96 of the flash, and *FLC_RLR3*[31] maps to page 127 of flash. Each flash page is 16,384 bytes. Write a 1 to clear 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 read protected. 1: The corresponding flash page is *not* read protected.

Table 7-22: Flash Read Lock 4 Register

Flash Read Lock 3		**FLC_RLR4**		**[0x00A4]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	rlr4	R/W1C	0xFFFF FFFF	**Read Lock Bit** Each bit in this register maps to a page of the internal flash. *FLC_RLR4*[0] maps to page 128 of the flash, and *FLC_RLR4*[31] maps to page 159 of flash. Each flash page is 16,384 bytes. Write a 1 to clear 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 read protected. 1: The corresponding flash page is *not* read protected.

8. Debug Access Port (DAP)

Some device versions might provide an Arm debug access port (DAP) which supports debugging during application development. Refer to the device data sheet's ordering information table to determine if a specific part number supports a customer-accessible DAP. *GCR_SYSST.icelock* = 0 if the device provides a customer-accessible DAP.

8.1 Instances

The DAP interface communicates through the serial wire debug (SWD) interface signals shown in *Table 8-1*.

Table 8-1: MAX78002 DAP Instances

Instance	Pin	Alternate Function	SWD Signal
0	P0.28	AF1	SWDIO
	P0.29	AF1	SWDCLK

8.2 Access Control

8.2.1 Factory Disabled DAP

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

8.2.2 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 the *GCR_SYSCTRL.swd_dis* field to 1. The read-only field, *GCR_SYSST.icelock*, is cleared to 0 by hardware, and the software has read and write access to the *GCR_SYSCTRL.swd_dis* field. The *GCR_SYSCTRL.swd_dis* field resets to 0 after every POR to allow access to the DAP during development.

The software can disable the DAP by setting the *GCR_SYSCTRL.swd_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.

8.3 Pin Configuration

Instances of SWD signals in the GPIO and Alternate Function matrices determine which GPIO pins are associated with a signal. It is unnecessary to configure a pin for an alternate function to use the DAP following a POR. By default, the pin associated with the bidirectional SWDIO signal is configured as SWDIO with high-impedance input after a POR.

9. 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, 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 that enable communication between the RV32 and CM4 cores. Additionally, either core can generate a semaphore interrupt for either the CM4 or the RV32 providing immediate notification of communication through the mailbox registers.

9.1 Instances

There is one instance of the semaphore peripheral, shown in *Table 9-1*.

Table 9-1: MAX78002 Semaphore Instances

Instance	Number of Semaphores
SEMA	8

9.2 Multiprocessor Communications

The semaphore includes support for multicore communications through 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 registers. Application firmware should manage how these registers are used to prevent collisions if both cores attempt to modify the registers at the same time.

9.2.1 Reset

Globally reset the semaphore peripheral by setting *GCR_RST1.smphr* to 1.

9.2.2 CM4 Semaphore Interrupt Generation

The *SEMA_IRQ0* register can 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 interrupt. The CM4 interrupt handler should write the *SEMA_IRQ0.en and/or the SEMA_IRQ0.cm4_irq* field(s) to 0 to clear the interrupt condition.

9.2.3 RV32 Semaphore Interrupt Generation

The *SEMA_IRQ1* register can 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 interrupt. The RV32 interrupt handler should write the *SEMA_IRQ1.en and/or the SEMA_IRQ1.rv32_irq* field(s) to 0 to clear the interrupt condition.

9.3 Registers

See *Table 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 9-2: Semaphore Register Summary

Offset	Register	Name
[0x0000]	*SEMA_SEMAPHORES[0]*	Semaphore 0 Register
[0x0004]	*SEMA_SEMAPHORES[1]*	Semaphore 1 Register
[0x0008]	*SEMA_SEMAPHORES[2]*	Semaphore 2 Register
[0x000C]	*SEMA_SEMAPHORES[3]*	Semaphore 3 Register
[0x0010]	*SEMA_SEMAPHORES[4]*	Semaphore 4 Register
[0x0014]	*SEMA_SEMAPHORES[5]*	Semaphore 5 Register
[0x0018]	*SEMA_SEMAPHORES[6]*	Semaphore 6 Register
[0x0020]	*SEMA_SEMAPHORES[7]*	Semaphore 7 Register
[0x0040]	*SEMA_IRQ0*	Semaphore Interrupt 0 Register
[0x0044]	*SEMA_MAIL0*	Semaphore Mailbox 0 Register
[0x0048]	*SEMA_IRQ1*	Semaphore Interrupt 1 Register
[0x004C]	*SEMA_MAIL1*	Semaphore Mailbox 1 Register
[0x0100]	*SEMA_STATUS*	Semaphore Status Register

9.3.1 Register Details

Table 9-3: Semaphore 0 Register

Semaphore 0			**SEMA_SEMAPHORES[0]**	**[0x0000]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
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 9-4: Semaphore 1 Register

Semaphore 1			**SEMA_SEMAPHORES[1]**	**[0x0004]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
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 9-5: Semaphore 2 Register

Semaphore 2		**SEMA_SEMAPHORES[2]**		**[0x0008]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
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 9-6: Semaphore 3 Register

Semaphore 3		**SEMA_SEMAPHORES[3]**		**[0x000C]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
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 9-7: Semaphore 4 Register

Semaphore 4		**SEMA_SEMAPHORES[4]**		**[0x0010]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
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 9-8: Semaphore 5 Register

Semaphore 5		**SEMA_SEMAPHORES[5]**		**[0x0014]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
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 9-9: Semaphore 6 Register

Semaphore 6		**SEMA_SEMAPHORES[6]**		**[0x0018]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Semaphore 6			SEMA_SEMAPHORES[6]	[0x0018]
Bits	**Field**	**Access**	**Reset**	**Description**
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 9-10: Semaphore 7 Register

Semaphore 7			SEMA_SEMAPHORES[7]	[0x001C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
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 9-11: Semaphore Interrupt 0 Register

Semaphore Interrupt 0			SEMA_IRQ0	[0x0040]
Bits	**Field**	**Access**	**Reset**	**Description**
31:17	-	RO	0	**Reserved**
16	cm4_irq	R/W	0	**CM4 Interrupt** The RV32 can use this bit to communicate with the CM4 through the semaphore interrupt. The RV32 generates a semaphore interrupt for the CM4 by setting this field to 1 and also setting the *SEMA_IRQ0.en* bit to 1.
15:1	-	RO	0	**Reserved**
0	en	R/W	0	**Interrupt Enable** Set this field to enable interrupt generation on semaphore events. 0: Disabled 1: Enabled

Table 9-12: 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 interrupt. Alternately, the CM4 can write to this register and then notify the RV32 using the *SEMA_IRQ1* register to generate an RV32 semaphore interrupt event. *Note: The management of the SEMA_MAIL0 and SEMA_MAIL1 registers is left to the software. 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.*

Table 9-13: Semaphore Interrupt 1 Register

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

Table 9-14: 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 interrupt. Alternately, the CM4 can write to this register and then notify the RV32 using the *SEMA_IRQ1* register to generate an RV32 semaphore interrupt event. *Note: The management of the SEMA_MAIL0 and SEMA_MAIL1 registers is left to the software. 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.*

Table 9-15: Semaphore Status Register

Semaphore Status		**SEMA_STATUS**		**[0x0100]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
7	status7	R	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: *SEMA_SEMAPHORES[7].status* is 0. 1: *SEMA_SEMAPHORES[7].status* is 1.
6	status6	R	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: *SEMA_SEMAPHORES[6].status* is 0. 1: *SEMA_SEMAPHORES[6].status* is 1.

Semaphore Status			SEMA_STATUS	[0x0100]
Bits	**Field**	**Access**	**Reset**	**Description**
5	status5	R	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: *SEMA_SEMAPHORES[5].status* is 0. 1: *SEMA_SEMAPHORES[5].status* is 1.
4	status4	R	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: *SEMA_SEMAPHORES[4].status* is 0. 1: *SEMA_SEMAPHORES[4].status* is 1.
3	status3	R	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: *SEMA_SEMAPHORES[3].status* is 0. 1: *SEMA_SEMAPHORES[3].status* is 1.
2	status2	R	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: *SEMA_SEMAPHORES[2].status* is 0. 1: *SEMA_SEMAPHORES[2].status* is 1.
1	status1	R	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: *SEMA_SEMAPHORES[1].status* is 0. 1: *SEMA_SEMAPHORES[1].status* is 1.
0	status0	R	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: *SEMA_SEMAPHORES[0].status* is 0. 1: *SEMA_SEMAPHORES[0].status* is 1.

10. Standard DMA (DMA)

The DMA is a peripheral that provides the ability to perform high-speed, block memory transfers of data independent of a CPU. All DMA transactions consist of a 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 RAM address,
- from a RAM address to a transmit FIFO, or
- from a source RAM address to a destination RAM address.

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

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

10.1 Instances

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

Table 10-1: MAX78002 DMA and Channel Instances

DMA Instance	DMA_CH*n* Channel Instance
DMA	DMA_CH0
	DMA_CH1
	DMA_CH2
	DMA_CH3

10.2 DMA Channel Operation (DMA_CH)

10.2.1 DMA Channel Arbitration and DMA Bursts

DMA 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, it 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 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 continues until complete.

Only an error condition can interrupt an ongoing data transfer.

10.2.2 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 10-2*. The *DMA_CHn_SRC* and *DMA_CHn_DST* registers hold the source and 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 10-2: MAX78002 DMA Source and Destination by Peripheral

DMA_CHn_CTRL.request	**Peripheral**	**DMA Source**	**DMA Destination**
0x00	Memory-to-Memory	*DMA_CHn_SRC*	*DMA_CHn_DST*
0x01	SPI1	SPI1 Receive FIFO	*DMA_CHn_DST*
0x02:0x03	Reserved		
0x04	UART0	UART0 Receive FIFO	*DMA_CHn_DST*
0x05	UART1	UART1 Receive FIFO	*DMA_CHn_DST*
0x06	Reserved		
0x07	I2C0	I2C0 Receive FIFO	*DMA_CHn_DST*
0x08	I2C1	I2C1 Receive FIFO	*DMA_CHn_DST*
0x09	ADC	ADC FIFO	*DMA_CHn_DST*
0x0A	I2C2	I2C2 Receive FIFO	*DMA_CHn_DST*
0x0B:0x0D	Reserved		
0x0E	UART2	UART2 Receive FIFO	*DMA_CHn_DST*
0x0F	SPI0	SPI0 Receive FIFO	*DMA_CHn_DST*
0x10	AES	AES Receive	*DMA_CHn_DST*
0x11:0x1D	Reserved		
0x1E	I^2S	I^2S Receive	*DMA_CHn_DST*
0x1F:0x20	Reserved		
0x21	SPI1	*DMA_CHn_SRC*	SPI1 Transmit FIFO
0x22:0x23	Reserved		
0x24	UART0	*DMA_CHn_SRC*	UART0 Transmit FIFO
0x25	UART1	*DMA_CHn_SRC*	UART1 Transmit FIFO
0x26	Reserved		
0x27	I2C0	*DMA_CHn_SRC*	I2C0 Transmit FIFO
0x28	I2C1	*DMA_CHn_SRC*	I2C1 Transmit FIFO
0x29	Reserved		
0x2A	I2C2	*DMA_CHn_SRC*	I2C2 Transmit FIFO
0x2B	Reserved		
0x2C	CRC	*DMA_CHn_SRC*	CRC
0x2D	Reserved		
0x2E	UART2	*DMA_CHn_SRC*	UART2 Transmit FIFO
0x2F	SPI0	*DMA_CHn_SRC*	SPI0 Transmit FIFO
0x30	AES	*DMA_CHn_SRC*	AES
0x31:0x3D	Reserved		
0x3E	I^2S	*DMA_CHn_SRC*	I^2S Transmit FIFO
0x3F	Reserved		

10.2.3 Data Movement from Source to DMA

Table 10-3 shows the fields that control the burst movement of data into the DMA FIFO. The source is a peripheral or memory.

Table 10-3: Data Movement from Source to DMA FIFO

Register/Field	Description	Comments
DMA_CHn_SRC	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.
DMA_CHn_CNT	Number of bytes to transfer before a CTZ condition occurs	This register is decremented on each read of the burst.
DMA_CHn_CTRL.burst_size	Burst size (1-32)	This maximum number of bytes moved during the burst read.
DMA_CHn_CTRL.srcwd	Source width	This field 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 *DMA_CHn_CNT* is not great enough to supply all the needed bytes.
DMA_CHn_CTRL.srcinc	Source increment enable	Increments *DMA_CHn_SRC*. This field is ignored when the DMA source is a peripheral.

10.2.4 Data Movement from DMA to Destination

Table 10-4 shows the fields that control the burst movement of data out of the DMA FIFO. The destination is a peripheral or memory.

Table 10-4: Data Movement from the DMA FIFO to Destination

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

10.3 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 DMA transfer destination, configure the *DMA_CHn_DST* register to the destination memory's starting address.
3. If using memory for the DMA transfer source, configure the *DMA_CHn_SRC* register 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.srcinc* 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_CNTRLD* 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.
7. Set *DMA_CHn_CTRL.en* to 1 to start the DMA transfer immediately.
8. Wait for the interrupt flag to become 1 to indicate the completion of the DMA transfer.

10.4 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, two possible responses are possible depending on the value of the *DMA_CHn_CTRL.rlden* field:

- If *DMA_CHn_CTRL.rlden* = 1
  - 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.
- If *DMA_CHn_CTRL.rlden* = 0
  - The channel is disabled, and *DMA_CHn_STATUS.status* is cleared.

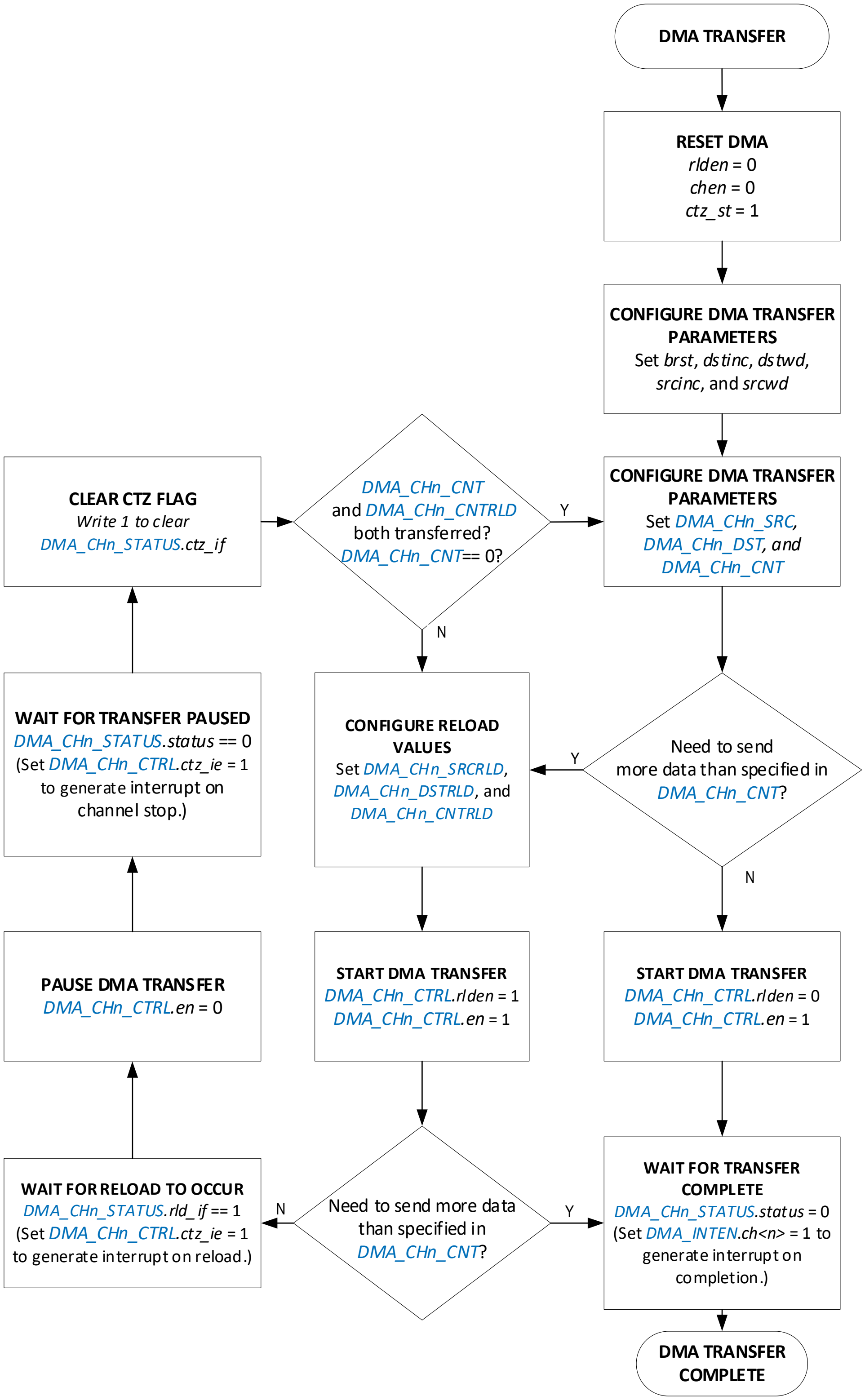
10.5 Chaining Buffers

Chaining buffers reduces the DMA interrupt response time and allows the DMA to service requests without intermediate processing from the CPU. *Figure 10-1* 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_CTRL*
  - *DMA_CHn_SRC*
  - *DMA_CHn_DST*
  - *DMA_CHn_CNT*
  - *DMA_CHn_SRCRLD*
  - *DMA_CHn_DSTRLD*
  - *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 or 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 *DMA_INTEN.ch<n>* bit. Then, poll the *DMA_CHn_STATUS.status* bit to verify that the channel is disabled.

Figure 10-1: DMA Block-Chaining Flowchart


DMA TRANSFER
RESET DMA
rlden = 0
chen = 0
ctz_st = 1
CONFIGURE DMA TRANSFER PARAMETERS
Set brst, dstinc, dstwd, srcinc, and srcwd
CONFIGURE DMA TRANSFER PARAMETERS
Set DMA_CHn_SRC, DMA_CHn_DST, and DMA_CHn_CNT
DMA_CHn_CNT and DMA_CHn_CNTRLD both transferred? DMA_CHn_CNT== 0?
Y
N
CLEAR CTZ FLAG
Write 1 to clear DMA_CHn_STATUS.ctz_if
Need to send more data than specified in DMA_CHn_CNT?
Y
N
CONFIGURE RELOAD VALUES
Set DMA_CHn_SRCRLD, DMA_CHn_DSTRLD, and DMA_CHn_CNTRLD
WAIT FOR TRANSFER PAUSED
DMA_CHn_STATUS.status == 0
(Set DMA_CHn_CTRL.ctz_ie = 1 to generate interrupt on channel stop.)
PAUSE DMA TRANSFER
DMA_CHn_CTRL.en = 0
START DMA TRANSFER
DMA_CHn_CTRL.rlden = 1
DMA_CHn_CTRL.en = 1
START DMA TRANSFER
DMA_CHn_CTRL.rlden = 0
DMA_CHn_CTRL.en = 1
WAIT FOR RELOAD TO OCCUR
DMA_CHn_STATUS.rld_if == 1
(Set DMA_CHn_CTRL.ctz_ie = 1 to generate interrupt on reload.)
N
Need to send more data than specified in DMA_CHn_CNT?
Y
WAIT FOR TRANSFER COMPLETE
DMA_CHn_STATUS.status = 0
(Set DMA_INTEN.ch<n> = 1 to generate interrupt on completion.)
DMA TRANSFER COMPLETE

10.6 DMA Interrupts

Enable interrupts for each channel by setting *DMA_INTEN.ch<n>*. When an interrupt for a channel is pending, the corresponding *DMA_INTFL.ch<n>* = 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).

10.7 Channel Timeout Detect

Each channel can optionally generate an interrupt when the 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 10-5*. 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 10-5: DMA Channel Timeout Configuration

DMA_CHn_CTRL.to_clkdiv	**Timeout Period (µs)**
0	Channel timeout disabled
1	$\frac{2^{8} * [Value\ from\ \mathrm{DMA\_CHn\_CTRL.to\_per}]}{f_{\mathrm{HCLK}}}$
2	$\frac{2^{16} * [Value\ from\ \mathrm{DMA\_CHn\_CTRL}.tosel]}{f_{\mathrm{HCLK}}}$
3	$\frac{2^{24} * [Value\ from\ \mathrm{DMA\_CHn\_CTRL}.tosel]}{f_{\mathrm{HCLK}}}$

The start of the timeout period is controlled by the *DMA_CHn_CTRL.to_wait* field as follows:

- If *DMA_CHn_CTRL.to_wait* = 0, the timer begins counting immediately after the *DMA_CHn_CTRL.to_clkdiv* field is configured to a value other than 0.
- 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, the 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.

10.8 Memory-to-Memory DMA

Memory-to-memory transfers are processed as if the request is permanently active. 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.

10.9 DMA Registers

See *Table* for this peripheral/module's base address. See *Table 1-1* 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 10-6: DMA Register Summary

Offset	Register	Description
[0x0000]	*DMA_INTEN*	DMA Interrupt Enable register
[0x0004]	*DMA_INTFL*	DMA Interrupt Flag register

10.9.1 Register Details

Table 10-7: DMA Interrupt Enable Register

DMA Interrupt Enable		DMA_INTEN		[0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	*ch<n>*	R/W	0	**DMA Channel *n* Interrupt Enable** Each bit in this field enables the corresponding channel interrupt *n* in *DMA_INTFL*. Register bits associated with unimplemented channels should not be changed from their default reset value. 0: Disabled 1: Enabled

Table 10-8: DMA Interrupt Flag Register

DMA Interrupt Flag		DMA_INTFL		[0x0004]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	*ch<n>*	RO	0	**DMA Channel *n* Interrupt Flag** Each bit in this field represents an interrupt for the corresponding channel interrupt m. To clear an interrupt, clear the corresponding active interrupt bit in the *DMA_CHn_STATUS* register. An interrupt bit in this field is only set if the corresponding interrupt enable field is set in the *DMA_INTEN* register. Register bits associated with unimplemented channels should be ignored. 0: Normal operation 1: Interrupt pending

10.10 DMA Channel Register Summary

Table 10-9: Standard DMA Channel 0 to Channel 7 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

10.11 DMA Channel Registers

See *Table* for this peripheral/module's base address. If multiple instances of the peripheral are provided, each instance has its own independent set of the registers shown in *Table 10-10*. Register names for a specific instance are defined by replacing "n" with the instance number. For example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See *Table 1-1* 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 10-10: DMA Channel Registers Summary

Offset	Register	Description
[0x0000]	*DMA_CHn_CTRL*	DMA Channel *n* Control 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_CNTRLD*	DMA Channel *n* Count Reload Register

10.11.1 Register Details

Table 10-11: DMA Channel 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. *DMA_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. *DMA_INTFL.ch<n>_ipend* bit is set to 1 whenever *DMA_CHn_STATUS.status* changes from 1 to 0.
29	-	RO	0	**Reserved**

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. 0: 1 byte 1: 2 bytes 2: 3 bytes ... 31: 32 bytes
23	-	RO	0	**Reserved**
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** This field selects the width of each AHB transaction to the destination peripheral or memory. The actual width can be less than this field's setting if fewer bytes are in the DMA FIFO than this field's selection. 0: 1 byte 1: 2 bytes 2: 4 bytes 3: Reserved
19	-	RO	0	**Reserved**
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** This field selects the width of each AHB transaction from the source peripheral or memory. The actual width can be less than this field's setting if the *DMA_CHn_CNT* register indicates a smaller value than the width setting. 0: 1 byte 1: 2 bytes 2: 4 bytes 3: Reserved
15:14	to_clkdiv	R/W	0	**Timeout Timer Clock Pre-Scale Select** This field selects the pre-scale divider for the timeout clock input. 0: Timeout timer disabled. 1: $\frac{f_{HClK}}{28}$ 2: $\frac{f_{HCLK}}{216}$f 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** This field 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 0: 3 - 4 1: 7 - 8 2: 15 - 16 3: 31 - 32 4: 63 - 64 5: 127 - 128 6: 255 - 256 7: 511 - 512
10	to_wait	R/W	0	**Request DMA Timeout Timer Wait Enable** 0: Start timer immediately when enabled. 1: Delay the timer's 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 10-2*.
3:2	pri	R/W	0	**Channel Priority** This field sets the priority of the channel relative to other DMA channels. Channels set to the same priority are serviced in a round-robin fashion. 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 with their corresponding reload registers upon CTZ. *Note: This bit is also writeable in the DMA_CHn_CNTRLD register.*
0	en	R/W	0	**Channel Enable** This bit is automatically cleared when *DMA_CHn_STATUS.status* changes from 1 to 0. 0: Disabled 1: Enabled

Table 10-12: DMA Status Register

DMA Channel *n* Status			**DMA_CHn_STATUS**	**[0x0104]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:7	-	DNM	0	**Reserved, Do Not Modify**
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**

DMA Channel *n* Status			DMA_CHn_STATUS	[0x0104]
Bits	**Field**	**Access**	**Reset**	**Description**
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 channel's configuration, address, and count registers. Whenever this bit is cleared by hardware, the *DMA_CHn_CTRL.en* bit is also cleared. 0: Channel configuration can be changed 1: Channel busy

Table 10-13: 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 Address** This field is the source RAM address for memory-to-peripheral and memory-to-memory transfers. This field is ignored for peripheral-to-memory transfers. 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 selected using *DMA_CHn_CTRL.srcwd*. 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 10-14: 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** This field is the destination RAM address for peripheral-to-memory and memory-to-memory transfers. This field is ignored for memory-to-peripheral transfers. If *DMA_CHn_CTRL.dstinc* = 1, then this field is incremented on every AHB transfer cycle by one, two, or four bytes depending on the data width selected using *DMA_CHn_CTRL.dstwd*. 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 10-15: DMA Channel n Count Register

DMA Channel *n* Count		**DMA_CHn_CNT**		**[0x0110]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:24	-	RO	0	**Reserved**
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_CNTRLD* register.

Table 10-16: DMA Channel n Source Reload Register

DMA Channel *n* Source Reload		**DMA_CHn_SRCRLD**		**[0x0114]**
Bits	**Field**	**Access**	**Reset**	**Description**
31	-	RO	0	**Reserved**
30:0	addr	R/W	0	**Source Address Reload Value** If *DMA_CHn_CTRL.rlden* = 1, then this register's value is loaded into *DMA_CHn_SRC* upon a CTZ condition.

Table 10-17: DMA Channel n Destination Reload Register

DMA Channel *n* Destination Reload		**DMA_CHn_DSTRLD**		**[0x0118]**
Bits	**Field**	**Access**	**Reset**	**Description**
31	-	RO	0	**Reserved**
30:0	addr	R/W	0	**Destination Address Reload Value** If *DMA_CHn_CTRL.rlden* = 1, then this register's value is loaded into *DMA_CHn_DST* upon a CTZ condition.

Table 10-18: DMA Channel n Count Reload Register

DMA Channel *n* Count Reload			**DMA_CHn_CNTRLD**	**[0x011C]**
Bits	**Field**	**Access**	**Reset**	**Description**
31	ren	R/W	0	**Reload Enable.** Enables automatic loading of the *DMA_CHn_SRC*, *DMA_CHn_DST*, and *DMA_CHn_CNT* registers when a CTZ event occurs. Set this bit after the address reload registers are programmed. *Note: This bit is automatically cleared to 0 when reload occurs.* *Note: This bit is also seen in the DMA_CHn_CTRL register.* 0: Reload disabled 1: Reload enabled
30:24	-	RO	0	**Reserved**
23:0	cnt	R/W	0	**Count Reload Value.** If *DMA_CHn_CNTRLD.en* = 1, then this register's value is loaded into *DMA_CHn_CNT* upon a CTZ condition.

11. ADC

The 12-bit successive approximation (SAR) ADC includes a single-ended input multiplexer and an integrated reference generator. It can measure up to 12 single-ended external analog inputs, internal power supplies, or a differential internal temperature sensor.

The device samples any or all of the inputs in a user-defined conversion sequence which can execute once or run continuously. The conversion sequence can immediately begin when enabled by software or a specific hardware event such as a timer interrupt or transition on an external GPIO pin. A user-programmable delay can be inserted between conversions in continuous mode.

- 12-bit successive approximation ADC
- Conversion speed up to 1MSPS
- Internal reference without external capacitor
- Support for external reference from 2.048V to V_{DDA}
- Capacitor calibration
- Internal die temperature sensor

11.1 Operation

Measurements are performed in a series of user-defined channel measurements called a conversion sequence. Conversion sequences can be set up as a single conversion sequence or continuous conversion sequences. Software triggered and hardware triggered conversion sequences are supported.

Conversion sequences can measure single or multiple channels. The specific channels for a conversion sequence are set using the ADC channel select registers (*ADC_CHSEL7*:*ADC_CHSEL0*) and the *slot0_id* through *slot31_id* fields. A conversion sequence begins with the channel configured for slot 0 and continues sequentially through the software configured number of slots (*ADC_CTRL1*.*num_slots*) up to slot 31.

Each measurement is pushed onto the FIFO to be read by software. Threshold interrupts alert the software when the FIFO must be read to avoid overwriting previous measurements. Several data formats are available to the user.

11.1.1 Input Channels

Each of the input channels is shown in *Table 11-1*.

Table 11-1: MAX78002 Channel Assignments (All tables need updates)

Channel ID	Source	Mode	Alternate Function Name[1]
0	AIN0	Single-ended	AIN0
1	AIN1	Single-ended	AIN1
2	AIN2	Single-ended	AIN2
3	AIN3	Single-ended	AIN3
4	AIN4	Single-ended	AIN4
5	AIN5	Single-ended	AIN5
6	AIN6	Single-ended	AIN6
7	AIN7	Single-ended	AIN7
8	V_{COREA}	Single-ended	-
9	V_{COREB}	Single-ended	-
10	$\frac{V_{LDO2P5}}{4}$	Single-ended	-
11	V_{LDO0P9}	Single-ended	-

Channel ID	Source	Mode	Alternate Function Name[1]
12	$\frac{V_{DDA}}{2}$	Single-ended	-
13	Temperature Sensor	Differential	-
14	$\frac{V_{BB}}{4}$	Single-ended	-
15	$\frac{V_{DDB}}{4}$	Single-ended	
16	V_{SS}	Single-ended	N/A
31 - 17	Reserved	-	-

[1] Refer to the device data sheet's pin description table for pin numbers and alternate function assignments.

Table 11-2: ADC Voltage Divider Configuration for Channels 0 through 12

Setting	Divider Selection *MCR_ADCCFG2.ch<n>*[1]	Dynamic Pullup Enable *MCR_ADCCFG1.ch<n>_pu_dyn*[1]	Automatic Disable During Device Low Power Modes (Channels 0 to 11)[2]
Pass-through divide by 1	0	N/A	0
Voltage divide by 2, 5kΩ	1	0: Divider enabled always 1: Divider enabled only when channel active	*MCR_ADCCFG0.lp_5k_dis* = 1
Voltage divide by 2, 50kΩ	2		*MCR_ADCCFG0.lp_50k_dis* = 1

[1] <n> = Channel number (0 to 12)

[2] The disable settings only apply to channels 0 through 11. Channel 12's pullup, if enabled, is always disabled during low-power modes.

11.2 Clocks and Timing

Clock and timing configurations are calculated based on the application-specific sampling rate requirements. Several parameters can be adjusted to optimize the ADC power consumption, accuracy, and startup time. *Table 11-3* shows the ADC clock sources available for the device.

Table 11-3: MAX78002 ADC Clock Sources

ADC_CLKCTRL.clksel	Source (f_{ADC_SRC})
0	SYS_CLK
1	ADC_CLK_EXT (P1.10 / AF2)
2	IBRO
3	ERFO

The ADC clock frequency (f_{SAR_CLK}) is derived from a selectable clock source (f_{ADC_SRC}) and divided by a selectable clock divider, as shown in *Equation 11-1*. *Table 11-3* lists the available sources for f_{ADC_SRC}. The clock divider is selected using the *ADC_CLKCTRL.clksel* field.

Equation 11-1: ADC Clock Generation

$$For\ ADC\_CLKCTRL.clkdiv \leq 3 \quad f_{SAR\_CLK} = \frac{f_{ADC\_SRC}}{2^{(ADC\_CLKCTRL.clkdiv+1)}}$$

$$For\ ADC\_CLKCTRL.clkdiv > 3\ f_{SAR\_CLK} = f_{ADC\_SRC}$$

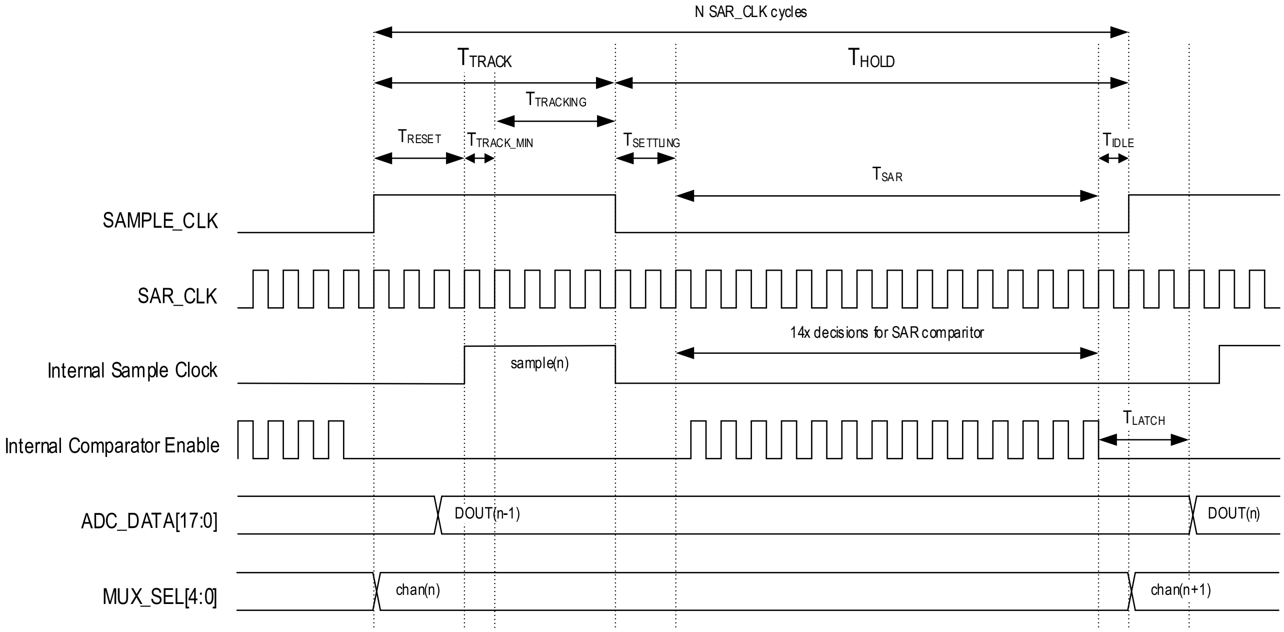
$$f_{SAR\_CLK} \leq 25MHz$$

The SAMPLE_CLK frequency determines the sampling rate, conversion time, and the delay between conversions. It is defined by a high track time and a low hold time of SAR_CLK periods. The sum of the track and hold defines the SAMPLE_CLK frequency (sample rate). *Figure 11-1* shows the SAMPLE_CLK and its relationship to the track and hold values.

Equation 11-2: Sample Clock Frequency Calculation

$$t_{SAMPLE\_CLK} = (TRACK + HOLD) \times t_{SAR\_CLK}$$

Figure 11-1: ADC Sample Clock



NOTE: $T_{RESET} = 3$
$T_{TRACK\_MIN} = 1$
$T_{TRACKING} =$ *ADC_SAMPCLKCTRL.track_cnt*
$T_{SETTLING} = 2$
$T_{SAR} = 14$
$T_{IDLE} = 1 +$ *ADC_SAMPCLKCTRL.idle_cnt*

Equation 11-3: T_{TRACK} Calculation

$$T_{TRACK} = T_{RESET} + T_{TRACK\_MIN} + T_{TRACKING}$$
$$T_{TRACK} = 4 + ADC\_SAMPCLKCTRL.track\_cnt$$
$$T_{TRACK} \geq 8$$

Equation 11-4: T_{HOLD} Calculation

$$T_{HOLD} = T_{SETTLING} + T_{SAR} + T_{IDLE}$$
$$T_{HOLD} = 17 + ADC\_SAMPCLKCTRL.idle\_cnt$$
$$T_{HOLD} \geq 17$$

The ADC requires a minimum T_{TRACK} of 8 SAR_CLK cycles and a minimum T_{HOLD} of 17 SAR_CLK cycles. The ADC_SAMPCLKCTRL.*track_cnt* and ADC_SAMPCLKCTRL.*idle_cnt* fields add SAR_CLK cycles to the track and hold, as shown in *Equation 11-3* and *Equation 11-4*.

As an example, the following steps show the settings required to achieve a 1MSPS rate for the ADC using the ERFO as the clock source.

1. Select the ADC_SRC clock as the ERFO.
    a. Set *ADC_CLKCTRL.clksel* to 3.
2. Select the clock divider to achieve a valid SAR_CLK frequency using *Equation 11-1*.
    a. Set *ADC_CLKCTRL.clkdiv* to 4 (divide by 1, $f_{SAR\_CLK} \leq 25MHz$)
    b. $t_{SAR\_CLK} = 40ns$
3. Determine the SAMPLE_CLK for 1MSPS
    a. $TRACK + HOLD = \frac{25MHz}{1MHz} = 25$
4. Determine the *ADC_SAMPCLKCTRL.track_cnt* setting using *Equation 11-3*.
    a. *ADC_SAMPCLKCTRL.track_cnt* = 4 ($T_{TRACK} \geq 8$)
5. Determine the *ADC_SAMPCLKCTRL.idle_cnt* setting using *Equation 11-4*.
    a. HOLD = 25 - T_{TRACK} = 25 - 8 = 17
    b. *ADC_SAMPCLKCTRL.idle_cnt* = 0 ($T_{HOLD} \geq 17$)

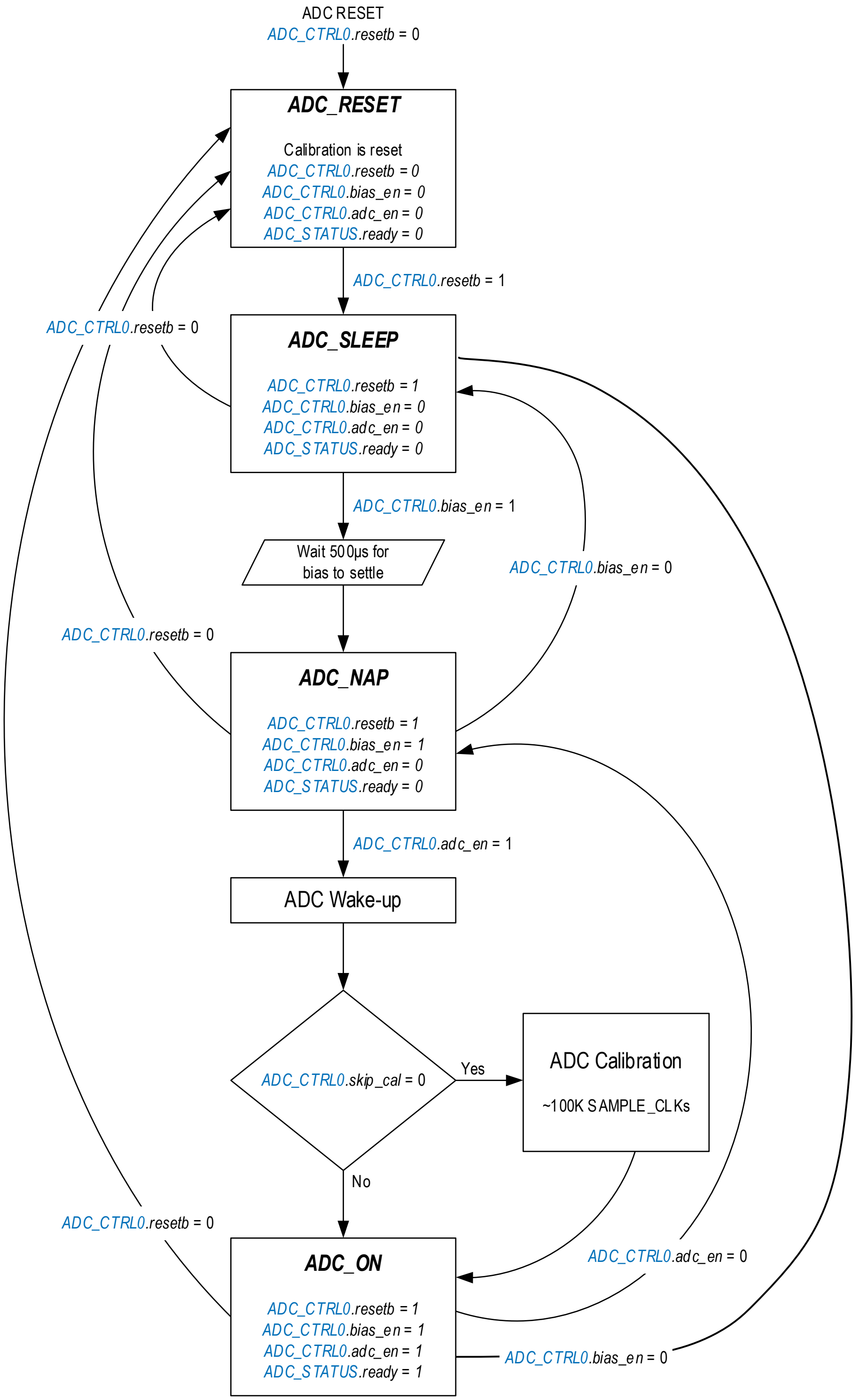
11.3 Operating Modes

Four operating modes allow the ADC to minimize power consumption based on the current needs of the peripheral. After a POR, system reset, peripheral reset (*GCR_RST0.adc* = 1), or software directly resetting the ADC (*ADC_CTRL0.resetb* = 0), the ADC bias regulator is disabled, and the ADC calibration values are reset to 0. The bias regulator must be enabled before performing a measurement, and a capacitor calibration can optionally be performed. Enabling the bias regulator requires 500µs before performing a conversion. Section *11.3.1* describes initializing the ADC from *ADC_RESET*. Section *11.3.1.2* describes entering *ADC_NAP* state. Section *11.3.1.3* describes the steps required to enter the *ADC_ON* state and perform an ADC capacitor calibration, and section *11.3.1.4* describes the steps to enter the *ADC_ON* state without performing a calibration. ADC calibration is only necessary after an ADC reset occurs, changing the reference, or changing environmental conditions such as temperature. For example, a device moving from an indoor environment to an outdoor environment might require a recalibration depending on application requirements. *Figure 11-2* shows the ADC operating modes state diagram. *Table 11-4* shows the configuration bits' state and the status bit's state for each operating mode.

Table 11-4: ADC Operating States

Instance	***ADC_CTRL0.resetb***	***ADC_CTRL0.bias_en***	***ADC_CTRL0.adc_en***	***ADC_STATUS.ready*** **(Status)**
ADC_ON	1	1	1	1
ADC_NAP	1	1	0	0
ADC_SLEEP	1	0	0	0
ADC_RESET	0	0	0	0

Figure 11-2: ADC Operating Modes State Diagram



The ADC remains in the *ADC_RESET* state while the *ADC_CTRL0.resetb* field is 0. The ADC enters *ADC_SLEEP* when the *ADC_CTRL0.resetb* field is set to 1. *ADC_SLEEP* is a low-power mode with the bias regulator disabled.

Enabling the bias regulator transitions the ADC to *ADC_NAP* state. Setting *ADC_CTRL0.bias_en* to 1 turns on the bias regulator required for ADC conversions. The bias regulator requires approximately 500µs to warm up. There is no dedicated status bit indicating the transition is complete, so software must measure the required time. The ADC's sample rate should be configured with the ADC in the *ADC_NAP* state.

The peripheral enters the *ADC_ON* state when the *ADC_CTRL0.adc_en* field is set to 1. If the *ADC_CTRL0.skip_cal* field is 1, the device performs the ADC auto-calibration, which takes approximately 100ms to complete. After the auto-calibration is complete, or immediately if it was not performed, the peripheral enters the *ADC_ON* state. An ADC-ready event occurs, indicating conversions can begin. The *ADC_STATUS.ready* field remains 1 while in the *ADC_ON* state.

11.3.1 ADC Initialization

11.3.1.1 Entering ADC_SLEEP State

The ADC must be initialized before use. These steps are performed once and are not needed before every conversion. Analog inputs are usually dedicated and not dynamically switched with digital functions.

To initialize the ADC and enter *ADC_SLEEP*:

1. Clear *GCR_PCLKDIS0.adc* to 0 to enable the ADC peripheral clock.
2. Clear *ADC_CTRL0.resetb* to 0 to enter reset.
3. Select the ADC_SRC clock from *Table 11-3* and set it to the selected clock using the *ADC_CLKCTRL.clksel* field.
4. Configure the SAR_CLK as described in *Clocks and Timing* using the *ADC_CLKCTRL.clkdiv* field.
5. Select the ADC reference source:
    - External: *MCR_ADCCFG0.ext_ref* to 1
    - Internal, 1.25V: Clear *MCR_ADCCFG0.ext_ref* to 0 and clear *MCR_ADCCFG0.ref_sel* to 0
    - Internal, 2.048: Clear *MCR_ADCCFG0.ext_ref* to 0 and set *MCR_ADCCFG0.ref_sel* to 1
6. If desired, enable the external input voltage dividers for the desired channels, as shown in *Table 11-2*. The voltage divider can always be active or only during the channel measurement. It can also be configured to disable the voltage divider during low-power modes.
7. Configure the GPIO associated with the desired external channels as inputs in high impedance mode. Configure the alternate function mode as indicated in *Table 11-1*.
8. Set the *ADC_CTRL0.resetb* to 1 to enter the *ADC_SLEEP* state.

11.3.1.2 Entering ADC_NAP State

After the ADC is in *ADC_SLEEP*, enter *ADC_NAP* state as follows:

1. Enable the ADC bias regulator by setting *ADC_CTRL0.bias_en* to 1.
2. Wait 500µs for the bias regulator to settle.
3. The ADC is now in the *ADC_NAP* state.

11.3.1.3 Entering ADC_ON State Using Calibration

Autocalibration can only be performed when the ADC is in *ADC_NAP*. The autocalibration settings remain loaded as long as the ADC does not enter *ADC_RESET*. Perform the following steps to perform calibration when the ADC is in *ADC_NAP*:

1. Clear the *ADC_CTRL0.skip_cal* bit to 0.
2. Configure the ADC SAMPLE_CLK using the *ADC_SAMPCLKCTRL.track_cnt* and *ADC_SAMPCLKCTRL.idle_cnt* fields as described in *Clocks and Timing*.
3. Clear the ADC interrupt flags register by writing 0xFFFF FFFF to the *ADC_INTFL* register.
4. Load the reference trim values for the desired reference. See *ADC SFR Interface* for details.
5. Set the *ADC_CTRL0.adc_en* field to 1.
6. The calibration is complete, and the ADC enters the *ADC_ON* state when the *ADC_INTFL.ready* field reads 1.

Once the ADC is in the *ADC_ON* state, conversions can be started.

11.3.1.4 Entering ADC_ON State Skipping Calibration

If calibration has previously been performed, the calibration step can be skipped. Enter *ADC_ON* state without calibration by performing the following steps:

1. Set the *ADC_CTRL0.skip_cal* bit to 1.
2. Configure the ADC SAMPLE_CLK using the *ADC_SAMPCLKCTRL.track_cnt* and *ADC_SAMPCLKCTRL.idle_cnt* fields as described in *Clocks and Timing*.
3. Clear the ADC interrupt flags register by writing 0xFFFF FFFF to the *ADC_INTFL* register.
4. Set the *ADC_CTRL0.adc_en* field to 1.
5. The ADC enters the *ADC_ON* state when the *ADC_INTFL.ready* field reads 1.

Once the ADC is in the *ADC_ON* state, conversions can be started.

11.4 ADC SFR Interface

The ADC supports loading of several configuration and trim values. Each reference includes specific trim values that are stored in the FCR_ADCREFTRIM0, FCR_ADCREFTRIM1, and *FCR_ADCREFTRIM2* registers. Additionally, the bias counter and wake-up counter are configurable to achieve optimum performance.

11.4.1 Determination of Bias and Wake-up Counter Settings

The ADC bias and wake-up are configurable to achieve optimum performance based on the sample rate of the ADC. The bias must be at least 500µs and the ADC wake-up timer must be at least 30µs to ensure optimum reference buffer settling requirements. The settings for the bias counter and wake-up counter are dependent on the configured ADC sample rate. *Table 11-5* shows the settings for the bias and wake-up counters and the resulting number of clock cycles each setting achieves. The following steps show how to determine the settings for the bias counter and wake-up counter for a sample rate of 1MSPS.

1. The bias counter must be 500µs, which results in $1MHz \times 500\mu s = 500 \text{ cycles}$.
   a. Referring to *Table 11-5*, the closest bias counter setting to achieve at least 500 cycles is 7 (512 clock cycles).
2. The wake-up counter must be 30µs, which results in $1MHz \times 30\mu s = 30 \text{ cycles}$.
   a. Referring to *Table 11-5*, the closest wake-up counter setting to achieve at least 30 cycles is 11 (32 clock cycles).

Table 11-5: Bias and Wake-up Clock Cycle Selection

Config Counter Setting	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Bias Counter Clock Cycles	4	8	16	32	64	128	256	512	1024	1536	2048	2560	3072	3584	4094	4608
Wake-up Clock Cycles	2	4	6	8	10	12	14	16	20	24	28	32	36	40	44	48

11.4.2 Using the ADC SFR Interface to Load the Reference Trim, and Bias/Wake-up Counter Settings

Note: The loading of the reference trim values must be performed while the ADC is in the ADC NAP state and are only applied when the ADC enters the ON state using calibration. See Entering ADC_ON State Using Calibration.

11.4.3 1.25V Internal Reference Trim

Perform the following steps to load the trim values for the 1.25V internal reference.

1. Write address 0x0B to *ADC_SFRADDR*.
2. Read the SFR data by reading a byte from the *ADC_SFRRDDATA* register.
3. Mask off the upper two bits of the data read by performing a bit-wise AND of the value read with 0xC0.
4. Perform a bit-wise OR of the result of step 3 with FCR_ADCREFTRIM0.*vx2_tune*.
5. Write the byte from step 4 to the *ADC_SFRWRDATA* register.
6. Write address 0x0C to *ADC_SFRADDR*.
7. Write the following byte to the *ADC_SFRWRDATA* register.
    a. Perform a bitwise OR of:
        1) *FCR_ADCREFTRIM2*.*iboost_1p25* shift left 7 and
        2) FCR_ADCREFTRIM0.*vrefp*
8. Write address 0x0D to *ADC_SFRADDR*.
9. Read the SFR data by reading a byte from the *ADC_SFRRDDATA* register.
10. Mask off the upper bit of the data read by performing a bit-wise AND of the value read with 0x80.
11. Perform a bit-wise OR of the result of step 10 with FCR_ADCREFTRIM0.*vrefm.*
12. Write the byte from step 11 to the *ADC_SFRWRDATA* register.
13. Write address 0x0E to *ADC_SFRADDR*.
14. Read the SFR data by reading a byte from the *ADC_SFRRDDATA* register.
15. Mask off bits 2 and 3 of the data read by performing a bit-wise AND of the value read with 0x0C.
16. Write the following byte to the *ADC_SFRWRDATA* register.
    a. Perform a bitwise OR of the data from step 15 and:
        1) *FCR_ADCREFTRIM2*.*idrv_1p25* shift left 4
        2) FCR_ADCREFTRIM0.*vcm*
17. Write address 0x05 to *ADC_SFRADDR*.
18. Read the SFR data by reading a byte from the *ADC_SFRRDDATA* register.
19. Mask off bits 4 - 7 of the data read by performing a bit-wise AND of the value read with 0xF0.
20. Write the following byte to the *ADC_SFRWRDATA* register.
    a. Perform a bitwise OR of the data from step 19 and the bias counter setting. See *Determination of Bias and Wake-up Counter Settings* for details on calculating this value.
21. Write address 0x06 to *ADC_SFRADDR*.
22. Read the SFR data by reading a byte from the *ADC_SFRRDDATA* register.
23. Mask off bits 4 - 7 of the data read by performing a bit-wise AND of the value read with 0xF0.
24. Write the following byte to the *ADC_SFRWRDATA* register.
    a. Perform a bitwise OR of the data from step 23 and the calculated wake-up counter setting. See *Determination of Bias and Wake-up Counter Settings* for details on calculating this value.
25. Move the ADC into the ADC_ON state using calibration.

11.4.4 2.048V Internal Reference Trim

Perform the following steps to load the trim values for the 2.048V internal reference.

1. Write address 0x0B to *ADC_SFRADDR*.
2. Read the SFR data by reading a byte from the *ADC_SFRRDDATA* register.
3. Mask off the upper two bits of the data read by performing a bit-wise AND of the value read with 0xC0.
4. Perform a bit-wise OR of the result of step 3 with FCR_ADCREFTRIM1.*vx2_tune*.
5. Write the byte from step 4 to the *ADC_SFRWRDATA* register.
6. Write address 0x0C to *ADC_SFRADDR*.
7. Write the following byte to the *ADC_SFRWRDATA* register.
    a. Perform a bitwise OR of:
        1) *FCR_ADCREFTRIM2*.*iboost_2p048* shift left 7 and
        2) FCR_ADCREFTRIM1.*vrefp*
8. Write address 0x0D to *ADC_SFRADDR*.
9. Read the SFR data by reading a byte from the *ADC_SFRRDDATA* register.
10. Mask off the upper bit of the data read by performing a bit-wise AND of the value read with 0x80.
11. Perform a bit-wise OR of the result of step 10 with FCR_ADCREFTRIM1.*vrefm.*
12. Write the byte from step 11 to the *ADC_SFRWRDATA* register.
13. Write address 0x0E to *ADC_SFRADDR*.
14. Read the SFR data by reading a byte from the *ADC_SFRRDDATA* register.
15. Mask off bits 2 and 3 of the data read by performing a bit-wise AND of the value read with 0x0C.
16. Write the following byte to the *ADC_SFRWRDATA* register.
    a. Perform a bitwise OR of the data from step 15 and:
        1) *FCR_ADCREFTRIM2*.*idrv_2p048* shift left 4
        2) FCR_ADCREFTRIM1.*vcm*
17. Write address 0x05 to *ADC_SFRADDR*.
18. Read the SFR data by reading a byte from the *ADC_SFRRDDATA* register.
19. Mask off bits 4 - 7 of the data read by performing a bit-wise AND of the value read with 0xF0.
20. Write the following byte to the *ADC_SFRWRDATA* register.
    a. Perform a bitwise OR of the data from step 19 and the bias counter setting. See *Determination of Bias and Wake-up Counter Settings* for details on calculating this value.
21. Write address 0x06 to *ADC_SFRADDR*.
22. Read the SFR data by reading a byte from the *ADC_SFRRDDATA* register.
23. Mask off bits 4 - 7 of the data read by performing a bit-wise AND of the value read with 0xF0.
24. Write the following byte to the *ADC_SFRWRDATA* register.
    a. Perform a bitwise OR of the data from step 23 and the calculated wake-up counter setting. See *Determination of Bias and Wake-up Counter Settings* for details on calculating this value.
25. Move the ADC into the ADC_ON state using calibration.

11.4.5 External Reference Trim

Perform the following steps to load the trim values for the external reference.

1. Write address 0x0B to *ADC_SFRADDR*.
2. Read the SFR data by reading a byte from the *ADC_SFRRDDATA* register.
3. Mask off the upper bit of the data read by performing a bit-wise AND of the value read with 0x80.
4. Perform a bit-wise OR of the result of step 3 with *FCR_ADCREFTRIM2.vx2_tune*.
5. Write the byte from step 4 to the SFR by writing it to the *ADC_SFRWRDATA* register.
6. Write address 0x0E to *ADC_SFRADDR*.
7. Read the SFR data by reading a byte from the *ADC_SFRRDDATA* register.
8. Write the following byte to the *ADC_SFRWRDATA* register.
    a. Perform a bitwise OR of the data from step 7 and *FCR_ADCREFTRIM2.vcm*
9. Write address 0x05 to *ADC_SFRADDR*.
10. Read the SFR data by reading a byte from the *ADC_SFRRDDATA* register.
11. Mask off bits 4 - 7 of the data read by performing a bit-wise AND of the value read with 0xF0.
12. Write the following byte to the *ADC_SFRWRDATA* register.
    a. Perform a bitwise OR of the data from step 11 and the bias counter setting. See *Determination of Bias and Wake-up Counter Settings* for details on calculating this value.
13. Write address 0x06 to *ADC_SFRADDR*.
14. Read the SFR data by reading a byte from the *ADC_SFRRDDATA* register.
15. Mask off bits 4 - 7 of the data read by performing a bit-wise AND of the value read with 0xF0.
16. Write the following byte to the *ADC_SFRWRDATA* register.
    a. Perform a bitwise OR of the data from step 16 and the calculated wake-up counter setting. See *Determination of Bias and Wake-up Counter Settings* for details on calculating this value.
17. Move the ADC into the ADC_ON state using calibration.

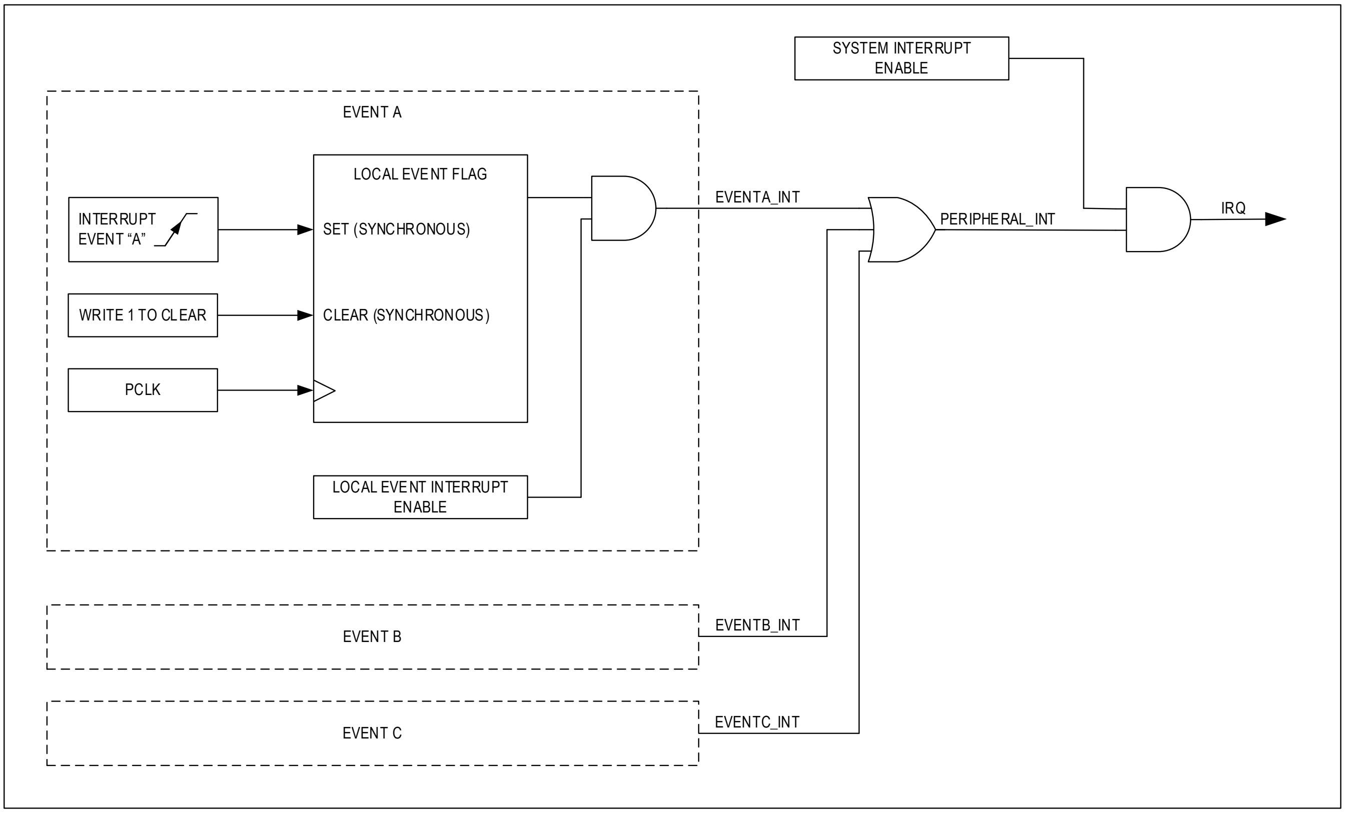
11.5 Interrupts

Multiple interrupt events are supported. Each event has a flag and interrupt enable field in the peripheral's register set unless specified otherwise. The event flag is "edge-triggered" and set when the event occurs. Further occurrences of the event do not cause any additional effect if the flag is set to 1. The interrupt signal from the event is active whenever the flag and enable fields are both set to 1. An event flag should always be cleared by writing 1 to the flag's bit position before setting its interrupt enable field.

All the interrupt signals from local events in a peripheral are OR'd together to create a peripheral interrupt for the NVIC. Some peripherals can further qualify the generation of the interrupt with one or more higher-level system interrupt enables.

Figure 11-3 is a functional diagram showing this relationship.

Figure 11-3: Interrupt Event Signal Generation



Clear a local event flag by writing a 1 to the flag. Always clear a local event flag before setting the corresponding interrupt enable field.

The interrupt events supported are listed in *Table 11-6*.

Table 11-6: MAX78002 Interrupt Events

Event	Description	Interrupt Flag	Interrupt Enable
Receive FIFO Threshold	*ADC_STATUS*. *fifo_level* > *ADC_FIFODMACTRL*.*thresh*.	*ADC_INTFL*.*fifo_lvl*	*ADC_INTEN*.*fifo_lvl*
Receive FIFO Overflow	Hardware FIFO write when *ADC_STATUS*.*fifo_level*= 0b111.	*ADC_INTFL*.*fifo_ofl*	*ADC_INTEN*.*fifo_ofl*
Receive FIFO Underflow	Read from FIFO when *ADC_STATUS*.*fifo_level* = 0	*ADC_INTFL*.*fifo_ufl*	*ADC_INTEN*.*fifo_ufl*
Data Clipped	An ADC measurement has been clipped	*ADC_INTFL*.*clipped*	*ADC_INTEN*.*clipped*
Conversion Sequence Done	A continuous conversion sequence completed while *ADC_CTRL1*.*start* is 1 or a single conversion sequence completed.	*ADC_INTFL*.*conv_done*	*ADC_INTEN*.*conv_done*
Conversion Sequence Complete	A continuous or single conversion sequence is finished.	*ADC_INTFL*.*seq_done*	*ADC_INTEN*.*seq_done*
Conversion Sequence Started	A continuous or single conversion sequence has started. This field can be used to tell when a hardware trigger occurred.	*ADC_INTFL*.*seq_started*	*ADC_INTEN*.*seq_started*
Start Bit Set	*ADC_CTRL1*.*start* transitioned from 0 to 1	*ADC_INTFL*.*start_det*	*ADC_INTEN*.*start_det*

Event	Description	Interrupt Flag	Interrupt Enable
Conversion Sequence Abort	*ADC_CTRL1.start* transitioned from 1 to 0 before a conversion started.	*ADC_INTFL.abort*	*ADC_INTEN.abort*
ADC Ready	ADC transitioned to the *ADC_ON* state.	*ADC_INTFL.ready*	*ADC_INTEN.ready*

11.6 FIFO Operation

Measurement results are pushed onto the 8-word FIFO. Access the FIFO by reading the *ADC_DATA* register. Software must be sure to read the FIFO often enough to prevent data from being lost.

The current level of the FIFO is read from *ADC_STATUS.fifo_level.* The same register also contains empty and full status flags for the FIFO. Multiple FIFO events are supported.

- A FIFO threshold event occurs when *ADC_STATUS.fifo_level* exceeds the value *ADC_FIFODMACTRL.thresh*. This event is an indication that data should be read from the FIFO soon or can be lost.
- A FIFO overflow event occurs when a measurement is loaded into the FIFO when *ADC_STATUS.fifo_level* equals 7. Previous data in the FIFO has been overwritten and lost. It is possible to use the channel ID field in the *ADC_DATA* register to determine which measurement results have been overwritten. The FIFO should be flushed, and the current conversion sequence restarted.
- A FIFO underflow event occurs when the FIFO is read when *ADC_STATUS.fifo_level* is equal to 0.

11.7 Averaging

The ADC can take multiple measurements of a channel and average the results. Averaging is enabled when the *ADC_CTRL1.avg* field is set to a non-zero value. Each slot in the conversion is sampled 2^N times where N is the *ADC_CTRL1.avg* value. The results are then averaged and reported in the slot. The averaging setting applies to all measured channels. A clipped measurement of any of the samples sets the clipped status field for the averaged result.

Note: The averaging is applied equally to all slots. Setting a large averaging value can result in a long conversion time for a sequence to complete if multiple channels are enabled.

11.8 Conversion Results

The results and status information are read from the *ADC_DATA* register. The selection of the format of the *ADC_DATA* register is determined using the *ADC_FIFODMACTRL.data_format* field. Each of the format options is shown in *Table 11-7*. A visual representation of the corresponding format of the *ADC_DATA* register for the temperature sensor input is shown in *Figure 11-5*, and all other input channels are shown in *Figure 11-4*.

In processed modes, the status information includes:

- A channel identifier (*ADC_DATA.chan*) to assist in identifying the channel associated with the data. Although the channel is known when reading the slot, this helps identify data later if saved to memory.
- A clipped status (*ADC_DATA.clipped*) field indicating if the result was beyond the ADC limits (either positive or negative). For an averaged measurement, the result is marked clipped if any of the samples were clipped.
- The validity of the data (*ADC_DATA.invalid*). Data is marked as invalid if clipped, has an invalid channel assignment, or the channel is not ready.

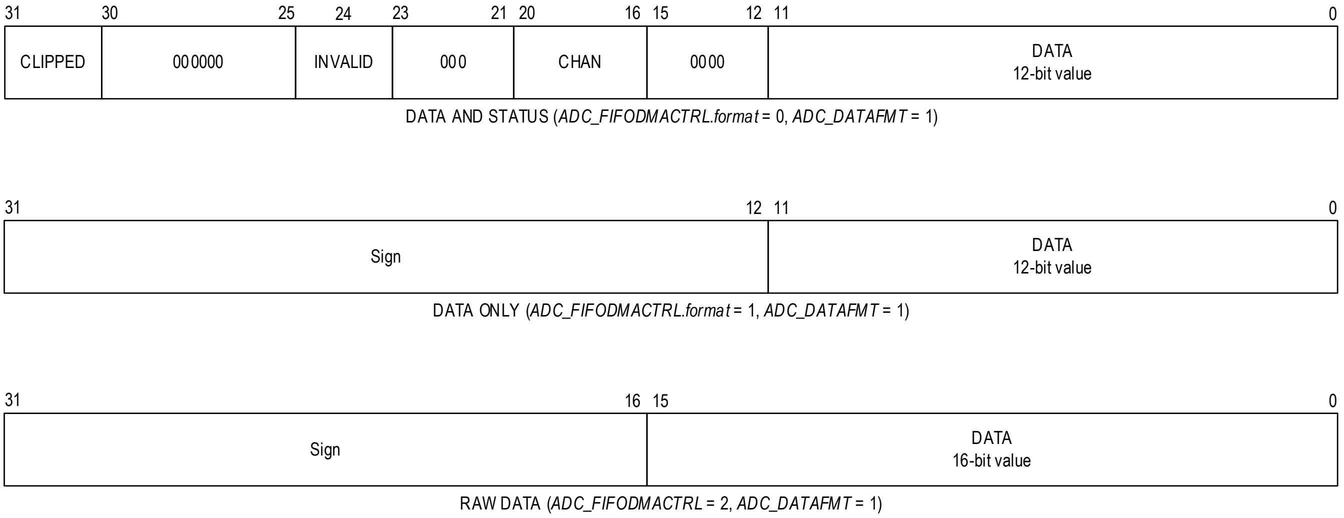
The result formatting options are shown in *Table 11-7*.

Table 11-7: ADC_DATA Register Result Formatting

Mode	*ADC_FIFODMACTRL.* *data_format*	Format code	Channel ID	Clipped	Invalid Flag	Data Format
Single-ended	0	Data and Status	Yes	Yes	Yes	12-bit unsigned
	1	Data Only	No	*ADC_CHSTATUS*	No	12-bit unsigned
	2	Raw data only	-	*ADC_CHSTATUS*	Yes	16-bit signed 2's complement bit 16 is the sign bit
Differential (Temperature Sensor Only)	0	Data and Status	Yes	Yes	Yes	12-bit signed 2's complement
	1	Data Only	No	*ADC_CHSTATUS*	No	12-bit signed 2's complement
	2	Raw data only	-	*ADC_CHSTATUS*	Yes	16-bit signed 2's complement bit 16 is the sign bit

The information structure depends on the channel mode (single-ended or differential) and the selected data format, as shown in *Figure 11-3*.

Figure 11-4: ADC Result Formats (Single-Ended)

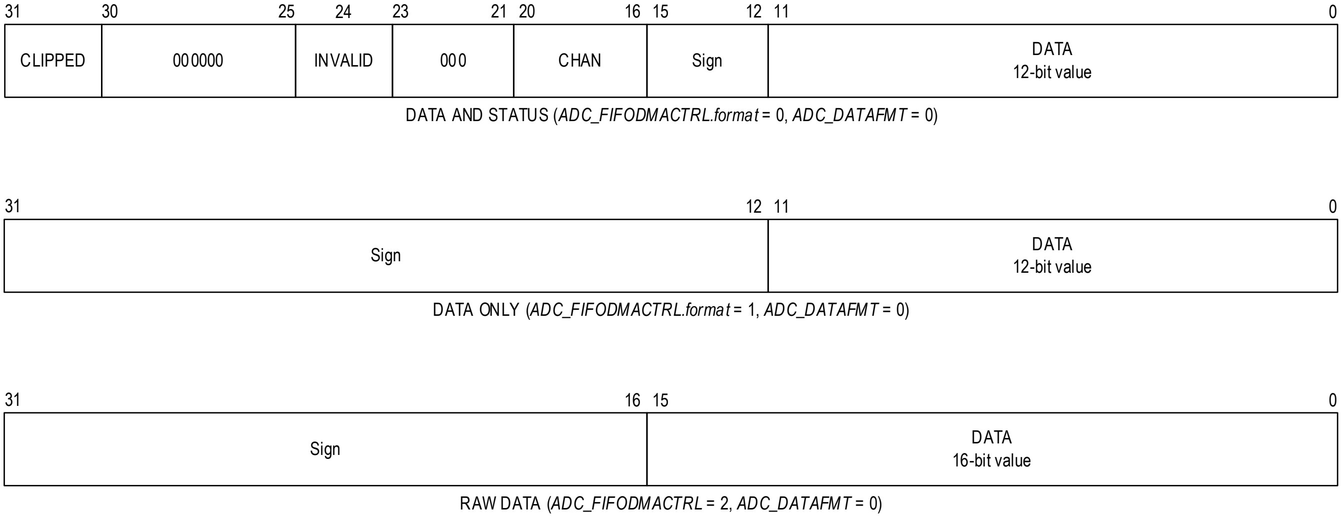


DATA AND STATUS (*ADC_FIFODMACTRL.format* = 0, *ADC_DATAFMT* = 1)

DATA ONLY (*ADC_FIFODMACTRL.format* = 1, *ADC_DATAFMT* = 1)

RAW DATA (*ADC_FIFODMACTRL* = 2, *ADC_DATAFMT* = 1)

Figure 11-5: ADC Result Formats (Differential, Temperature Sensor Only)



31	30 25	24	23 21	20 16	15 12	11 0
CLIPPED	00 0000	INVALID	000	CHAN	Sign	DATA 12-bit value

DATA AND STATUS (*ADC_FIFODMACTRL.format* = 0, *ADC_DATAFMT* = 0)

31 12	11 0
Sign	DATA 12-bit value

DATA ONLY (*ADC_FIFODMACTRL.format* = 1, *ADC_DATAFMT* = 0)

31 16	15 0
Sign	DATA 16-bit value

RAW DATA (*ADC_FIFODMACTRL* = 2, *ADC_DATAFMT* = 0)

11.9 Conversions

11.9.1 Conversion Sequence Triggers

A conversion sequence is initiated by either a software or hardware trigger. This flexibility allows either manual or on-demand measurements using a timer peripheral or an external GPIO. *Table 11-8* lists the hardware triggers available to start a conversion sequence.

Table 11-8: MAX78002 Hardware Conversion Triggers

ADC_CTRL1.trig_sel	**Source**
0	TMR0 output
1	TMR1 output
2	TMR2 output
3	TMR3 output
4	ADC_HW_TRIG_A[1]
5	ADC_HW_TRIG_B[1]
6	ADC_HW_TRIG_C[1]
7	Temperature sensor measurement ready
1. Refer to the device data sheet's pin description table for alternate function assignments. Not all alternate functions are available on all packages.	

A software-triggered conversion sequence can run once and stop (single conversion sequence) or continuously (continuous conversion sequence). A conversion sequence begins when the software changes the *ADC_CTRL1.start* field from 0 to 1. Conversion sequences run until all slots are completed for both a continuous or single sequence.

A software-triggered continuous sequence converts all slots and then repeats the process, with a programmable delay between sequences, as long as the *ADC_CTRL1.start* field is set to 1. Setting *ADC_CTRL1.start* to 0 during an active continuous conversion sequence stops the sequence at the completion of the active sequence.

A hardware-triggered conversion sequence starts when the selected trigger becomes active. Only one of the hardware triggers, shown in *Table 11-8*, can be selected for the conversion sequence.

The hardware trigger is armed when the software changes *ADC_CTRL1.start* from a 0 to a 1. The device waits until the trigger event occurs, performs one conversion sequence, and then idles until the trigger event occurs again or software clears the *ADC_CTRL1.start* field to 0.

The hardware trigger source must be running and properly configured to generate the trigger signal. Trigger sources are edge-triggered; the event is only recognized if a GPIO pin transitions from low to high or a timer output signal transitions from inactive to active. As a result, software must clear the GPIO or timer output each time before the hardware trigger event occurs for the trigger to be recognized. See *Configuration* in the GPIO chapter for details on configuring port pin alternate functions.

Table 11-9: Conversion Sequence Configurations

Conversion Sequence Type	Sequence Start	*ADC_CTRL1.cnv_mode*	*ADC_CTRL1.trig_mode*
Software-Triggered, continuous	*ADC_CTRL1.start* 0 → 1 Or *ADC_CTRL1.start* 0 → 1 and *ADC_INTFL.seq_done* 0 → 1 after delay	1	0
Software-Triggered, single conversion sequence	*ADC_CTRL1*.start 0 → 1	0	0
Hardware-Triggered, continuous	*ADC_CTRL1.start* 0 → 1 (armed) Trigger event Or *ADC_CTRL1.start* 0 → 1 and *ADC_INTFL.seq_done* 0 → 1 after delay	1	1
Hardware-Triggered, single conversion sequence	*ADC_CTRL1.start* 0 → 1 (armed) Trigger event	0	1

11.9.2 Single Conversion Sequences

11.9.2.1 Software Triggered

To perform a software-triggered single conversion sequence:

1. Configure the ADC and enter the *ADC_ON* state as described in *Operating Modes*.
2. Clear the *ADC_CTRL1*.*trig_mode* field to 0 to select software triggering.
3. Set *ADC_CTRL1*.*cnv_mode* to 0 to select a single conversion sequence.
4. Select the number of channels to convert by setting the *ADC_CTRL1*.*num_slots* to the number of channels minus 1.
    a. As an example, set *ADC_CTRL1*.*num_slots* to 4 to perform a single conversion on 5 channels.
5. Set the desired channels for the conversion using the *ADC_CHSEL7*:*ADC_CHSEL0* registers slot fields.
    a. As an example, to perform a single conversion sequence on channels 6, 3, 8, 4, and 2 (*ADC_CTRL1*.*num_slots* = 4), set the channel select fields as follows:
        i.) *ADC_CHSEL0*.*slot0_id* = 6
        ii.) *ADC_CHSEL0*.*slot1_id* = 3
        iii.) *ADC_CHSEL0*.*slot2_id* = 8
        iv.) *ADC_CHSEL0*.*slot3_id* = 4
        v.) *ADC_CHSEL1*.*slot4_id* = 2
6. Configure *ADC_CTRL1*.*avg* to the desired number of samples to average.
    a. Set this field to 0 for a single conversion per channel. See *Averaging* for details on sample averaging.
7. Set the data format for the conversion results using the *ADC_FIFODMACTRL*.*data_format* field. See *Conversion Results* for details.
8. Clear the interrupt flags by writing 0xFFFF FFFF to the *ADC_INTFL* register.
9. Set *ADC_CTRL1*.*start* to 1 to start the conversion sequence. The conversion sequence starts immediately.

At the end of a software triggered single conversion sequence:

- Hardware sets *ADC_INTFL*.*conv_done* to 1.
- Hardware sets *ADC_INTFL*.*seq_done* to 1, indicating a sequence done event has occurred.
- Software should set *ADC_CTRL1*.*start* to 0 in to prevent additional conversions.
- The converted data is available in the *ADC_DATA* register. See *FIFO Operation* for details on the FIFO.

11.9.2.2 Hardware-Triggered

Perform a hardware-triggered single conversion sequence using the following steps:

1. Configure the ADC and enter the *ADC_ON* state as described in *Operating Modes*.
2. Clear the *ADC_CTRL1*.*start* field to 0.
3. Set *ADC_CTRL1*.*trig_mode* to 1 to select hardware triggering.
4. Configure *ADC_CTRL1*.*trig_sel* for the desired hardware trigger. See *Table 11-8* for details of available hardware triggers.
5. Set *ADC_CTRL1*.*cnv_mode* to 0 to select a single conversion sequence.
6. Configure the selected hardware trigger.
7. Select the number of channels to convert by setting the *ADC_CTRL1*.*num_slots* to the number of channels minus 1.
    a. As an example, set *ADC_CTRL1*.*num_slots* to 1 to perform a single conversion on 2 channels.
8. Set the desired channels for the conversion using the *ADC_CHSEL7*:*ADC_CHSEL0* registers slot fields.
    a. As an example, to perform a single conversion sequence on channels 11 and 12 (*ADC_CTRL1*.*num_slots* = 1), set the channel select fields as follows:
        i.) *ADC_CHSEL0*.*slot0_id* = 11
        ii.) *ADC_CHSEL0*.*slot1_id* = 12
9. Configure *ADC_CTRL1*.*avg* to the desired number of samples to average.
    a. Set this field to 0 for a single conversion per channel. See *Averaging* for details on sample averaging.
10. Set the data format for the conversion results using the *ADC_FIFODMACTRL*.*data_format* field. See *Conversion Results* for details.
11. Clear the interrupt flags by writing 0xFFFF FFFF to the *ADC_INTFL* register.
12. Set *ADC_CTRL1*.*start* to 1 to arm the conversion sequence. The conversion sequence begins when the hardware trigger is activated.
13. When the sequence is triggered, hardware sets the *ADC_INTFL*.*seq_started* field to 1.

At the end of a hardware-triggered single conversion sequence:

- Hardware sets the *ADC_INTFL*.*seq_done* field to 1, indicating a sequence done event has occurred.
- Hardware sets the *ADC_INTFL*.*conv_done* field to 1 and does not perform another conversion sequence.
- Software should set *ADC_CTRL1*.*start* to 0 to prevent additional conversions.
- The converted data is available in the *ADC_DATA* register. See *FIFO Operation* for details on the FIFO.

11.9.3 Continuous Conversion Sequences

11.9.3.1 Software-Triggered, Continuous Conversion Sequence

To configure the ADC for a software-triggered continuous conversion sequence:

1. Configure the ADC and enter the *ADC_ON* state as described in *Operating Modes*.
2. Clear the *ADC_CTRL1*.*trig_mode* field to 0 to select software triggering.
3. Set *ADC_CTRL1*.*cnv_mode* to 1 to select continuous conversion mode.
4. Select the number of channels to convert by setting the *ADC_CTRL1*.*num_slots* to the number of channels minus 1.
5. Set the desired channels for the conversion using the *ADC_CHSEL7*:*ADC_CHSEL0* registers slot fields.
6. Configure *ADC_CTRL1*.*avg* to the desired number of samples to average.
    a. Set this field to 0 for a single conversion per channel. See *Averaging* for details on sample averaging.
7. Set the data format for the conversion results using the *ADC_FIFODMACTRL*.*data_format* field. See *Conversion Results* for details.
8. Clear the interrupt flags by writing 0xFFFF FFFF to the *ADC_INTFL* register.
9. If a delay between continuous conversion sequences is desired, set the number of SAMPLE_CLKs to delay using the *ADC_RESTART*.*cnt* field.
10. Set *ADC_CTRL1*.*start* to 1 to activate the conversion sequence. The conversion sequence starts immediately.
11. Software should clear *ADC_CTRL1*.*start* to stop a continuous conversion sequence when desired.

At the end of each continuous conversion sequence:

- Hardware sets the *ADC_INTFL*.*seq_done* field to 1, indicating a sequence is complete.

  If *ADC_CTRL1*.*start* remains set to 1, the device idles for the number of sample periods specified in *ADC_RESTART*.*cnt* before repeating the conversion sequence.
- If *ADC_CTRL1*.*start* is set to 0, hardware sets *ADC_INTFL*.*conv_done* to 1 and does not perform another conversion sequence.

11.9.3.2 Hardware-Triggered, Continuous Conversion Sequence

To perform a hardware-triggered continuous conversion sequence:

1. Configure the ADC and enter the *ADC_ON* state as described in *Operating Modes*.
2. Set *ADC_CTRL1.cnv_mode* to 1 to select continuous conversion mode.
3. Set *ADC_CTRL1.trig_mode* to 1 to select hardware triggering.
4. Configure the *ADC_CTRL1.trig_sel* field for the desired hardware trigger. See *Table 11-8* for a list of hardware triggers.
5. Configure the selected hardware trigger. See *Timers (TMR/LPTMR)* for timer configuration and *Alternate Function Configuration* in the GPIO chapter for details on configuring alternate functions.
6. Select the number of channels to convert by setting the *ADC_CTRL1.num_slots* to the number of channels minus 1.
7. Set the desired channels for the conversion using the *ADC_CHSEL7*:*ADC_CHSEL0* registers slot fields.
8. Configure *ADC_CTRL1.avg* to the desired sample averaging.
   a. Set this field to 1 for a single conversion per channel. If this field is set to greater than 1, each channel selected is converted 2^{avg} number of times and averaged before moving to the next channel.
9. Set the data format for the conversion results using the *ADC_FIFODMACTRL.data_format* field. See *Conversion Results* for details.
10. Clear the interrupt flags by writing 0xFFFF FFFF to the *ADC_INTFL* register.
11. If a delay between continuous conversion sequences is desired, set the number of SAMPLE_CLKs to delay using the *ADC_RESTART.cnt* field.
12. Set *ADC_CTRL1.start* to 1 to arm the conversion sequence.
13. When the sequence is triggered, the hardware sets *ADC_INTFL.seq_started* to 1. Continuous sequences can be stopped by clearing the *ADC_CTRL1.start* field to 0.

At the end of a continuous conversion sequence:

- Hardware sets *ADC_INTFL.seq_done* to 1, indicating a sequence done event has occurred.
- If *ADC_CTRL1.start* is set to 1:
  - The device idles for the number of sample periods specified in the *ADC_RESTART.cnt* field and then starts another conversion sequence.
- If *ADC_CTRL1.start* is set to 0:
  - The hardware sets the *ADC_INTFL.conv_done* field to 1 and does not perform another conversion sequence.

11.9.3.3 Temperature Sensor

The internal temperature sensor provides a measurement of die temperature. Depending on the application, environmental changes might necessitate recalibration of the RTC or ADC. The temperature sensor returns its results in a differential measurement format. The temperature measurement takes approximately 500µs for a measurement. This time is required after the temperature sensor is enabled. The temperature sensor can be measured in a single conversion sequence or as part of a continuous conversion sequence. If the actual measurement of the temperature sensor occurs before the temperature sensor is ready, the measurement is marked as invalid.

To perform a temperature sensor conversion, a minimum of three channels must be converted. The first channel or the channel immediately before the temperature sensor must be $\frac{V_{DDA}}{2}$. Additionally, the IBRO must be enabled for the temperature sensor conversion to complete correctly. Any channel can be converted immediately after the temperature sensor.

The following steps describe how to measure the temperature sensor using a hardware triggered, single conversion sequence:

1. Configure the ADC and enter the *ADC_ON* state as described in *Operating Modes*.
    a. Entering the *ADC_ON* state with calibration ensures the most accurate temperature readings.
2. Enable the IBRO by setting *GCR_CLKCTRL.ibro_en* to 1.
3. Clear the *ADC_CTRL1.start* field to 0.
4. Set *ADC_CTRL1.trig_mode* to 1 to select hardware triggering.
5. Set *ADC_CTRL1.trig_sel* to 7 to select the temperature sensor as the hardware trigger.
6. Set *ADC_CTRL1.cnv_mode* to 0 to select a single conversion sequence.
7. Select the number of channels to convert by setting the *ADC_CTRL1.num_slots* to 2.
    a. At least three channels must be converted to measure the temperature sensor.
8. Set the *ADC_CHSEL0.slot0_id* to 12 to select $\frac{V_{DDA}}{2}$ for the channel.
9. Set the *ADC_CHSEL0.slot1_id* to 13 to select the temperature sensor for the channel.
10. Set the *ADC_CHSEL0.slot2_id* to 20 to select V_{SSA} for the channel.
    a. Any channel other than the temperature sensor can be chosen for this slot.
11. See *ADC_CTRL1.avg* to 0 for a single conversion.
    a. Averaging can be used for a more stable temperature sensor reading.
12. Set the data format for the conversion results using the *ADC_FIFODMACTRL.data_format* field. See *Conversion Results* for details. The temperature sensor is a differential measurement and always returns a signed value.
13. Clear the interrupt flags by writing 0xFFFF FFFF to the *ADC_INTFL* register.
14. Set the *ADC_CTRL1.ts_sel* field to 1 to enable the temperature sensor.
15. Set *ADC_CTRL1.start* to 1 to arm the conversion sequence. The conversion sequence begins when the temperature sensor is ready.
16. When the sequence is triggered, hardware sets the *ADC_INTFL.seq_started* field to 1.

At the end of the temperature sensor measurement:

- Hardware sets the *ADC_INTFL.seq_done* field to 1, indicating a sequence done event has occurred.
- Hardware sets the *ADC_INTFL.conv_done* field to 1 and does not perform another conversion sequence.
- Software should set *ADC_CTRL1.start* to 0 to prevent additional conversions.
- The converted data is available in the *ADC_DATA* register.
    - The temperature sensor converted data is the second value read from the *ADC_DATA* register.

Equation 11-5: Temperature Conversion Equation

$$T(°K) = \frac{Measured\ Code \times V_{REF} \times 530.582}{Full\ Scale\ Code}$$

11.10 Low-Power Analog Wake-Up Comparators

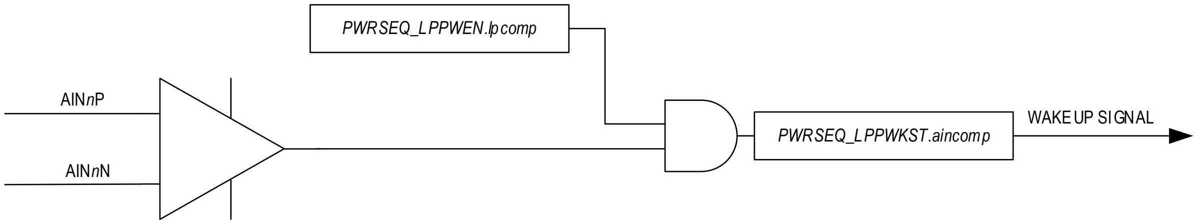
The four differential analog comparators can be used as wake-up sources for the device. These are simple op-amps, which generate an internal digital signal whenever the positive input is above the negative input.

Table 11-10 lists the alternate function name and number for each of the analog comparators positive and negative inputs.

Table 11-10: MAX78002 Analog Comparator 0 Input Selection

Comparator Number	Alternate Function Name	Alternate Function Number
0	AIN0P	1
	AIN0N	1
1	AIN1P	1
	AIN1N	1
2	AIN2P	1
	AIN2N	1
3	AIN3P	1
	AIN3N	1
Note: Refer to the device datasheet's pin description table for alternate function mapping to pin number.		

Figure 11-6: Analog Wakeup Comparators



The comparator status field dynamically shows the comparator output when both the corresponding positive, negative, and the GPIO are configured for the appropriate alternate function. When enabled, the transition of the digital signal from 0 to 1 generates a wake-up event. The wake-up comparators function independently from the ADC converter circuitry and are not affected by the ADC operating states, settings, or enable status.

Configure the comparators 1, 2, and 3 as follows:

1. Configure the comparator's inputs.
    a. Enable the comparators alternate function for the GPIO. See *Alternate Function Configuration* in the GPIO chapter for details. Refer to the device data sheet alternate function table for pin assignments.
2. Enable the comparator peripheral clock by setting *LPGCR_PCLKDIS*.*lpcomp* to 0.
    3. Clear the appropriate comparator interrupt flag by writing 1 to the comparator's interrupt flag. For example, clear comparator 1's interrupt flag by writing 1 to *LPCMP[0]*.*if*.
3. Configure the comparator's polarity by setting the polarity field as desired for the comparator.
4. Read the comparator's output by reading the comparator's *out* field. For example, to read comparator 2's output, read *LPCMP[1]*.*out*.

11.11 ADC Registers

See *Table* for the base address of this peripheral/module. See *Table 1-1* 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 11-11: ADC Register Summary

Offset	Register	Description
[0x0000]	*ADC_CTRL0*	ADC Control 0 Register
[0x0004]	*ADC_CTRL1*	ADC Control 1 Register
[0x0008]	*ADC_CLKCTRL*	ADC Clock Control Register
[0x000C]	*ADC_SAMPCLKCTRL*	ADC Sample Clock Control Register
[0x0010]	*ADC_CHSEL0*	ADC Channel Select 0 Register
[0X0014]	*ADC_CHSEL1*	ADC Channel Select 1 Register
[0x0018]	*ADC_CHSEL2*	ADC Channel Select 2 Register
[0x001C]	*ADC_CHSEL3*	ADC Channel Select 3 Register
[0x0020]	*ADC_CHSEL4*	ADC Channel Select 4 Register
[0x0024]	*ADC_CHSEL5*	ADC Channel Select 5 Register
[0x0028]	*ADC_CHSEL6*	ADC Channel Select 6 Register
[0x002C]	*ADC_CHSEL7*	ADC Channel Select 7 Register
[0x0030]	*ADC_RESTART*	ADC Conversion Restart Delay
[0x003C]	*ADC_DATAFMT*	ADC Data Format Register
[0x0040]	*ADC_FIFODMACTRL*	ADC FIFO and DMA Control Register
[0x0044]	*ADC_DATA*	ADC FIFO Register
[0x0048]	*ADC_STATUS*	ADC Status Register
[0x004C]	*ADC_CHSTATUS*	ADC Channel Status Register
[0x0050]	*ADC_INTEN*	ADC Interrupt Enable Register
[0x0054]	*ADC_INTFL*	ADC Interrupt Flags Register
[0x0060]	*ADC_SFRADDROFFSET*	ADC Address Offset Register
[0x0064]	*ADC_SFRADDR*	ADC SFR Address Register
[0x0068]	*ADC_SFRWRDATA*	ADC SFR Write Data Register
[0x006C]	*ADC_SFRRDDATA*	ADC SFR Read Data Register
[0x0070]	*ADC_SFRSTATUS*	ADC SFR Status Register

11.11.1 Register Details

Table 11-12: ADC Control 0 Register

ADC Control 0			**ADC_CTRL0**	**[0x0000]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:5	-	RO	0	**Reserved**
4	resetb	R/W	0	**Reset ADC** 0: ADC is in *ADC_RESET.* 1: Not in *ADC_RESET.*
3	chop_force	R/W	0	**Input Chopping** 0: Disabled. 1: Enabled.
2	skip_cal	R/W	0	**Skip Calibration** Set this field to 1 to skip automatic calibration before starting a conversion sequence. 0: Perform automatic calibration. 1: Skip automatic calibration.
1	bias_en	R/W	0	**Bias Enable** 0: Disabled. 1: Enabled.
0	adc_en	R/W	0	**ADC Enable** 0: Disabled. 1: Enabled.

Table 11-13: ADC Control 1 Register

ADC Control 1			**ADC_CTRL1**	**[0x0004]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:21	-	RO	0	**Reserved**
20:16	num_slots	R/W	0	**Number of Slots Enabled per Conversion Sequence** 0: 1 slot. 1: 2 slots. 2: 3 slots. … : … 30: 31 slots. 31: Reserved.
15:11	-	RO	0	**Reserved**
10:8	avg	R/W	0	**Sample Averaging** 0: No averaging All other values: Average 2^{avg} samples on each channel before reporting the results.
7	ts_sel	R/W	0	**Temperature Sensor Select** 0: Disabled. 1: Enabled.
6:4	trig_sel	R/W	0	**Hardware Trigger Source** See *Table 11-8* for field settings.
3	samp_ck_off	R/W	0	**Sample Clock Control** 0: Continuous sample clock. 1: Sample clock runs only while a channel is being measured.

ADC Control 1			ADC_CTRL1	[0x0004]
Bits	**Field**	**Access**	**Reset**	**Description**
2	cnv_mode	R/W	0	**Conversion Mode** 0: Single conversion sequence. 1: Continuous conversion sequence.
1	trig_mode	R/W	0	**Trigger Mode Control** 0: Software trigger. 1: Hardware trigger.
0	start	R/W	0	**Conversion Start** In software-triggered mode (*ADC_CTRL1*.*trig_mode* = 0), a conversion sequence starts immediately when this field is set to 1. After a sequence (*ADC_INTFL*.*seq_done* = 1) or when a conversion is complete (*ADC_INTFL*.*conv_done* = 1), software should set this field to 0. In hardware-triggered mode (*ADC_CTRL1*.*trig_mode* = 1), a conversion sequence is armed when this field is set to 1. A conversion sequence starts when the selected hardware trigger becomes active. Any time after the hardware triggered conversion is started (*ADC_INTFL*.*seq_started* = 1), software should set this field to 0 to prevent subsequent conversion sequences from starting if desired. See *Conversions* for details. 0: Conversion complete. 1: Start a conversion sequence or arm the ADC to start a hardware trigger.

Table 11-14: ADC Clock Control Register

ADC Clock Control			ADC_CLKCTRL	[0x0008]
Bits	**Field**	**Access**	**Reset**	**Description**
31:7	-	RO	0	**Reserved**
6:4	clkdiv	R/W	3	**Clock Divider** See *Clocks and Timing* for details on determining the required setting for this field. The maximum SAR_CLK frequency is 25MHz. 0: Divide by 2. 1: Divide by 4. 2: Divide by 8. 3: Divide by 16. 4-7: Divide by 1.
3:2	-	RO	0	**Reserved**
1:0	clksel	R/W1C	0	**Clock Source** This field selects the ADC peripheral clock. See *Table 11-3* for available clock sources.

Table 11-15: ADC Sample Clock Control Register

Sample Clock Control Register			ADC_SAMPCLKCTRL	[0x000C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:16	idle_cnt	R/W	0	**Sample Clock Hold Time** The number of cycles to add to the minimum hold time. See *Clocks and Timing* for details on determining the required setting for this field to achieve the desired sample rate.
15:8	-	RO	0	**Reserved**

Sample Clock Control Register			ADC_SAMPCLKCTRL	[0x000C]
Bits	**Field**	**Access**	**Reset**	**Description**
7:0	track_cnt	R/W	0	**Sample Clock Track Time** The number of cycles to add to the minimum track time. See *Clocks and Timing* for details on determining the required setting for this field to achieve the desired sample rate.

Table 11-16: ADC Channel Select 0 Register

ADC Channel Select 0			ADC_CHSEL0	[0x0010]
Bits	**Field**	**Access**	**Reset**	**Description**
31:29	-	RO	0	**Reserved**
28:24	slot3_id	R/W	0	**Slot 3 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
23:21	-	RO	0	**Reserved**
20:16	slot2_id	R/W	0	**Slot 2 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
15:13	-	RO	0	**Reserved**
12:8	slot1_id	R/W	0	**Slot 1 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
7:5	-	RO	0	**Reserved**
4:0	slot0_id	R/W	0	**Slot 0 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.

Table 11-17: ADC Channel Select 1 Register

ADC Channel Select 1			ADC_CHSEL1	[0x0014]
Bits	**Field**	**Access**	**Reset**	**Description**
31:29	-	RO	0	**Reserved**
28:24	slot7_id	R/W	0	**Slot 7 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
23:21	-	RO	0	**Reserved**
20:16	slot6_id	R/W	0	**Slot 6 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
15:13	-	RO	0	**Reserved**
12:8	slot5_id	R/W	0	**Slot 5 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
7:5	-	RO	0	**Reserved**
4:0	slot4_id	R/W	0	**Slot 4 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.

Table 11-18: ADC Channel Select 2 Register

ADC Channel Select 2			**ADC_CHSEL2**	**[0x0018]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:29	-	RO	0	**Reserved**
28:24	slot11_id	R/W	0	**Slot 11 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
23:21	-	RO	0	**Reserved**
20:16	slot10_id	R/W	0	**Slot 10 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
15:13	-	RO	0	**Reserved**
12:8	slot9_id	R/W	0	**Slot 9 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
7:5	-	RO	0	**Reserved**
4:0	slot8_id	R/W	0	**Slot 8 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.

Table 11-19: ADC Channel Select 3 Register

ADC Channel Select 3			**ADC_CHSEL3**	**[0x001C]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:29	-	RO	0	**Reserved**
28:24	slot15_id	R/W	0	**Slot 15 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
23:21	-	RO	0	**Reserved**
20:16	slot14_id	R/W	0	**Slot 14 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
15:13	-	RO	0	**Reserved**
12:8	slot13_id	R/W	0	**Slot 13 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
7:5	-	RO	0	**Reserved**
4:0	slot12_id	R/W	0	**Slot 12 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.

Table 11-20: ADC Channel Select 4 Register

ADC Channel Select 4			**ADC_CHSEL4**	**[0x0020]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:29	-	RO	0	**Reserved**
28:24	slot19_id	R/W	0	**Slot 19 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
23:21	-	RO	0	**Reserved**

ADC Channel Select 4			ADC_CHSEL4	[0x0020]
Bits	**Field**	**Access**	**Reset**	**Description**
20:16	slot18_id	R/W	0	**Slot 18 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
15:13	-	RO	0	**Reserved**
12:8	slot17_id	R/W	0	**Slot 17 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
7:5	-	RO	0	**Reserved**
4:0	slot16_id	R/W	0	**Slot 16 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.

Table 11-21: ADC Channel Select 5 Register

ADC Channel Select 5			ADC_CHSEL5	[0x0024]
Bits	**Field**	**Access**	**Reset**	**Description**
31:29	-	RO	0	**Reserved**
28:24	slot23_id	R/W	0	**Slot 23 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
23:21	-	RO	0	**Reserved**
20:16	slot22_id	R/W	0	**Slot 22 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
15:13	-	RO	0	**Reserved**
12:8	slot21_id	R/W	0	**Slot 21 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
7:5	-	RO	0	**Reserved**
4:0	slot20_id	R/W	0	**Slot 20 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.

Table 11-22: ADC Channel Select 6 Register

ADC Channel Select 6			ADC_CHSEL6	[0x0028]
Bits	**Field**	**Access**	**Reset**	**Description**
31:29	-	RO	0	**Reserved**
28:24	slot27_id	R/W	0	**Slot 27 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
23:21	-	RO	0	**Reserved**
20:16	slot26_id	R/W	0	**Slot 26 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
15:13	-	RO	0	**Reserved**
12:8	slot25_id	R/W	0	**Slot 25 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.

ADC Channel Select 6			ADC_CHSEL6	[0x0028]
Bits	**Field**	**Access**	**Reset**	**Description**
7:5	-	RO	0	**Reserved**
4:0	slot24_id	R/W	0	**Slot 24 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.

Table 11-23: ADC Channel Select 7 Register

ADC Channel Select 7			ADC_CHSEL7	[0x002C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:29	-	RO	0	**Reserved**
28:24	slot31_id	R/W	0	**Slot 31 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
23:21	-	RO	0	**Reserved**
20:16	slot30_id	R/W	0	**Slot 30 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
15:13	-	RO	0	**Reserved**
12:8	slot29_id	R/W	0	**Slot 29 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.
7:5	-	RO	0	**Reserved**
4:0	slot28_id	R/W	0	**Slot 28 Channel Assignment** Channel number assigned to the slot. Invalid channel numbers are ignored.

Table 11-24: ADC Restart Count Register

ADC Restart Count			ADC_RESTART	[0x002C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:16	-	RO	0	**Reserved**
15:0	cnt	R/W	0	**Sample Delay Before Continuous Conversion Restart** The number of SAMPLE_CLK periods to delay before restarting a continuous mode conversion sequence.

Table 11-25: ADC Data Format Register

ADC Data Format			ADC_DATAFMT	[0x003C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	mode	DNM	0xFFFF DFFF	**Channel Format** This field defines the data format of each channel. Each bit position corresponds to a specific channel number, i.e., *ADC_DATAFMT.mode[0]* is the data format for channel 0, *ADC_DATAFMT.mode[1]* is the data format for channel 1. Do not change this register from its default value. All channels operate in single-ended mode except the temperature sensor, which operates in differential mode. Bit positions corresponding to unimplemented channels are ignored. 0: Differential mode. 1: Single-ended mode.

Table 11-26: ADC FIFO and DMA Control Register

ADC FIFO and DMA Control			ADC_FIFODMACTRL	[0x0040]
Bits	**Field**	**Access**	**Reset**	**Description**
31:16	-	RO	0	**Reserved**
15:8	thresh	R/W	0	**FIFO and DMA Threshold** When the number of words in the FIFO exceeds the threshold, a DMA request is triggered and, the *ADC_INTFL.fifo_lvl* interrupt flag is set.
7:4	-	RO	0	**Reserved**
3:2	format	R/W	0	**FIFO Data Format** 0b00: Data and status (12 bits processed plus status fields). 0b01: Data only (12 bits processed). 0b10: Raw Data Only (18-bit raw data). 0b11: Reserved.
1	flush	R/W1O	0	**FIFO Flush** Write 1 to flush the FIFO. This bit always reads 0. 0: Normal operation. 1: Flush FIFO.
0	dma_en	R/W	0	**DMA Enable** 0: Disabled. 1: Enabled.

Table 11-27: ADC Data Register

ADC Data			ADC_DATA	[0x0044]
Bits	**Field**	**Access**	**Reset**	**Description**
31	clipped	R	1	**Clipped** This field is set if the ADC sample or samples was clipped.
30:25	-	RO	0	**Reserved**
24	invalid	R	1	**Invalid Flag** This field is set if the data is invalid, e.g., reading from an empty FIFO, reading an invalid channel, reading the temperature sensor when the temperature sensor is not ready.
23:21	0	RO	0	**Reserved**
20:16	chan	R	0x1F	**Channel Identifier** This field is the channel identifier associated with the *ADC_DATA.data* field.
15:0	data	R/W	0xFFFF	**Data** The format of this data is configurable. See *Conversion Results* for details.

Table 11-28: ADC Status Register

ADC Status			ADC_STATUS	[0x0048]
Bits	**Field**	**Access**	**Reset**	**Description**
31:16	-	RO	0	**Reserved**

ADC Status			**ADC_STATUS**	**[0x0048]**
Bits	**Field**	**Access**	**Reset**	**Description**
15:8	fifo_level	R	0	**FIFO Level** This field returns the number of words available to read from the FIFO. *Note: Valid values of this field are 0 to 16.*
7:3	-	RO	0	**Reserved**
2	full	R	0	**FIFO Full** 0: FIFO not full 1: FIFO full
1	empty	R	1	**FIFO Empty** 0: FIFO not empty 1: FIFO empty
0	ready	R	0	**ADC Ready** 0: ADC is not in *ADC_ON* state. 1: ADC is in *ADC_ON* state.

Table 11-29: ADC Channel Status Register

ADC Channel Status			**ADC_CHSTATUS**	**[0x004C]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:16	-	RO	0	**Reserved**
15:0	clipped	W1C	0	**Clipped Data** This register identifies channels that have experienced a conversion result that was clipped. Each bit position corresponds to a specific channel number, i.e., clipped[0] is the clipped status for channel 0, clipped[1] is the clipped status for channel 1, clipped[14] is the clipped status for channel 14. Once a clipped conversion occurs, the bit remains set until cleared by software. 0: Not clipped 1: Clipped

Table 11-30: ADC Interrupt Enable Register

ADC Interrupt Enable			**ADC_INTEN**	**[0x0050]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:11	-	RO	0	**Reserved**
10	fifo_ofl	W1C	0	**FIFO Overflow Event Interrupt Enable** 0: Disabled 1: Enabled
9	fifo_ufl	W1C	0	**FIFO Underflow Event Interrupt Enable** 0: Disabled 1: Enabled
8	fifo_lvl	W1C	0	**FIFO Level Event Interrupt Enable** 0: Disabled 1: Enabled
7	clipped	W1C	0	**Data Clipped Event Interrupt Enable** 0: Disabled 1: Enabled

ADC Interrupt Enable			ADC_INTEN	[0x0050]
Bits	**Field**	**Access**	**Reset**	**Description**
6	conv_done	W1C	0	**Conversion Done Event Interrupt Enable** 0: Disabled 1: Enabled
5	seq_done	W1C	0	**Sequence Done Event Interrupt Enable** 0: Disabled 1: Enabled
4	seq_started	W1C	0	**Sequence Started Event Interrupt Enable** 0: Disabled 1: Enabled
3	start_det	W1C	0	**Command Start Event Interrupt Enable** 0: Disabled 1: Enabled
2	abort	W1C	0	**Command Aborted Event Interrupt Enable** 0: Disabled 1: Enabled
1	-	RO	0	**Reserved**
0	ready	W1C	0	**ADC Ready Event Interrupt Enable** 0: Disabled 1: Enabled

Table 11-31: ADC Interrupt Flags Register

ADC Interrupt Flags			ADC_INTFL	[0x0054]
Bits	**Field**	**Access**	**Reset**	**Description**
31:11	-	RO	0	**Reserved**
10	fifo_ofl	R/W	0	**FIFO Overflow Event** 0: Normal operation 1: Event occurred
9	fifo_ufl	R/W	0	**FIFO Underflow Event** 0: Normal operation 1: Event occurred
8	fifo_lvl	R/W	0	**FIFO Level Event** 0: Normal operation 1: Event occurred
7	clipped	R/W	0	**Data Clipped Event** 0: Normal operation 1: Event occurred
6	conv_done	R/W	0	**Conversion Done Event** 0: Normal operation 1: Event occurred
5	seq_done	R/W	0	**Sequence Done Event** 0: Normal operation 1: Event occurred
4	seq_started	R/W	0	**Sequence Started Event** 0: Normal operation 1: Event occurred

ADC Interrupt Flags			ADC_INTFL	[0x0054]
Bits	**Field**	**Access**	**Reset**	**Description**
3	start_det	R/W	0	**Command Start Event** The conversion command was started. 0: Normal operation 1: Event occurred
2	abort	R/W	0	**Command Aborted Event** The conversion command was aborted before conversions were complete. 0: Normal operation 1: Event occurred
1	-	RO	0	**Reserved**
0	ready	R/W	0	**ADC Ready Event** 0: Normal operation 1: Event occurred

Table 11-32: ADC SFR Address Offset Register

ADC SFR Address Offset			ADC_SFRADDROFFSET	[0x0060]
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
7:0	offset	R/W	0	**Base Address Offset for SFR Registers** See *ADC SFR Interface* for details.

Table 11-33: ADC SFR Address Register

ADC SFR Address			ADC_SFRADDR	[0x0064]
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
7:0	addr	R/W	0	**SFR Configuration Address** See *ADC SFR Interface* for details.

Table 11-34: ADC SFR Write Data Register

ADC SFR Write Data			ADC_SFRWRDATA	[0x0068]
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
7:0	data	R/W	0	**SFR Write Data** See *ADC SFR Interface* for details.

Table 11-35: ADC SFR Read Data Register

ADC SFR Read Data			ADC_SFRRDDATA	[0x006C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
7:0	data	R/W	0	**SFR Read Data** See *ADC SFR Interface* for details.

Table 11-36: ADC SFR Status Register

ADC SFR Status			ADC_SFRSTATUS	[0x0070]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
0	nack	R	0	**Last SFR Transaction Status** This field indicates if an error occurred during the last SFR transaction. It is cleared at the start of a write to *ADC_SFRADDR*, *ADC_SFRWRDATA*, or the start of a read of the *ADC_SFRRDDATA* register. 0: No error 1: SFR transaction error

11.12 Low-Power Comparator Registers

See *Table 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 11-37: Low-Power Comparator Registers Summary

Offset	Name	Description
[0x0000]	*LPCMP[0]*	Low-Power Comparator 1 Register
[0x0004]	*LPCMP[1]*	Low-Power Comparator 2 Register
[0x0008]	*LPCMP[2]*	Low-Power Comparator 3 Register

11.12.1 Low-Power Comparator Register Details

Table 11-38: Low-Power Comparator n Registers

Low-Power Comparator 1	**LPCMP[0]**	**[0x0000]**
Low-Power Comparator 2	**LPCMP[1]**	**[0x0004]**
Low-Power Comparator 3	**LPCMP[2]**	**[0x0008]**

Bits	Field	Access	Reset	Description
31:16	-	RO	0	**Reserved**
15	if	R/W1C	0	**Low-Power Comparator Interrupt Flag** This field is set to 1 by hardware when the comparator output changes to the active state, as set using the *pol* field. Write 1 to clear this flag. 0: No interrupt 1: Interrupt occurred
14	out	RO	*	**Low-Power Comparator Output** This field is the comparator's output state. 0: Output low. 1: Output high.
13:7	-	RO	0	**Reserved**
6	int_en	R/W	0	**Low-Power Comparator Interrupt Enable** Set this field to 1 to enable the interrupt for the low-power comparator. 0: Disabled 1: Enabled
5	pol	R/W	0	**Comparator 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**
0	en	R/W	0	**Low-Power Comparator Enable** Set this field to 1 to enable the comparator 0: Disabled 1: Enable

12. UART (UART)

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 is independently configurable unless using a shared external clock source.

The LPUART is a special version of the peripheral that can receive characters at up to 9600 baud while in low-power modes. The hardware loads valid received characters into the receive FIFO and wakes the device when an enabled interrupt condition occurs.

The peripheral provides the following features:

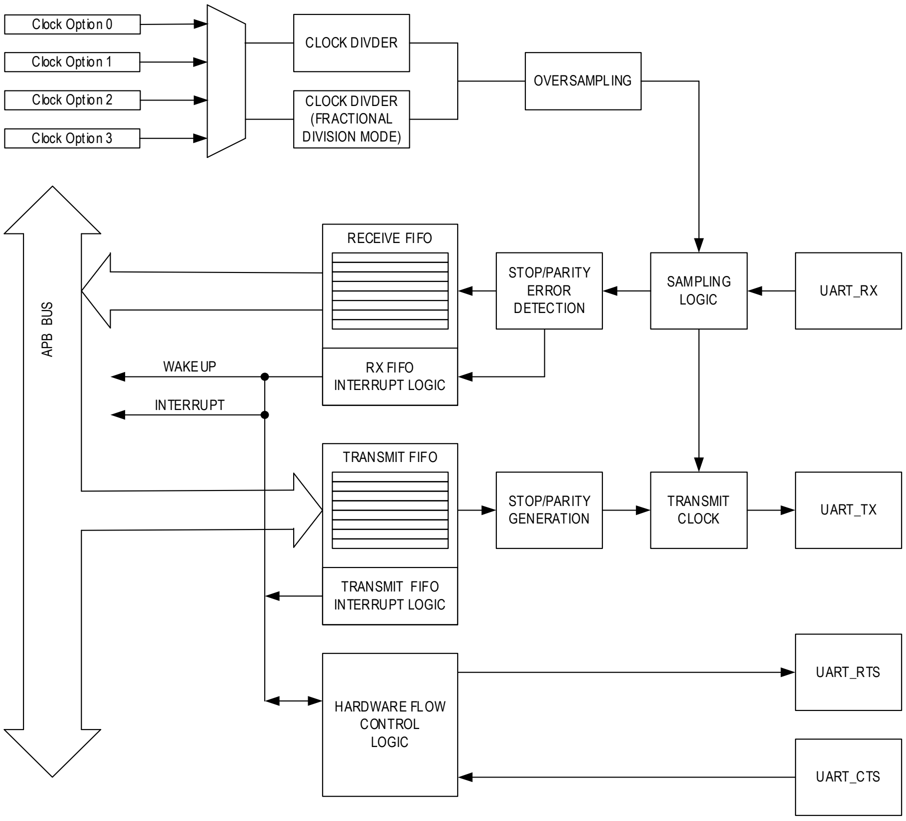
- Flexible baud rate generation for standard UART instances
- Programmable character size of 5-bits to 8-bits
- Stop bit settings of 1, 1.5, or 2-bits
- Parity settings of even, odd, mark (always 1), space (always 0), and no parity
- Automatic parity error detection with selectable parity bias
- Automatic frame error detection
- Separate 8-byte transmit and receive FIFOs
- Flexible interrupt conditions
- Hardware flow control (HFC) using ready-to-send (RTS) and clear-to-send (CTS) pins
- Separate DMA channels for transmit and receive
    - DMA support is available in *ACTIVE* and *SLEEP*

The LPUART instance provides these additional features:

- Baud rate support for up to 1.85Mbps in *ACTIVE*
- Receive characters in *SLEEP*, *DEEPSLEEP*, and *BACKUP* at up to 9600 baud
- Fractional baud rate divisor improves baud rate accuracy for 9600 and lower baud rates
- Wakeup from low-power modes to *ACTIVE* on multiple receive FIFO conditions

Figure 12-1 shows a high-level diagram of the UART peripheral.

Figure 12-1: UART Block Diagram



Note: See Table 12-1 for the clock options supported by each UART instance.

12.1 Instances

Instances of the peripheral are shown in *Table 12-1*. The standard UARTs and the LPUARTs are functionally similar; they are referred to as UART for common functionality. The LPUART instance supports fractional division mode (FDM) and is referenced as LPUART for feature-specific options.

Table 12-1: MAX78002 UART/LPUART Instances

Instance	Register Access Name	LPUART	Power Modes	Clock Option				HFC	Transmit FIFO Depth	Receive FIFO Depth
				0	1	2	3			
UART0	UART0	No	*ACTIVE* *SLEEP*	PCLK	-	IBRO	-	Yes	8	8
UART1	UART1	No	*ACTIVE* *SLEEP*	PCLK	-	IBRO	-	No		
UART2	UART2									
LPUART0	UART3	Yes	*ACTIVE* *SLEEP* *DEEPSLEEP* *BACKUP*	IBRO	ERTCO	-	-	No		

12.2 DMA

Each UART instance supports DMA for both transmit and receive; separate DMA channels can be connected to the receive and transmit FIFOs.

The UART DMA channels are configured using the UART DMA configuration register, *UARTn_DMA*. Enable the receive FIFO DMA channel by setting *UARTn_DMA.rx_en* to 1 and enable the transmit FIFO DMA channel by setting *UARTn_DMA.tx_en* to 1. DMA transfers are automatically triggered by the hardware based on the number of bytes in the receive FIFO and transmit FIFO.

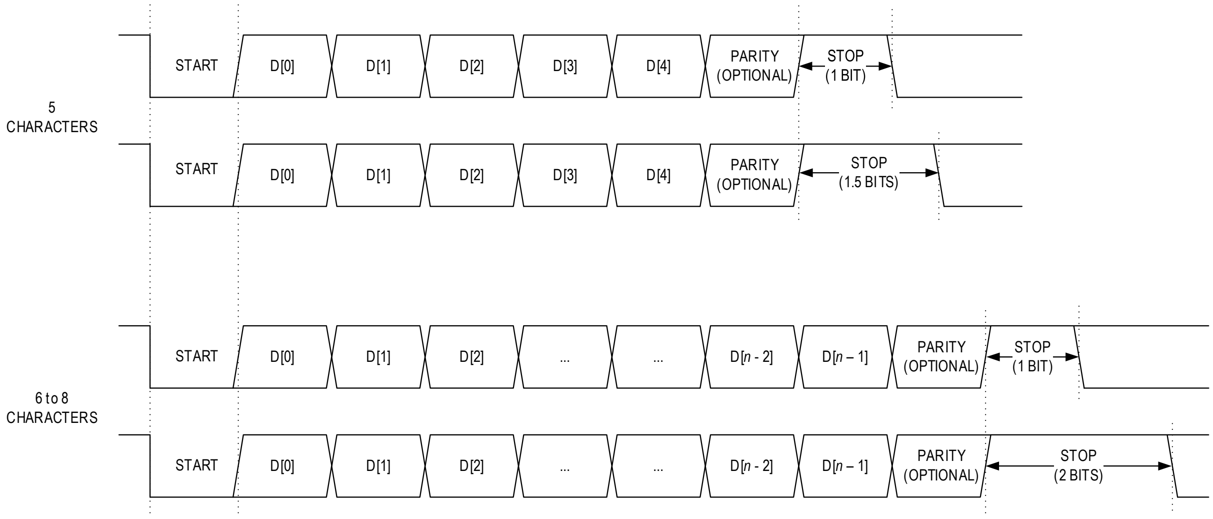
When DMA is enabled, the following describes the behavior of the DMA requests:

- A receive DMA request is asserted when the number of bytes in the receive FIFO transitions to be greater than or equal to the receive FIFO threshold.
- A transmit DMA request is asserted when the number of bytes in the transmit FIFO transitions to be less than the transmit FIFO threshold.

12.3 UART Frame

Figure 12-2 shows the UART frame structure. Character sizes of 5 to 8 bits are configurable through the *UARTn_CTRL.char_size* field. Stop bits are configurable 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, and none.

Figure 12-2: UART Frame Structure



12.4 FIFOs

Separate receive and transmit FIFOs are provided. The FIFOs are both accessed through the same *UARTn_FIFO.data* field. The current level of the transmit FIFO is read from *UARTn_STATUS.tx_lvl*, and the receive FIFO current level is read from *UARTn_STATUS.rx_lvl*. Data for character sizes less than 7 bits are right justified.

12.4.1 TX FIFO Operation

Writing data to the *UARTn_FIFO.data* field increments the TX FIFO pointer, *UARTn_STATUS.tx_lvl*, and loads the data into the TX FIFO. The *UARTn_TXPEEK.data* register provides a feature that allows the software to "peek" at the current value of the write-only TX FIFO without changing the *UARTn_STATUS.tx_lvl*. Writes to the TX FIFO are ignored while *UARTn_STATUS.tx_lvl* = C_TX_FIFO_DEPTH.

12.4.2 RX FIFO Operation

Reads of the *UARTn_FIFO.data* field return the character values in the RX FIFO and decrement the *UARTn_STATUS.rx_lvl*. An overrun event occurs if a valid frame, including parity, is detected while *UARTn_STATUS.rx_lvl* = C_RX_FIFO_DEPTH. When an overrun event occurs, the data is discarded by hardware.

A parity error event indicates that the value read from *UARTn_FIFO.data* contains a parity error.

12.4.3 Flushing

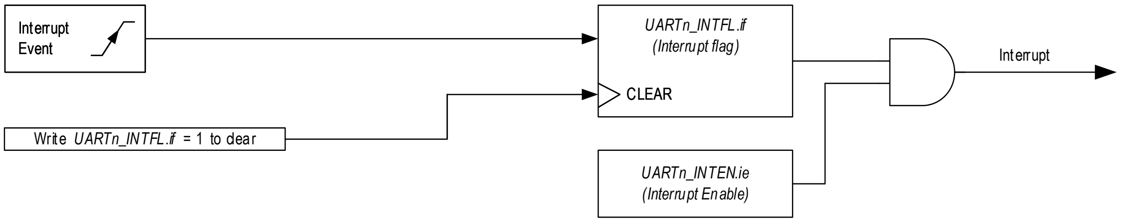
The FIFOs are flushed on the following conditions:

- Setting the *UARTn_CTRL.rx_flush* field to 1 flushes the RX FIFO by setting its pointer to 0.
- Setting the *UARTn_CTRL.tx_flush* field to 1 flushes the TX FIFO by setting its pointer to 0.
- Flush the FIFOs by setting the respective UART's reset field (*GCR_RST0*) to 1.

12.5 Interrupt Events

The peripheral generates interrupts for the events shown in *Table 12-2*. Unless noted otherwise, each instance has its own set of interrupts and higher-level flag and enable fields, as shown in *Table 12-2*

Figure 12-3: UART Interrupt Functional Diagram



Some activity can set one or more event flags and cause more than one event. An event interrupt occurs if the corresponding interrupt enable is set. The interrupt flags, when set, must be cleared by the software by writing 1 to the corresponding interrupt flag field.

Table 12-2: MAX78002 Interrupt Events

Event	Interrupt Flag	Interrupt Enable
Frame Error	*UARTn_INT_FL.rx_ferr*	*UARTn_INT_EN.rx_ferr*
Parity Error	*UARTn_INT_FL.rx_par*	*UARTn_INT_EN.rx_par*
CTS Signal Change	*UARTn_INT_FL.cts_ev*	*UARTn_INT_EN.cts_ev*
Receive FIFO Overrun	*UARTn_INT_FL.rx_ov*	*UARTn_INT_EN.rx_ov*
Receive FIFO Threshold	*UARTn_INT_FL.rx_thd*	*UARTn_INT_EN.rx_thd*
Transmit FIFO Half-Empty	*UARTn_INT_FL.tx_he*	*UARTn_INT_EN.tx_he*

12.5.1 Frame Error

A frame error is generated when the UART sampling circuitry detects an invalid bit. Each bit is sampled three times, as shown in *Figure 12-4*, and can generate a frame error on the start bit, stop bit, data bits, and optionally the parity bit. When a frame error occurs, the data is discarded.

The frame error criteria are different based on the following:

- Standard UART and LPUART with FDM disabled
  - The start bit is sampled 3 times, and all samples must be 0, or a frame error is generated.
  - Each data bit is sampled, and 2 of the 3 samples must match, or a frame error is generated.
  - If parity is enabled, the parity bit is sampled 3 times, and all samples must match, or a frame error is generated.
  - The stop bit is sampled 3 times, and all samples must be 1, or a frame error is generated.
  - See *Table 12-3* for details
- LPUART with FDM enabled (*UARTn_CTRL.fdm* = 1) and data/parity edge detect enabled (*UARTn_CTRL.dpfe_en* = 1).
  - The start bit is sampled 3 times, and all samples must be 0, or a frame error is generated.
  - Each data bit is sampled 3 times, and all samples must match, or a frame error is generated.
  - If parity is enabled, the parity bit is sampled 3 times, and all samples must match, or a frame error is generated.
  - The stop bit is sampled 3 times, and all samples must be 1, or a frame error is generated.
  - See *Table 12-4* for details.

Table 12-3: Frame Error Detection for Standard UARTs and LPUART

UARTn_CTRL .par_en	*UARTn_CTRL .par_md*	*UARTn_CTRL .par_eo*	Start Samples	Data Samples	Parity Samples	Stop Samples
0	N/A	N/A	3 of 3 must be 0	2/3 must match	Not Present	3 of 3 must be 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"	

Table 12-4: Frame Error Detection for LPUARTs with UARTn_CTRL.fdm = 1 and UARTn_CTRL.dpfe_en = 1

UARTn_CTRL .par_en	*UARTn_CTRL .par_md*	*UARTn_CTRL .par_eo*	Start Samples	Data Samples	Parity Samples	Stop Samples
0	N/A	N/A	3 of 3 must be 0	3 of 3 must match	Not Present	3 of 3 must be 1
1	0	0			3 of 3 = 1 if even number of 1s 3 of 3 = 0 if odd number 0s	
	0	1			3 of 3 = 1 if odd number 1s 3 of 3 = 0 if even number 0s	
	1	0			3 of 3 = 1 if even number 0s 3 of 3 = 0 if odd number 1s	
	1	1			3 of 3 = 1 if odd number 0s 3 of 3 = 0 if even number 1s	

12.5.2 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. The data received is saved to the receive FIFO when a parity error occurs.

12.5.3 CTS Signal Change

A CTS signal change condition occurs if HFC is enabled, the UART baud clock is enabled, and the CTS pin changes state.

12.5.4 Overrun

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

12.5.5 Receive FIFO Threshold

A receive FIFO threshold event occurs when a valid frame is received that causes the number of bytes to exceed the configured receive FIFO threshold UARTn_CTRL.*rx_thd_val*.

12.5.6 Transmit FIFO Half-Empty

The transmit FIFO half-empty event occurs when UARTn_STATUS.*tx_lvl* transitions from more than half-full to half-empty, as shown in Equation 12-1.

Note: When this condition occurs, verify the number of bytes in the transmit FIFO (UARTn_STATUS.tx_lvl) before refilling.

Equation 12-1: UART Transmit FIFO Half-Empty Condition

$$\left(\frac{C\_TX\_FIFO\_DEPTH}{2}+1\right)\xrightarrow{Transistions\ from}\left(\frac{C\_TX\_FIFO\_DEPTH}{2}\right)$$

12.6 LPUART Wakeup Events

LPUART instances can receive characters while in the low-power modes listed in Table 12-1. If enabled, each of the receive FIFO conditions shown in Table 12-5 wakes the device, exits the low-power mode, and returns the device to *ACTIVE*.

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

Table 12-5: MAX78002 Wakeup Events

Receive FIFO Condition	Wakeup Flag UARTn_WKFL	Wakeup Enable UARTn_WKEN	Low-Power Peripheral Wakeup Enable
Threshold	*rx_thd*	*rx_thd*	PWRSEQ_LPPWEN.*uart3*
Full	*rx_full*	*rx_full*	
Not Empty	*rx_ne*	*rx_ne*	

12.6.1 Receive FIFO Threshold

This condition persists while UARTn_STATUS.*rx_lvl* ≥ UARTn_CTRL.*rx_thd_val*.

12.6.2 Receive FIFO Full

This condition persists while UARTn_STATUS.*rx_lvl* ≥ C_RX_FIFO_DEPTH.

12.6.3 Receive Not Empty

This condition persists while UARTn_STATUS.*rx_lvl* > 0.

12.7 Inactive State

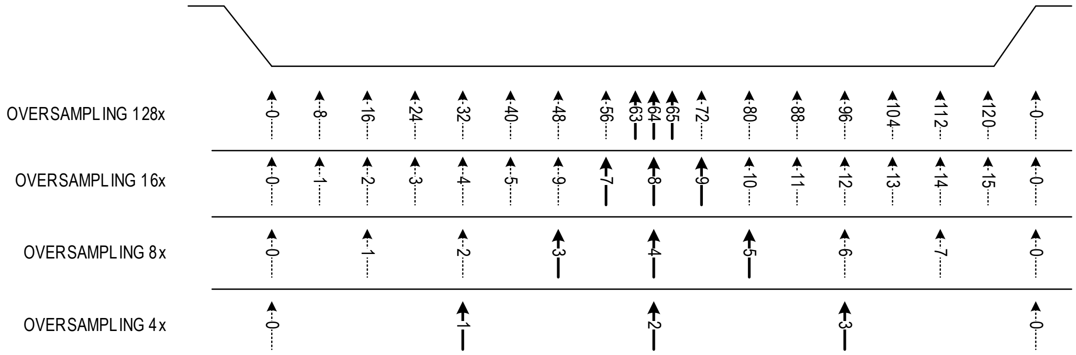
The following conditions result in the UART being inactive:

- When *UARTn_CTRL.bclken* = 0
- After setting *UARTn_CTRL.bclken* to 1 until *UARTn_CTRL.bclkrdy* = 1
- Any write to the *UARTn_CLKDIV.clkdiv field* while *UARTn_CTRL.bclken* = 1
- Any write to the *UARTn_OSR.osr* field when *UARTn_CTRL.bclken* = 1

12.8 Receive 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 is evaluated based on three samples at the midpoint of each bit time, as shown in *Figure 12-4*.

Figure 12-4: Oversampling Example



Whenever *UARTn_CLKDIV.clkdiv* < 0x10 (i.e., division rate less than 8.0), OSR is not used, and the oversampling rate is adjusted to full sampling by the hardware. In full sampling, the receive input is sampled on every clock cycle regardless of the OSR setting.

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

12.9 Baud Rate Generation

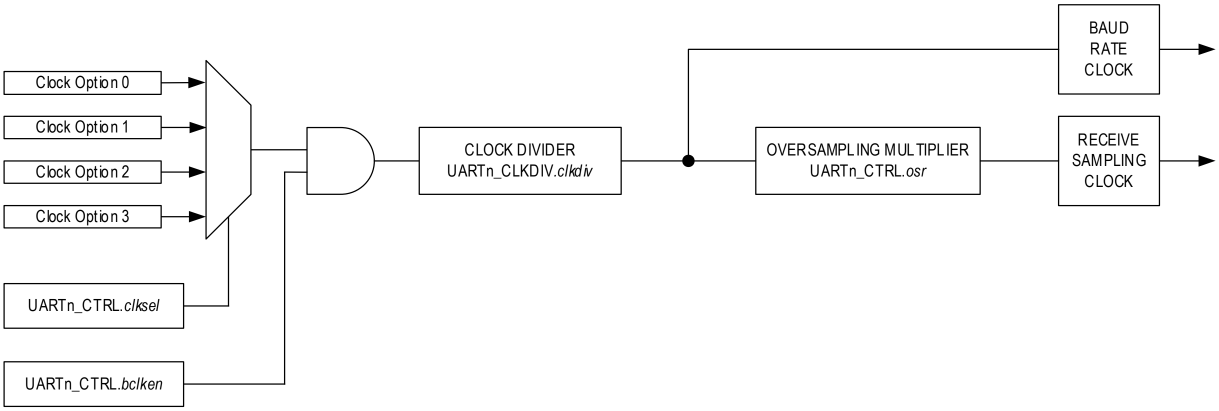
The baud rate is determined by the selected UART clock source and the value of the clock divisor. Multiple clock sources are available for each UART instance. See *Table 12-1* for available clock sources.

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

12.9.1 UART Clock Sources

Standard UART instances operate only in *ACTIVE* and *SLEEP*. Standard UART instances can only wake the device from *SLEEP*. *Figure 12-5* shows the baud rate generation path for standard UARTs.

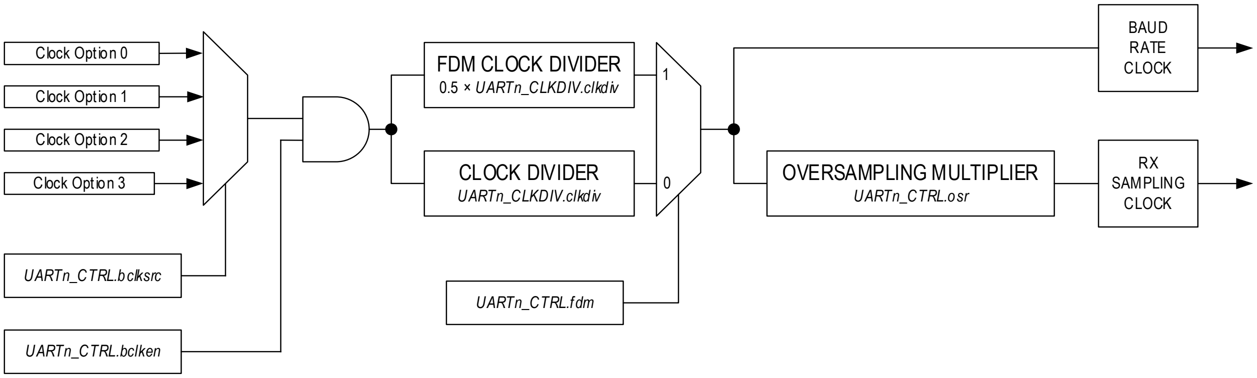
Figure 12-5: UART Baud Rate Generation



12.9.2 LPUART Clock Sources

LPUART instances support FDM and are configurable for operation at 9600 and lower baud rates for operation in *SLEEP*, *DEEPSLEEP*, and *BACKUP*. Operation in *DEEPSLEEP* and *BACKUP* require the use of the *ERTCO* as the baud rate clock source. The *ERTCO* can be configured to remain active in *DEEPSLEEP* and *BACKUP*, allowing the LPUART to receive data and serve as a wakeup source while power consumption is minimized.

Figure 12-6: LPUART Timing Generation



12.9.3 Baud Rate Calculation

The transmit and receive circuits share a common baud rate clock: 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. The FDM allows for greater accuracy when operating at low baud rates and finer granularity for the oversampling rate.

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

Equation 12-2: UART Clock Divisor Formula

$$UARTn\_CTRL.fdm = 0:$$

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

Equation 12-3: LPUART Clock Divisor Formula for UARTn_CTRL.fdm = 1

$$UARTn\_CTRL.fdm = 1:$$

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

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

- When UARTn_CTRL.*fdm* = 0, $UARTn\_CLKDIV.clkdiv = \left(\frac{50{,}000000}{115{,}200}\right) = 434$
- When UARTn_CTRL.*fdm* = 1, $UARTn\_CLKDIV.clkdiv = \left(\frac{50{,}000{,}000}{115{,}200}\right) \times 2 = 434.03 \times 2 = 868$

12.9.4 Low-Power Mode Operation of LPUARTs for 9600 Baud and Below

LPUART instances have the option to configure the receiver for 9600 and lower baud rates and enable the LPUART in the low-power modes *SLEEP*, *DEEPSLEEP*, and *BACKUP*. Receipt of a valid frame loads the receive FIFO and increments *UARTn_STATUS.rx_lvl*. If a wakeup event, shown in *Table 12-5*, is enabled, the device exits the current low-power mode and returns to *ACTIVE*. See *Baud Rate Calculation* and *Equation 12-3* for details on setting the baud rate for LPUART instances with *UARTn_CTRL.fdm* set to 1.

Table 12-6: LPUART Low Baud Rate Generation Examples (UARTn_CTRL.fdm = 1)

Clock Source	BAUD (bits/s)	Ratio (Clock/BAUD)	Calculated *UARTn_CLKDIV.clkdiv*	Error	*UARTn_OSR.osr*
ERTCO	9,600	3.413	7	-2.5%	N/A (1×)
	7,200	4.551	9	+1.1%	N/A (1×)
	4,800	6.827	14	-2.5%	N/A (1×)
	2,400	13.653	27	+1.1%	0: 8× 1: 12×
	1,800	18.204	36	+1.1%	0: 8× 1: 12× 2: 16×
	1,200	27.307	54	+1.1%	0: 8× 1: 12× 2: 16× 3: 20× 4: 24×

12.9.4.1 Configuring an LPUART for Low-Power Modes of Operation

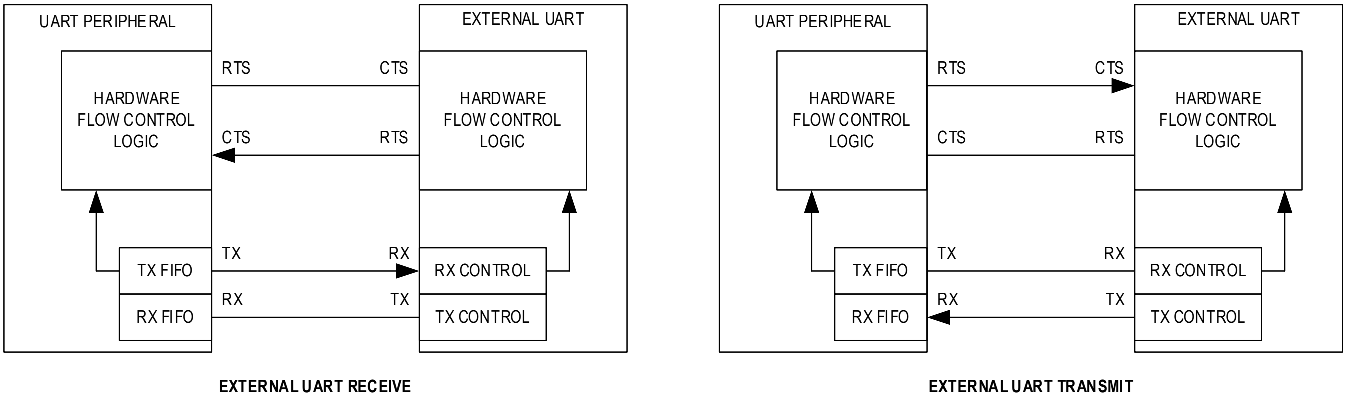
Use the following procedure to receive characters at 9600 or lower baud rates while in low-power modes:

1. Clear *UARTn_CTRL.bclken* = 0 to disable the baud clock. The hardware immediately clears *UARTn_CTRL.bclkrdy* to 0.
2. Ensure *UARTn_CTRL.ucagm* = 1.
3. Configure *UARTn_CTRL.bclksrc* to select the *ERTCO*.
4. Set *UARTn_CTRL.fdm* to 1 to enable FDM.
5. Set *UARTn_CLKDIV.clkdiv* to the calculated clock divisor shown in *Table 12-6* for the required baud rate.
6. Set *UARTn_CTRL.desm* to 1 to enable dual-edge sampling receive mode.
7. Choose the desired wakeup conditions from *Table 12-5*.
   a. Clear any of the wakeup conditions chosen if currently active in the *UARTn_WKFL* register.
   b. Enable the wakeup condition; set the wakeup field to 1 in the *UARTn_WKEN* register.
8. Set the *UARTn_CTRL.bclken* field to 1 to enable the baud clock.
9. Poll the *UARTn_CTRL.bclkrdy* field until it reads 1.
10. Enter the desired low-power mode.

12.10 Hardware Flow Control

The optional HFC uses two additional pins, CTS and RTS, 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 12-7*.

Figure 12-7: HFC Physical Connection



In HFC operation, a UART transmitter waits for the external device to assert its CTS pin. When CTS is asserted, the UART transmitter sends data to the external device. The external device keeps CTS asserted until it is unable to receive additional data, typically because the external device's receive FIFO is full. The external device then deasserts CTS until the device can receive more data. The external device then asserts CTS again, allowing additional data to be sent.

HFC can be fully automated by the peripheral hardware or by software through direct monitoring of the CTS input signal and control of the RTS output signal.

12.10.1 Automated HFC

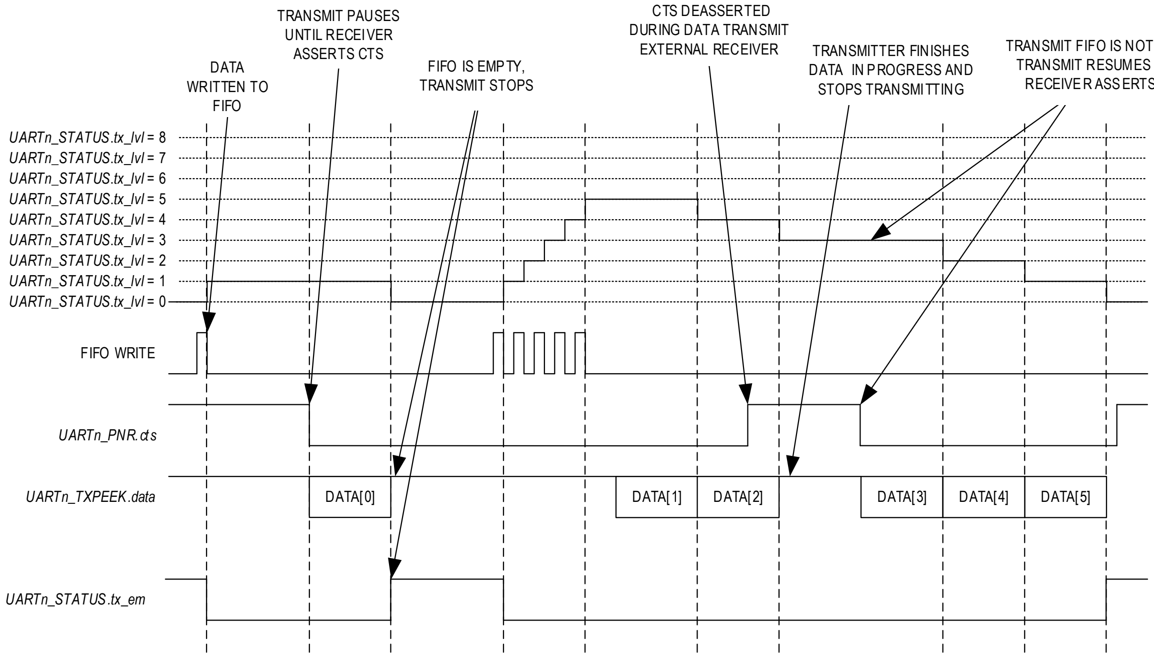
Setting *UARTn_CTRL.hfc_en* = 1 enables automated HFC. When automated HFC is enabled, the hardware manages the CTS and RTS signals. The deassertion of the RTS signal is configurable using the *UARTn_CTRL.rtsdc* field:

- *UARTn_CTRL.rtsdc* = 0: Deassert RTS when *UARTn_STATUS.rx_lvl* = C_RX_FIFO_DEPTH
- *UARTn_CTRL.rtsdc* = 1: Deassert RTS while *UARTn_STATUS.rx_lvl* ≥= *UARTn_CTRL.rx_thd_val*

The transmitter continues to send data as long as the CTS signal is asserted and there is data in the transmit FIFO. If the receiver deasserts the CTS pin, the transmitter finishes transmitting the current character and then waits until the CTS pin state is asserted before continuing transmission. *Figure 12-8* shows the state of the CTS pin during a transmission under automated HFC.

Automated HFC does not generate interrupt events related to the state of the transmit FIFO or the receive FIFO. The software must handle FIFO management. See *Interrupt Events* for additional information.

Figure 12-8: HFC Signaling for Transmitting to an External Receiver



12.10.2 Software Controlled HFC

Software controlled HFC requires the software to manually control the RTS output pin and monitor the CTS input pin. Using software controlled HFC requires the automated HFC to be disabled by setting the *UARTn_CTRL.hfc_en* field to 1. Additionally, the software should enable CTS sampling (*UARTn_CTRL.cts_dis* = 0) if performing software controlled HFC.

12.10.2.1 RTC/CTS Handling for Application Controlled HFC

The software can manually monitor the CTS pin state by reading the field *UARTn_PNR.cts*. The software can manually set the state of the RTS output pin and read the current state of the RTS output pin using the field *UARTn_PNR.rts*. The software must manage the state of the RTS pin when performing software controlled HFC.

Interrupt support for CTS input signal change events is supported even when automated HFC is disabled. Software can enable the CTS interrupt event by setting the *UARTn_INT_EN.cts_ev* field to 1. The CTS signal change interrupt flag is set by the hardware any time the CTS pin state changes. The software must clear this interrupt flag manually by writing 1 to the *UARTn_INT_FL*.*cts_ev* field.

Note: CTS pin state monitoring is disabled any time the UART baud clock is disabled (UARTn_CTRL.bclken = 0). The software must enable CTS pin monitoring by setting the field UARTn_CTRL.cts_dis to 0 after enabling the baud clock if CTS pin state monitoring is required.

12.11 Registers

See *Table* 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 12-7*. Register names for a specific instance are defined by replacing "n" with the instance number. For example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See *Table 1-1* 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 12-7: UART/LPUART Register Summary

Offset	Register	Name
[0x0000]	*UARTn_CTRL*	UART Control Register
[0x0004]	*UARTn_STATUS*	UART Status Register
[0x0008]	*UARTn_INT_EN*	UART Interrupt Enable Register
[0x000C]	*UARTn_INT_FL*	UART Interrupt Flag Register
[0x0010]	*UARTn_CLKDIV*	UART Clock Divisor Register
[0x0014]	*UARTn_OSR*	UART Oversampling Control Register
[0x0018]	*UARTn_TXPEEK*	UART Transmit FIFO
[0x001C]	*UARTn_PNR*	UART Pin Control Register
[0x0020]	*UARTn_FIFO*	UART FIFO Data Register
[0x0030]	*UARTn_DMA*	UART DMA Control Register
[0x0034]	*UARTn_WKEN*	UART Wakeup Interrupt Enable Register
[0x0038]	*UARTn_WKFL*	UART Wakeup Interrupt Flag Register

12.11.1 Register Details

Table 12-8: UART Control Register

UART Control				**UARTn_CTRL**	**[0x0000]**
Bits	**Field**	**Access**	**Reset**	**Description**	
31:23	-	DNM	0	**Reserved**	
22	desm	R/W	0	**Receive Dual Edge Sampling Mode** LPUART instances only. This field is reserved in standard UART instances. 0: Sample receive input signal on clock rising edge only. 1: Sample receive input signal on both rising and falling edges.	
21	fdm	R/W	0	**Fractional Division Mode** LPUART instances only. This field is reserved in standard UART instances. 0: Baud rate divisor is an integer. 1: Baud rate divisor supports 0.5 division resolution.	

UART Control				**UARTn_CTRL** **[0x0000]**
Bits	**Field**	**Access**	**Reset**	**Description**
20	ucagm	R/W	0	**UART Clock Auto Gating Mode** *Note: Software must set this field to 1 for proper operation.* 0: No gating. 1: UART clock is paused during transmit and receive idle states.
19	bclkrdy	R	0	**Baud Clock Ready** 0: Baud clock not ready. 1: Baud clock ready.
18	dpfe_en	R/W	0	**Data/Parity Bit Frame Error Detection Enable** LPUART instances only. This field is reserved in standard UART instances. 0: Disable. Do not detect receive frame errors between the start bit and stop bit. 1: Enable. Detect frame errors when receive changes at the center of a bit time.
17:16	bclksrc	R/W	0	**Baud Clock Source** This field selects the baud clock source. See *Table 12-1* for available clock options for each UART instance. 0: Clock option 0. 1: Clock option 1. 2: Clock option 2. 3: Clock option 3.
15	bclken	R/W	0	**Baud Clock Enable** 0: Disabled. 1: Enabled.
14	rtsdc	R	0	**HFC RTS Deassert Condition** 0: Deassert RTS when the receive FIFO Level = C_RX_FIFO_DEPTH (FIFO full). 1: Deassert RTS while the receive FIFO Level >= *UARTn_CTRL.rx_thd_val*.
13	hfc_en	R/W	0	**HFC 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	R/W1O	0	**Receive FIFO Flush** Write 1 to flush the receive FIFO. This bit always reads 0. 0: Normal operation. 1: Flush FIFO.
8	tx_flush	R/W1O	0	**Transmit FIFO Flush** Write 1 to flush the transmit FIFO. This bit always reads 0. 0: Normal operation. 1: Flush FIFO.
7	cts_dis	R/W	1	**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).

UART Control				UARTn_CTRL [0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**
5	par_eo	R/W	0	**Parity Odd/Even Select** 0: Even parity. 1: Odd parity.
4	par_en	R/W	0	**Transmit Parity Generation Enable** 0: Parity transmission disabled. 1: Parity bit is calculated and transmitted after the last character bit.
3:0	rx_thd_val	R/W	0	**Receive FIFO Threshold** Valid settings are from 1 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

Table 12-9: UART Status Register

UART Status				UARTn_STATUS [0x0004]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	RO	0	**Reserved**
15:12	tx_lvl	R	0	**Transmit FIFO Level** This field is the number of characters in the transmit FIFO. 0 - 8: Number of bytes in the transmit FIFO. 9 - 15: Reserved
11:8	rx_lvl	R	0	**Receive FIFO Level** This field is the number of characters in the receive FIFO. 0 - 8: Number of bytes in the receive FIFO 9 - 15: Reserved
7	tx_full	R	0	**Transmit FIFO Full** 0: Not full 1: Full
6	tx_em	R	1	**Transmit FIFO Empty** 0: Not empty 1: Empty
5	rx_full	R	0	**Receive FIFO Full** 0: Not full 1: Full
4	rx_em	R	1	**Receive FIFO Empty** 0: Not empty 1: Empty
3:2	-	RO	0	**Reserved**
1	rx_busy	R	0	**Receive Busy** 0: UART is not receiving a character. 1: UART is receiving a character.
0	tx_busy	R	0	**Transmit Busy** 0: UART is not transmitting data. 1: UART is transmitting data.

Table 12-10: UART Interrupt Enable Register

UART Interrupt Enable Register				UARTn_INT_EN [0x0008]
Bits	**Name**	**Access**	**Reset**	**Description**
31:7	-	RO	0	**Reserved**
6	tx_he	R/W	0	**Transmit FIFO Half-Empty Event Interrupt Enable** 0: Disabled 1: Enabled
5	-	RO	0	**Reserved**
4	rx_thd	R/W	0	**Receive FIFO Threshold Event Interrupt Enable** 0: Disabled 1: Enabled
3	rx_ov	R/W	0	**Receive FIFO Overrun Event Interrupt Enable** 0: Disabled 1: Enabled
2	cts_ev	R/W	0	**CTS Signal Change Event Interrupt Enable** 0: Disabled 1: Enabled
1	rx_par	R/W	0	**Receive Parity Event Interrupt Enable** 0: Disabled 1: Enabled
0	rx_ferr	R/W	0	**Receive Frame Error Event Interrupt Enable** 0: Disabled 1: Enabled

Table 12-11: UART Interrupt Flag Register

UART Interrupt Flag				UARTn_INT_FL [0x000C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:7	-	RO	0	**Reserved**
6	tx_he	R/W1C	0	**Transmit FIFO Half-Empty Interrupt Flag** 0: Disabled 1: Enabled
5	-	RO	0	**Reserved**
4	rx_thd	R/W1C	0	**Receive FIFO Threshold Interrupt Flag** 0: Disabled 1: Enabled
3	rx_ov	R/W1C	0	**Receive FIFO Overrun Interrupt Flag** 0: Disabled 1: Enabled
2	cts_ev	R/W1C	0	**CTS Signal Change Interrupt Flag** 0: Disabled 1: Enabled
1	rx_par	R/W1C	0	**Receive Parity Error Interrupt Flag** 0: Disabled 1: Enabled
0	rx_ferr	R/W1C	0	**Receive Frame Error Interrupt Flag** 0: Disabled 1: Enabled

Table 12-12: UART Clock Divisor Register

UART Clock Divisor				UARTn_CLKDIV [0x0010]
Bits	**Name**	**Access**	**Reset**	**Description**
31:20	-	RO	0	**Reserved**
19:0	clkdiv	R/W	0	**Baud Rate Divisor** This field sets the divisor used to generate the baud tick from the baud clock. For LPUART instances, 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. See *Baud Rate Generation* for information on how to use this field.

Table 12-13: UART Oversampling Control Register

UART Oversampling Control				UARTn_OSR [0x0014]
Bits	**Name**	**Access**	**Reset**	**Description**
31:3	-	RO	0	**Reserved**
2:0	osr	R/W	0	**LPUART Over Sampling Rate** For LPUART instances with FDM enabled (*UARTn_CTRL.fdm* = 1): 0: 8 × 1: 12 × 2: 16 × 3: 20 × 4: 24 × 5: 28 × 6: 32 × 7: 36 × For LPUART instances with FDM disabled (*UARTn_CTRL.fdm* = 0): 0: 128 × 1: 64 × 2: 32 × 3: 16 × 4: 8 × 5: 4 × 6 - 7: Reserved

Table 12-14: UART Transmit FIFO Register

UART Transmit FIFO				UARTn_TXPEEK [0x0018]
Bits	**Name**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
7:0	data	RO	0	**Transmit FIFO Data** Read the transmit FIFO next data without affecting the contents of the transmit FIFO. If there are no entries in the transmit FIFO, this field reads 0. *Note: The parity bit is available from this field.*

Table 12-15: UART Pin Control Register

UART Pin Control				UARTn_PNR [0x001C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:2	-	RO	0	**Reserved**
1	rts	R/W	1	**RTS Pin Output State** 0: RTS signal is driven to 0. 1: RTS signal is driven to 1.

UART Pin Control			UARTn_PNR	[0x001C]
Bits	**Name**	**Access**	**Reset**	**Description**
0	cts	RO	1	**CTS Pin State** This field returns the current sampled state of the GPIO associated with the CTS signal. 0: CTS state is 0. 1: CTS state is 1.

Table 12-16: UART Data Register

UART Data			UARTn_FIFO	[0x0020]
Bits	**Name**	**Access**	**Reset**	**Description**
31:9	-	RO	0	**Reserved**
8	rx_par	R	0	**Receive FIFO Byte Parity** If the parity feature is disabled, this bit always reads 0. If a parity error occurred during the reception of the character at the output end of the receive FIFO (that would be returned by reading the *UARTn_FIFO.data* field), this bit reads 1, otherwise it reads 0.
7:0	data	R/W	0	**Transmit/Receive FIFO Data** Writing to this field loads the next character into the transmit FIFO if the transmit FIFO is not full. Reading from this field returns the next character from the receive FIFO if the receive FIFO is not empty. If the receive FIFO is empty, 0 is returned by the hardware. For character widths less than 8, the unused bit(s) are ignored when the transmit FIFO is loaded, and the unused high bit(s) read 0 on characters read from the receive FIFO.

Table 12-17: UART DMA Register

UART DMA			UARTn_DMA	[0x0030]
Bits	**Name**	**Access**	**Reset**	**Description**
31:10	-	RO	0	**Reserved**
9	rx_en	0	0	**Receive DMA Channel Enable** 0: Disabled 1: Enabled
8:5	rx_thd_val	0	0	**Receive FIFO Level DMA Threshold** If *UARTn_STATUS.rx_lvl* < *UARTn_DMA.rx_thd_val*, then the receive FIFO DMA interface sends a signal to the DMA indicating characters are available in the UART receive FIFO to transfer to memory.
4	tx_en	R/W	0	**Transmit DMA Channel Enable** 0: Disabled 1: Enabled
3:0	tx_thd_val	R/W	0	**Transmit FIFO Level DMA Threshold** If *UARTn_STATUS.tx_lvl* < *UARTn_DMA.tx_thd_val*, the transmit DMA channel sends a signal to the DMA indicating that the UART transmit FIFO is ready to receive data from memory.

Table 12-18: UART Wakeup Enable

UART Wakeup Enable			UARTn_WKEN	[0x0034]
Bits	**Name**	**Access**	**Reset**	**Description**
31:3	-	RO	0	**Reserved**

UART Wakeup Enable			UARTn_WKEN	[0x0034]
Bits	**Name**	**Access**	**Reset**	**Description**
2	rx_thd	R/W	0	**Receive FIFO Threshold Wakeup Event Enable** 0: Disabled 1: Enabled
1	rx_full	R/W	0	**Receive FIFO Full Wakeup Event Enable** 0: Disabled 1: Enabled
0	rx_ne	R/W	0	**Receive FIFO Not Empty Wakeup Event Enable** 0: Disabled 1: Enabled

Table 12-19. UART Wakeup Flag Register

UART Wakeup Flag			UARTn_WKFL	[0x0038]
Bits	**Name**	**Access**	**Reset**	**Description**
31:3	-	RO	0	**Reserved**
2	rx_thd	R/W	0	**Receive FIFO Threshold Wakeup Event** 0: Disabled 1: Enabled
1	rx_full	R/W	0	**Receive FIFO Full Wakeup Event** 0: Disabled 1: Enabled
0	rx_ne	R/W	0	**Receive FIFO Not Empty Wakeup Event** 0: Disabled 1: Enabled

13. Serial Peripheral Interface (SPI)

The SPI peripheral is a configurable, flexible, and efficient synchronous interface between multiple SPI devices on a single bus. The SPI bus uses a single clock signal, single, dual, or quad data lines, and one or more peripheral select lines for communication with external SPI devices.

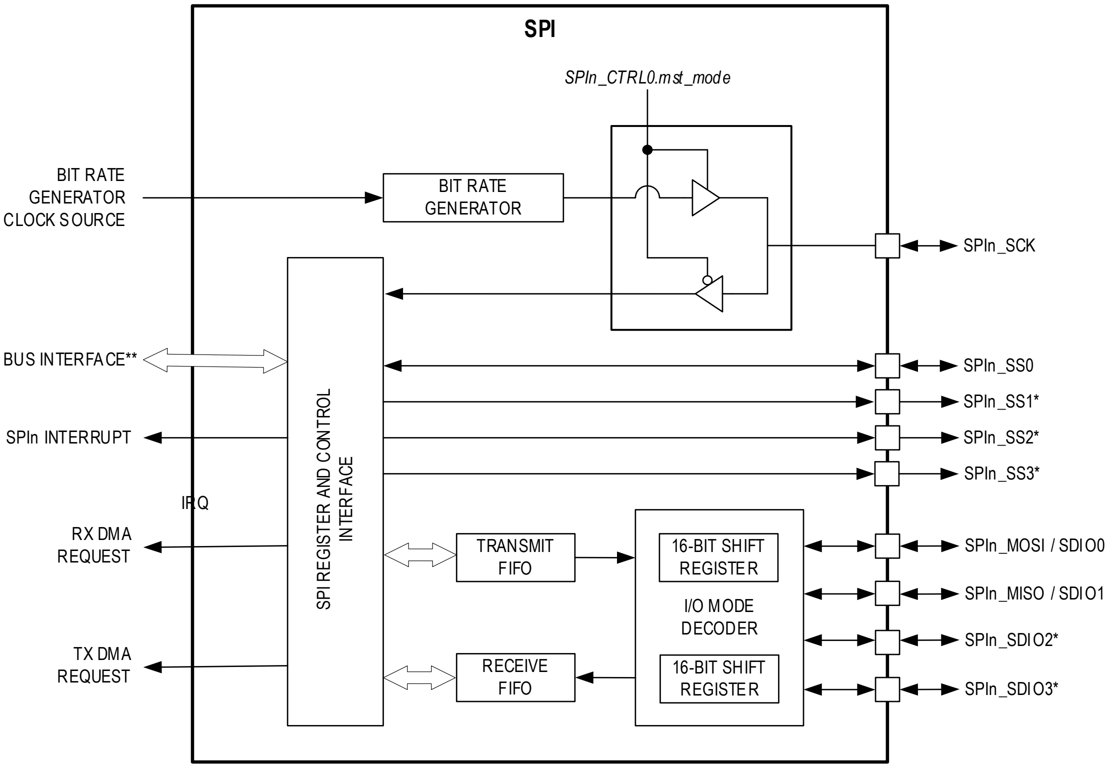
The provided SPI ports support full-duplex, bi-direction I/O, and each SPI includes a Bit Rate Generator (BRG) for generating the clock signal when operating in controller mode. Each SPI port operates independently and requires minimal processor overhead. All instances of the SPI peripheral support both controller and peripheral modes and support single controller and multi-controller networks.

Features include:

- Dedicated BRG for precision serial clock generation in controller 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 peripheral select lines
    - Programmable peripheral select level.
- Programmable peripheral select timing with respect to the SCK starting edge and ending edge.
- Multi-controller mode fault detection.

Figure 13-1 shows a high-level block diagram of the SPI peripheral. See *Table 13-1* for the peripheral-specific peripheral bus assignment and BRG clock source.

Figure 13-1: SPI Block Diagram



* The number of peripheral select and SDIO signals can vary for each instance of the peripheral.
** The bus interface (APB or AHB) can vary for each instance of the peripheral.

13.1 Instances

There are two instances of the SPI peripheral, as shown in *Table 13-1*. *Table 13-2* lists the locations of the SPI signals for each of the SPI instances.

Table 13-1: MAX78002 SPI Instances

Instance	Formats				Hardware Bus	Bit Rate Generator Clock Source	Peripheral Select Signals
	3-Wire	4-Wire	Dual	Quad			144-CSBGA
SPI0	Yes	Yes	Yes	Yes	AHB	f_{SYS_CLK}	3
SPI1	Yes	Yes	Yes	Yes	APB	f_{PCLK}	1

Note: Refer to the MAX78002 data sheet's pin description table for the list of alternate function assignments for each peripheral instance.

Table 13-2: MAX78002 SPI Peripheral Pins

Instance	Signal Description	Alternate Function
SPI0	SPI Clock	SPI0_SCK
	Peripheral Select 0	SPI0_SS0
	Peripheral Select 1	SPI0_SS1
	Peripheral Select 2	SPI0_SS2
	MOSI (SDIO0)	SPI0_MOSI
	MISO (SDIO1)	SPI0_MISO
	SDIO2	SPI0_SDIO2
	SDIO3	SPI0_SDIO3
SPI1	SPI Clock	SPI1_SCK
	Peripheral Select 0	SPI1_SS0
	MOSI (SDIO0)	SPI1_MOSI
	MISO (SDIO1)	SPI1_MISO
	SDIO2	SPI1_SDIO2
	SDIO3	SPI1_SDIO3

13.2 Formats

13.2.1 Four-Wire SPI

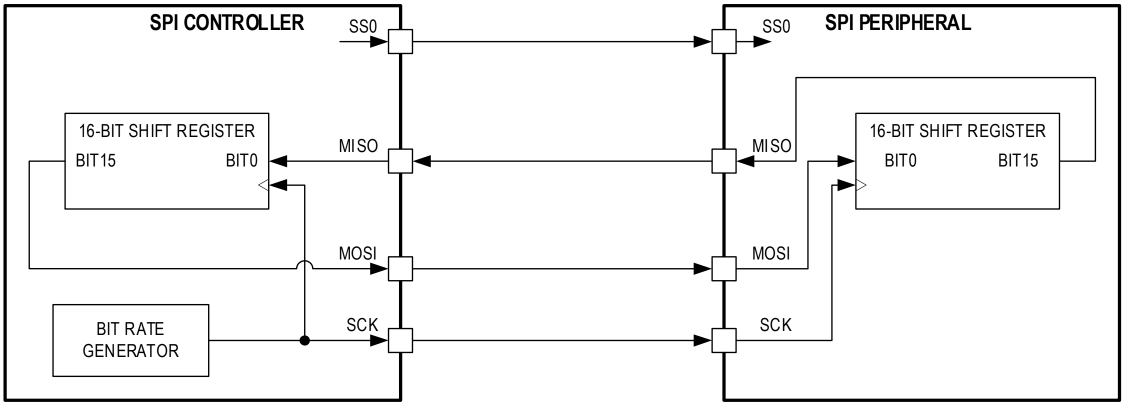
SPI devices operate as either a controller or peripheral device. Four signals are required for communication in four-wire SPI, as shown in *Table 13-3*.

Table 13-3: Four-Wire Format Signals

Signal	Description	Direction
SCK	Serial Clock	The controller generates the SCK signal, an output from the controller, and an input to the peripheral.
MOSI	Controller Output Peripheral Input	This signal is used as an output for sending data to the peripheral in controller mode. In peripheral mode, this is the input data from the controller.
MISO	Controller Input Peripheral Output	In controller mode, this signal is used as an input for receiving data from the peripheral. This signal is an output for transmitting data to the controller in peripheral mode.
SS	Peripheral Select	This signal is an output used to select a peripheral device before communication in controller mode. Peripherals may have multiple peripheral select outputs to communicate with one or more external peripheral devices. SPIn_SS0 is a dedicated input in peripheral mode that indicates an external controller is starting communication. Other peripheral select signals into the peripheral are ignored in peripheral mode.

The SPI controller starts communication with a peripheral by asserting the peripheral select output. The controller then starts the SPI clock through the SCK output pin. When a peripheral device's peripheral select pin is deasserted, the peripheral device is required to put the SPI pins in tri-state mode.

Figure 13-2: 4-Wire SPI Connection Diagram



13.2.2 Three-Wire SPI

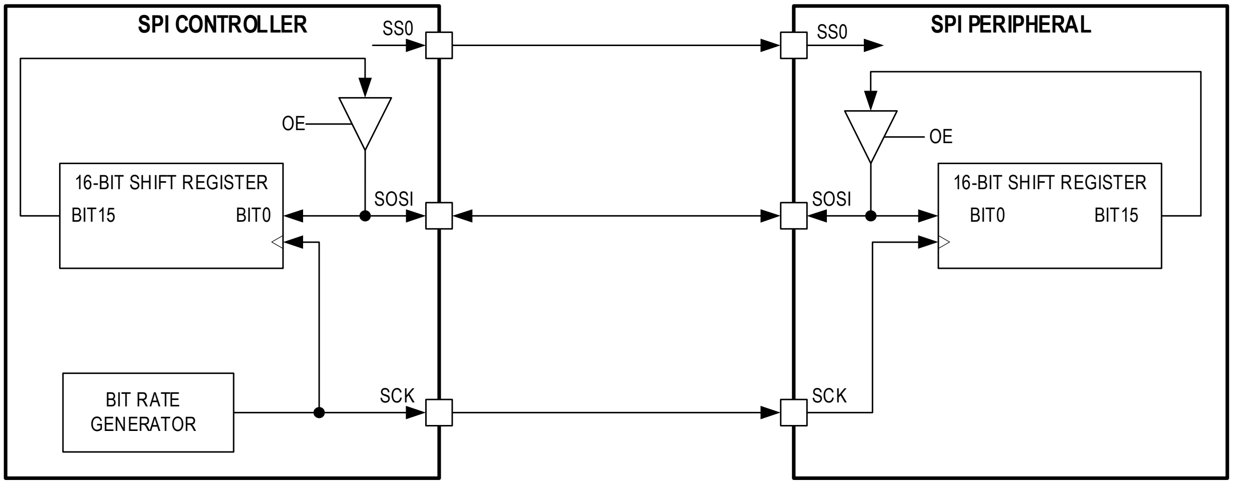
The signals in three-wire SPI operation are shown in *Table 13-4*. The MOSI signal is used as a bidirectional, half-duplex I/O referred to as peripheral input peripheral output (SISO). Three-wire SPI also uses a serial clock signal generated by the controller and a peripheral select pin controlled by the controller.

Table 13-4: Three-Wire Format Signals

Signal	Description	Direction
SCK	Serial Clock	The controller generates the serial clock signal, an output from the controller, and an input to the peripheral.
MOSI	Peripheral Input Peripheral Output	This is a half-duplex, bidirectional I/O pin used for communication between the SPI controller and peripheral. This signal is used to transmit data from the controller to the peripheral and to receive data from the peripheral by the controller.
SS	Peripheral Select	In controller mode, this signal is an output used to select a peripheral device before communication. In peripheral mode, SPIn_SS0 is a dedicated input that indicates an external controller is going to start communication. Other peripheral select signals into the peripheral are ignored in peripheral mode

A three-wire SPI network is shown in *Figure 13-3*. The controller device selects the peripheral device using the peripheral select output. The communication starts with the controller asserting the peripheral select line and then starting the clock (SCK). In three-wire SPI communication, the controller and peripheral must both know the intended direction of the data to prevent bus contention. For a write, the controller drives the data out the SISO pin. For a read, the controller must release the SISO line and let the peripheral 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 13-3: Generic 3-Wire SPI Controller to Peripheral Connection



13.3 Pin Configuration

Before configuring the SPI peripheral, first, disable any SPI activity for the port by clearing the *SPIn_CTRL0.en* field to 0.

13.3.1 SPI Alternate Function Mapping

Pin selection and configuration are required to use the SPI port. The following information applies to SPI controller and peripheral 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 MAX78002 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.

13.3.2 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 peripheral select pin for a transaction, resulting in more than four wires total. However, the communication is referred to as four-wire for historical 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.

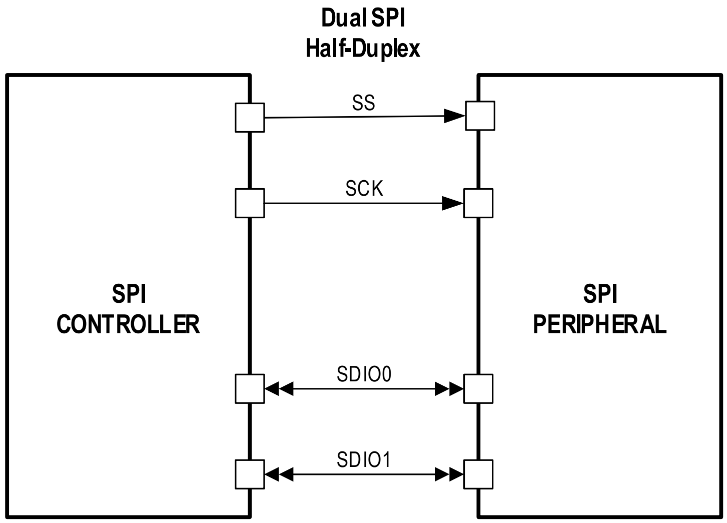
13.3.3 Three-Wire Format Configuration

Three-wire SPI uses SCK, MOSI, and one or more peripheral 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 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.

13.3.4 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 controller and peripheral for a given transaction. Dual-mode SPI uses SCK, SDIO0, SDIO1, and one or more peripheral select lines, as shown in *Figure 13-4*. The configuration of the GPIO pins for dual-mode SPI is identical to four-wire SPI, and the mode is controlled by setting *SPIn_CTRL2.data_width* to 1, indicating to the SPI hardware to use SDIO0 and SDIO1 for half-duplex communication rather than full-duplex communication.

Figure 13-4: Dual Mode SPI Connection Diagram



13.3.5 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 controller and peripheral must know the direction of transmission for each transaction. Quad-mode SPI uses SCK, SDIO0, SDIO1, SDIO2, SDIO3, and one or more peripheral select pins.

Quad-mode SPI transmits four bits per SCK cycle. Select quad-mode SPI by setting *SPIn_CTRL2.data_width* to 2.

13.4 Clock Configuration

13.4.1 Serial Clock

The SCK signal synchronizes data movement in and out of the device. The controller drives SCK as an output to the peripheral's SCK pin. When SPI is set to controller mode, the SPI bit rate generator creates the serial clock and outputs it on the configured SPIn_SCK pin. When SPI is configured for peripheral operation, the SPIn_SCK pin is an input from the external controller, and the SPI hardware synchronizes communications using the SCK input. Operating as a peripheral, if an SPI peripheral select input is not asserted, the SPI ignores any signals on the serial clock and serial data lines.

In both controller and peripheral 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 are controlled using the SPI phase control field, *SPIn_CTRL2.clkpha*. The SCK clock polarity field, *SPIn_CTRL2.clkpol*, controls if the SCK signal is active high or active low.

The SPI peripheral supports four combinations of SCK phase and polarity referred to as SPI modes 0, 1, 2, and 3. Clock Polarity (*SPIn_CTRL2.clkpol*) selects an active low/high clock and has no effect on the transfer format. Clock Phase (*SPIn_CTRL2.clkpha*) selects one of two different transfer formats.

For proper data transmission, the clock phase and polarity must be identical for the SPI controller and peripheral. The controller always places data on the MOSI line a half-cycle before the SCK edge for the peripheral to latch the data. See section *Clock Phase and Polarity Control* for additional details.

13.4.2 Peripheral Clock

See *Table 13-1* for the specific input clock, f_{INPUT_CLK}, used for each SPI instance. For 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 SPI peripheral clock. The SPI provides an internal clock, *SPI_CLK*, that is used within the SPI peripheral for the base clock to control the module and generate the SCK clock when in controller mode. Set the SPI internal clock using the field *SPIn_CLKCTRL.clkdiv* as shown in *Equation 13-1*. Valid settings for *SPIn_CLKCTRL.clkdiv* are 0 to 8, allowing a divisor of 1 to 256.

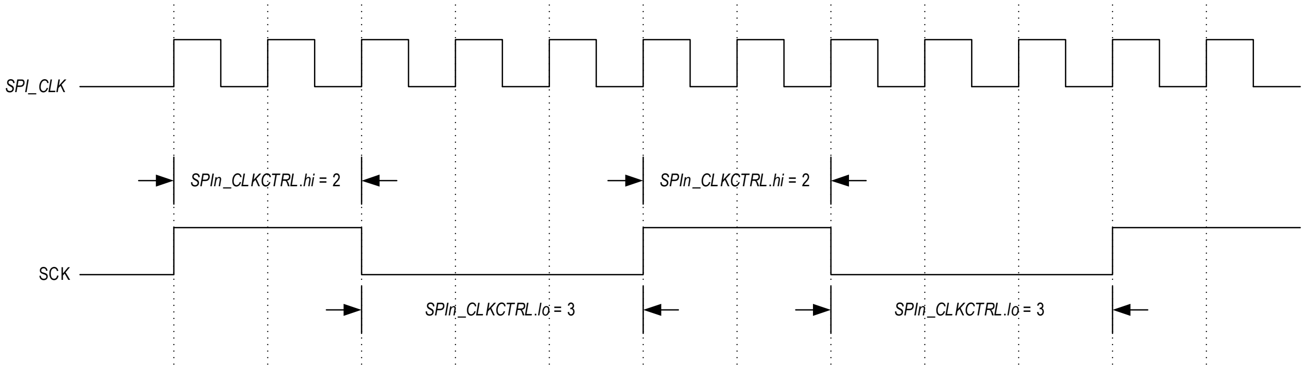
Equation 13-1: SPI Peripheral Clock

$$f_{SPI\_CLK} = \frac{f_{INPUT\_CLK}}{2^{clkdiv}}$$

13.4.3 Controller Mode Serial Clock Generation

In controller and multi-controller mode, the SCK clock is generated by the controller. The SPI peripheral 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 SPI peripheral clock as a base value, and the high and low values are a count of the number of f_{SPI_CLK} clocks. *Figure 13-5* 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 13-2* and *Equation 13-3* for calculating the SCK high and low time from the *SPIn_CLKCTRL.hi* and *SPIn_CLKCTRL.lo* field values.

Figure 13-5: SCK Clock Rate Control



Equation 13-2: SCK High Time

$$t_{SCK\_HI} = t_{SPIn\_CLK} \times SPIn\_CLKCTRL.hi$$

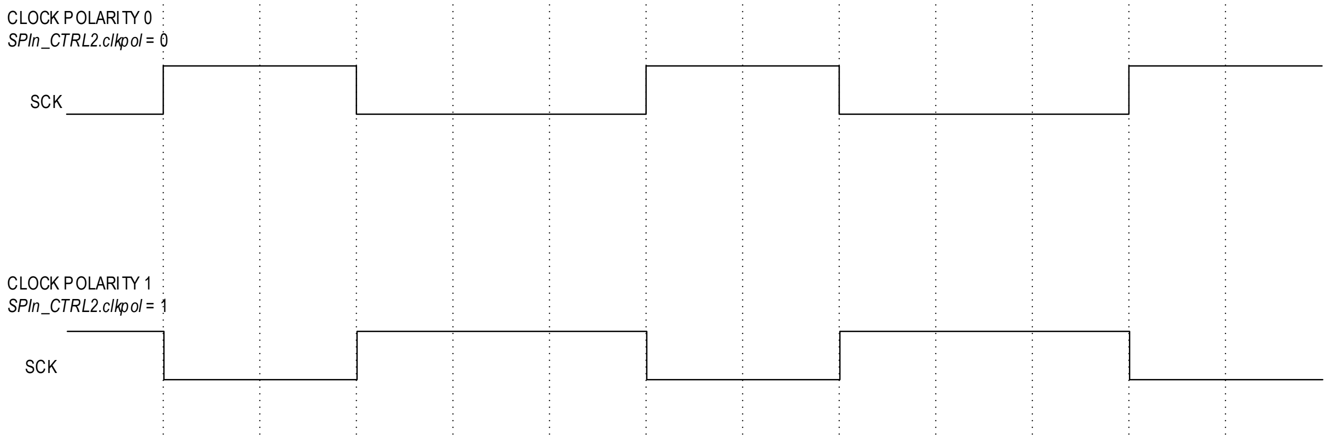
Equation 13-3: SCK Low Time

$$t_{SCK\_LOW} = t_{SPIn\_CLK} \times SPIn\_CLKCTRL.lo$$

13.4.4 Clock Phase and Polarity Control

SPI supports four combinations of clock and phase polarity, as shown in *Table 13-5*. 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 13-6*. Clock polarity does not affect the transfer format for SPI. The 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 13-6: SPI Clock Polarity



For proper data transmission, the clock phase and polarity must be identical for the SPI controller and peripheral. The controller always places data on the MOSI line a half-cycle before the SCK edge for the peripheral to latch the data.

Table 13-5: SPI Modes Clock Phase and Polarity Operation

SPI Mode	SPIn_CTRL2 *clkpha*	SPIn_CTRL2 *clkpol*	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

13.4.5 Transmit and Receive 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.

13.4.6 Interrupts and Wakeups

The SPI 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 the software 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.
    - Peripheral mode only, controller mode stalls the serial clock.
- Transmit FIFO Overrun.
- Receive FIFO Underrun.
- Receive FIFO Overrun.
    - Peripheral mode only, controller mode stalls the serial clock.
- SPI supports interrupts for the internal state of the SPI as well as external signals. The following transmission interrupts are supported:
    - SS asserted or deasserted.
    - SPI transaction complete.
        - Controller mode only.
    - Peripheral mode transaction aborted.
    - Multi-controller fault.

The SPI port can wake up the microcontroller from low-power modes when the wake event is enabled. SPI events that can wake the microcontroller are:

- Receive FIFO full.
- Transmit FIFO empty.
- Receive FIFO threshold.
- Transmit FIFO threshold.

13.5 Registers

See *Table 3-3* for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own independent set of registers, shown in *Table 13-6*. 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 1-1* 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 13-6: SPI Register Summary

Offset	Register Name	Access	Description
[0x0000]	*SPIn_FIFO32*	R/W	*SPI FIFO Data Register*
[0x0000]	*SPIn_FIFO16*	R/W	*SPI 16-bit FIFO Data Register*
[0x0000]	*SPIn_FIFO8*	R/W	*SPI 8-bit FIFO Data Register*

Offset	Register Name	Access	Description
[0x0004]	*SPIn_CTRL0*	R/W	*SPI Controller Signals Control Register*
[0x0008]	*SPIn_CTRL1*	R/W	*SPI Transmit Packet Size Register*
[0x000C]	*SPIn_CTRL2*	R/W	*SPI Static Configuration Register*
[0x0010]	*SPIn_SSTIME*	R/W	*SPI Peripheral Select Timing Register*
[0x0014]	*SPIn_CLKCTRL*	R/W	*SPI Controller Clock Configuration Register*
[0x001C]	*SPIn_DMA*	R/W	*SPI DMA Control Register*
[0x0020]	*SPIn_INTFL*	R/W1C	*SPI Interrupt Flag Register*
[0x0024]	*SPIn_INTEN*	R/W	*SPI Interrupt Enable Register*
[0x0028]	*SPIn_WKFL*	R/W1C	*SPI Wakeup Flags Register*
[0x002C]	*SPIn_WKEN*	R/W	*SPI Wakeup Enable Register*
[0x0030]	*SPIn_STAT*	RO	*SPI Status Register*

13.5.1 Register Details

Table 13-7: SPI FIFO32 Register

SPI FIFO Data			**SPIn_FIFO32**	**[0x0000]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	data	R/W	0	**SPI 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 13-8: SPI 16-bit FIFO Register

SPI FIFO Data			**SPIn_FIFO16**	**[0x0000]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	R/W	0	**Reserved**
15:0	data	R/W	0	**SPI 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 13-9: SPI 8-bit FIFO Register

SPI 8-bit FIFO Data			**SPIn_FIFO8**	**[0x0000]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	-	R/W	0	**Reserved**
7:0	data	R/W	0	**SPI 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 13-10: SPI Control 0 Register

SPI Control 0			SPIn_CTRL0	[0x0004]
Bits	**Name**	**Access**	**Reset**	**Description**
31:20	-	R/W	0	**Reserved**
19:16	ss_active	R/W	0	**Controller Peripheral Select** The SPI includes up to four peripheral select lines for each port. This field selects which peripheral select pin is active when the next SPI transaction is started (*SPIn_CTRL0.start* = 1). One or more peripheral select pins can be selected for each SPI transaction by setting the bit for each peripheral select pin. For example, use SPIn_SS0 and SPIn_SS2 by setting this field to 0b0101 or select all peripheral selects by setting this field to 0b1111. *Note: This field is only used when the SPI is configured for controller mode (SPIn_CTRL0.mst_mode = 1).*
15:9	-	R/W	0	**Reserved**
8	ss_ctrl	R/W	0	**Controller Peripheral Select Control** This field controls the behavior of the peripheral select pins at the completion of a transaction. The default behavior, *ss_ctrl* = 0, deasserts the peripheral select pin at the completion of the transaction. Set this field to 1 to leave the peripheral select pins asserted at the completion of the transaction. If the external device supports this behavior, leaving the peripheral select pins asserted allows multiple transactions without the delay associated with deassertion of the peripheral select pin between transactions. 0: Peripheral Select is deasserted at the end of a transmission. 1: Peripheral Select stays asserted at the end of a transmission.
7:6	-	R/W	0	**Reserved.**
5	start	R/W1O	0	**Controller Start Data Transmission** Set this field to 1 to start an SPI controller mode transaction. 0: No controller mode transaction active. 1: Initiate the data transmission. Ensure that all pending transactions are complete before setting this field to 1. *Note: This field is only used when the SPI is configured for controller mode (SPIn_CTRL0.mst_mode = 1).*
4	ss_io	R/W	0	**Controller Peripheral Select Signal Direction** Set the I/O direction for 0: Peripheral select is an output. 1: Peripheral select is an input. *Note: This field is only used when the SPI is configured for controller mode (SPIn_CTRL0.mst_mode = 1).*
3:2	-	R/W	0	**Reserved**
1	mst_mode	R/W	0	**SPI Controller Mode Enable** This field selects between peripheral mode and controller mode operation for the SPI port. Write this field to 0 to operate as an SPI peripheral. Set this field to 1 to set the port as an SPI controller. 0: Peripheral mode SPI operation. 1: Controller mode SPI operation.

SPI Control 0		SPIn_CTRL0		[0x0004]
Bits	**Name**	**Access**	**Reset**	**Description**
0	en	R/W	0	**SPI Enable/Disable** This field enables and disables the SPI port. Disable the SPI port by setting this field to 0. Disabling the SPI port does not affect the SPI FIFOs or register settings. Access to SPI registers is always available. 0: SPI port is disabled. 1: SPI port is enabled.

Table 13-11: SPI Control 1 Register

SPI Transmit Packet Size		SPIn_CTRL1		[0x0008]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	rx_num_char	R	0	**Number of Receive Characters** This field returns the number of characters to receive in receive FIFO. *Note: If the SPI port is set to operate in 4-wire mode, this field is ignored, and the SPIn_CTRL1.tx_num_char field is used for both the number of characters to receive and transmit.*
15:0	tx_num_char	R	0	**Number of Transmit Characters** This field returns the number of characters to transmit from transmit FIFO. *Note: If the SPI port is set to operate in 4-wire mode, this field is used for both the number of characters to receive and transmit.*

Table 13-12: SPI Control 2 Register

SPI Control 2		SPIn_CTRL2		[0x000C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:20	-	R/W	0	**Reserved**
19:16	ss_pol	R/W	0	**Peripheral 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. *Note: This field is ignored for Dual SPI, SPIn_CTRL2.data_width =1, and Quad SPI, SPIn_CTRL2.data_width =2.*
14	-	R/W	0	**Reserved**

SPI 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*: *data_width* = 0. Set this field to 0, indicating SPIn_MOSI is used for half-duplex communication. *Four-wire full-duplex SPI*: *data_width* = 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*: *data_width* = 1. Set this field to 1, indicating SPIn_SDIO0 and SPIn_SDIO1 are used for half-duplex communication. *Quad-mode SPI*: *data_width* = 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. *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.*
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. *Note: 1-bit and 9-bit character lengths are not supported.* *Note: 2-bit and 10-bit character lengths do not support maximum SCK speeds in controller mode. SPIn_CLKCTRL.clkdiv must be > 0.* *Note: For Dual and Quad mode SPI, the character size should be divisible by the number of bits per SCK cycle.*
7:2	-	R/W	0	**Reserved**
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 13-13: SPI Peripheral Select Timing Register

SPI Peripheral Select Timing			**SPIn_SSTIME**	**[0x0010]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:24	-	R/W	0	**Reserved**
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 (peripheral select inactive) and the start of the next transaction (peripheral select active). 0: 256. 1: 1. 2: 2. 3:3. … : … 254: 254. 255: 255. *Note: The SPIn_SSTIME register bit settings only apply when SPI is operating in controller mode (SPIn_CTRL0.mst_mode = 1)*
15:8	post	R/W	0	**Peripheral Select Hold Post Last SCK** Set this field to the number of system clock cycles for SS to remain active after the last SCK edge. 0: 256. 1: 1. 2: 2. 3:3. … : … 254: 254. 255: 255. *Note: The SPIn_SSTIME register bit settings only apply when SPI is operating in controller mode (SPIn_CTRL0.mst_mode = 1)*
7:0	pre	R/W	0	**Peripheral Select Delay to First SCK** Set the number of system clock cycles the peripheral select is held active before the first SCK edge. 0: 256. 1: 1. 2: 2. 3:3. … : … 254: 254. 255: 255. *Note: The SPIn_SSTIME register bit settings only apply when SPI is operating in controller mode (SPIn_CTRL0.mst_mode = 1)*

Table 13-14: SPI Controller Clock Configuration Registers

SPI Controller Clock Configuration			**SPIn_CLKCTRL**	**[0x0014]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:20	-	R/W	0	**Reserved**

SPI Controller 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^{clkdiv} to generate the SPI peripheral clock. $f_{SPInCLK} = \frac{f_{SPIn\_INPUT\_CLK}}{2^{clkdiv}}$ Valid values for scale are 0 to 8 inclusive. Values greater than 8 are reserved. *Note: 1-bit and 9-bit character lengths are not supported.* *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.*
15:8	hi	R/W	0	**SCK Hi Clock Cycles Control** 0: Hi duty cycle control disabled. Only valid if *SPIn_CLKCTRL.clkdiv* = 0. 1 - 15: The number of SPI peripheral clocks, $f_{SPInCLK}$, that SCK is high. *Note: 1-bit and 9-bit character lengths are not supported.* *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.*
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_CLKCTRL.hi* field. 0: Low duty cycle control disabled. Setting this field to 0 is only valid if *SPIn_CLKCTRL.clkdiv* = 0. 1 to 15: The number of SPI peripheral clocks, $f_{SPInCLK}$, that the SCK signal is low. *Note: 1-bit and 9-bit character lengths are not supported.* *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.*

Table 13-15: SPI DMA Control Registers

SPI DMA Control			SPIn_DMA	[0x001C]
Bits	**Name**	**Access**	**Reset**	**Description**
31	dma_rx_en	R/W	0	**Receive DMA Enable** 0: Disabled. Any pending DMA requests are cleared. 1: Enabled.
30:24	dma_rx_en	R	0	**Number of Bytes in the Receive FIFO** Read returns the number of bytes currently in the receive FIFO.
23	rx_flush	R/W1O	-	**Clear the Receive FIFO** 1: Clear the receive FIFO and any pending receive FIFO flags in *SPIn_INTFL*. This should be done when the receive FIFO is inactive. Note: Writing a 0 has no effect.
22	rx_fifo_en	R/W	0	**Receive FIFO Enabled** 0: Disabled. 1: Enabled.
21	-	R/W	0	**Reserved**

SPI DMA Control		SPIn_DMA		[0x001C]
Bits	**Name**	**Access**	**Reset**	**Description**
20:16	rx_thd_val	R/W	0	**Receive FIFO Threshold Level** Set this value to the desired receive FIFO threshold level. When the receive FIFO level crosses above this setting, a DMA request is triggered if enabled (SPIn_DMA.*dma_tx_en* = 1), and SPIn_INTFL.*rx_thd* is set. Valid values are 0 to 30. *Note: 31 is an invalid setting.*
15	dma_tx_en	R/W	0	**Transmit DMA Enable** 0: Disabled. Any pending DMA requests are cleared. 1: Transmit DMA is enabled.
14:8	tx_lvl	RO	0	**Number of Bytes in the Transmit FIFO** Read this field to determine the number of bytes currently in the transmit FIFO.
7	tx_flush	R/W	0	**Transmit FIFO Clear** Set this bit to clear the transmit FIFO and all transmit FIFO flags in the SPIn_INTFL register. *Note: The transmit FIFO should be disabled (SPIn_DMA.tx_fifo_en = 0) before setting this field.* *Note: Setting this field to 0 has no effect.*
6	tx_fifo_en	R/W	0	**Transmit FIFO Enabled** 0: Disabled. 1: Enabled.
5	-	R/W	0	**Reserved**
4:0	tx_thd_val	R/W	0x10	**Transmit FIFO Threshold Level** Set this value to the desired transmit FIFO threshold level. When the transmit FIFO count (SPIn_DMA.*tx_lvl*) falls below this value, a DMA request is triggered if enabled (SPIn_DMA.*dma_tx_en* = 1), and SPIn_INTFL.*tx_thd* becomes set.

Table 13-16: SPI Interrupt Status Flags Registers

SPI Interrupt Status Flags		SPIn_INTFL		[0x0020]
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	R/W	0	**Reserved**
15	rx_un	R/1	0	**Receive FIFO Underrun Flag** Set when a read is attempted from an empty receive FIFO.
14	rx_ov	R/W1C	0	**Receive FIFO Overrun Flag** Set if SPI is in peripheral mode, and a write to a full receive FIFO is attempted. If the SPI is in controller mode, this bit is not set as the SPI stalls the clock until data is read from the receive FIFO.
13	tx_un	R/W1C	0	**Transmit FIFO Underrun Flag** Set if SPI is in peripheral mode, and a read from empty transmit FIFO is attempted. If SPI is in controller mode, this bit is not set as the SPI stalls the clock until data is written to the empty transmit FIFO.
12	tx_ov	R/W1C	0	**Transmit FIFO Overrun Flag** Set when a write is attempted, and the transmit FIFO is full.

SPI Interrupt Status Flags			**SPIn_INTFL**	**[0x0020]**
Bits	**Name**	**Access**	**Reset**	**Description**
11	mst_done	R/W1C	0	**Controller Data Transmission Done Flag** Set if SPI is in controller mode and all transactions are complete. *SPIn_CTRL1.tx_num_char* has been reached.
10	-	R/W	0	**Reserved**
9	abort	R/W1C	0	**Peripheral Mode Transaction Abort Detected Flag** Set if the SPI is in peripheral mode, and SS is deasserted before a complete character is received.
8	fault	R/W1C	0	**Multi-Controller Fault Flag** Set if the SPI is in controller mode, multi-controller mode is enabled, and a peripheral select input is asserted. A collision also sets this flag.
7:6	-	R/W	0	**Reserved**
5	ssd	R/W1C	0	**Peripheral Select Deasserted Flag**
4	ssa	R/W1C	0	**Peripheral Select Asserted Flag**
3	rx_full	R/W1C	0	**Receive FIFO Full Flag**
2	rx_thd	R/W1C	0	**Receive FIFO Threshold Level Crossed Flag** Set when the receive FIFO exceeds the value in *SPIn_DMA.rx_lvl*. Cleared once receive FIFO level drops below *SPIn_DMA.rx_lvl.*
1	tx_em	R/W1C	1	**Transmit FIFO Empty Flag** This field is set to 1 by hardware if the transmit FIFO is empty.
0	tx_thd	R/W1C	0	**Transmit FIFO Threshold Level Crossed Flag** This field is set to 1 by hardware when the transmit FIFO is less than the value in *SPIn_DMA.tx_lvl*. This field is cleared by hardware once transmit FIFO level exceeds *SPIn_DMA.tx_lvl.*

Table 13-17: SPI Interrupt Enable Registers

SPI Interrupt Enable			**SPIn_INTEN**	**[0x0024]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	R/W	0	**Reserved**
15	rx_un	R/W	0	**Receive FIFO Underrun Interrupt Enable** 0: Disabled. 1: Enabled.
14	rx_ov	R/W	0	**Receive FIFO Overrun Interrupt Enable** 0: Disabled. 1: Enabled.
13	tx_un	R/W	0	**Transmit FIFO Underrun Interrupt Enable** 0: Disabled. 1: Enabled.
12	tx_ov	R/W	0	**Transmit FIFO Overrun Interrupt Enable** 0: Disabled. 1: Enabled.
11	mst_done	R/W	0	**Controller Data Transmission Done Interrupt Enable** 0: Disabled. 1: Enabled.
10	-	R/W	0	**Reserved**

SPI Interrupt Enable		SPIn_INTEN		[0x0024]
Bits	**Name**	**Access**	**Reset**	**Description**
9	abort	R/W	0	**Peripheral Mode Abort Detected Interrupt Enable** 0: Disabled. 1: Enabled.
8	fault	R/W	0	**Multi-Controller Fault Interrupt Enable** 0: Disabled. 1: Enabled.
7:6	-	R/W	0	**Reserved**
5	ssd	R/W	0	**Peripheral Select Deasserted Interrupt Enable** 0: Disabled. 1: Enabled.
4	ssa	R/W	0	**Peripheral Select Asserted Interrupt Enable** 0: Disabled. 1: Enabled.
3	rx_full	R/W	0	**Receive FIFO Full Interrupt Enable** 0: Disabled. 1: Enabled.
2	rx_thd	R/W	0	**Receive FIFO Threshold Level Crossed Interrupt Enable** 0: Disabled. 1: Enabled.
1	tx_em	R/W	0	**Transmit FIFO Empty Interrupt Enable** 0: Disabled. 1: Enabled.
0	tx_thd	R/W	0	**Transmit FIFO Threshold Level Crossed Interrupt Enable** 0: Disabled. 1: Enabled.

Table 13-18: SPI Wakeup Status Flags Registers

SPI Wakeup Flags		SPIn_WKFL		[0x0028]
Bits	**Name**	**Access**	**Reset**	**Description**
31:4	-	R/W	0	**Reserved**
3	rx_full	R/W1C	0	**Wake on Receive FIFO Full Flag** 0: Normal operation. 1: Wake condition occurred.
2	rx_thd	R/W1C	0	**Wake on Receive FIFO Threshold Level Crossed Flag** 0: Normal operation. 1: Wake condition occurred.
1	tx_em	R/W1C	0	**Wake on Transmit FIFO Empty Flag** 0: Normal operation. 1: Wake condition occurred.
0	tx_thd	R/W1C	0	**Wake on Transmit FIFO Threshold Level Crossed Flag** 0: Normal operation. 1: Wake condition occurred.

Table 13-19: SPI Wakeup Enable Registers

SPI Wakeup Enable		SPIn_WKEN		[0x002C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:4	-	R/W	0	**Reserved**

SPI Wakeup Enable			SPIn_WKEN	[0x002C]
Bits	**Name**	**Access**	**Reset**	**Description**
3	rx_full	R/W	0	**Wake On Receive FIFO Full Enable** 0: Wake event is disabled. 1: Wake event is enabled.
2	rx_thd	R/W	0	**Wake On Receive FIFO Threshold Level Crossed Enable** 0: Wake event is disabled. 1: Wake event is enabled.
1	tx_em	R/W	0	**Wake On Transmit FIFO Empty Enable** 0: Wake event is disabled. 1: Wake event is enabled.
0	tx_thd	R/W	0	**Wake On Transmit FIFO Threshold Level Crossed Enable** 0: Wake event is disabled. 1: Wake event is enabled.

Table 13-20: SPI Peripheral Select Timing Registers

SPI Status			SPIn_STAT	[0x0030]
Bits	**Name**	**Access**	**Reset**	**Description**
31:1	-	R/W	0	**Reserved**
0	busy	R	0	**SPI Active Status** This field returns the SPI status. 0: SPI is not active. In controller mode, the *busy* flag is cleared when the last character is sent. In peripheral mode, the *busy* field is cleared when the configured peripheral select input is deasserted. 1: SPI is active. In controller mode, the *busy* flag is set when a transaction starts. In peripheral mode, the *busy* flag is set when a configured peripheral select input is asserted. *Note: SPIn_CTRL0, SPIn_CTRL1, SPIn_CTRL2, SPIn_SSTIME, and* SPIn_CLKCTRL *should not be configured if this bit is set.*

14. I^{2}C Controller/Peripheral Serial Communications Peripheral

The I^{2}C peripherals can be configured as either an I^{2}C controller or an I^{2}C peripheral at standard data rates. For simplicity, I2Cn is used throughout this section to refer to any of the I^{2}C peripherals.

For detailed information on I^{2}C bus operation, refer to Analog Devices Application Note 4024 SPI/I^{2}C Bus Lines Control Multiple Peripherals.

14.1 I^{2}C Controller/Peripheral Features

Each I^{2}C controller/peripheral is compliant with the I^{2}C Bus Specification and includes the following features:

- Communicates through a serial data bus (SDA) and a serial clock line (SCL)
- Operates as either a controller or peripheral device as a transmitter or receiver
- Supports I^{2}C 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-controller systems, including support for arbitration and clock synchronization for Standard Mode, Fast Mode, and Fast Mode Plus
- Supports 7- and 10-bit addressing
- Supports RESTART condition
- Supports clock stretching
- Provides transfer status interrupts and flags
- Provides DMA data transfer support
- Supports I^{2}C timing parameters fully controllable through software
- Provides glitch filter and Schmitt trigger hysteresis on SDA and SCL
- Provides control, status, and interrupt events for maximum flexibility.
- Provides independent 8-byte receive FIFO and 8-byte transmit FIFO.
- Provides transmit FIFO preloading
- Provides programmable interrupt threshold levels for the transmit and receive FIFO.

14.2 Instances

The three instances of the peripheral are shown in *Table 14-1*. The table lists the locations of the SDA and SCL signals for each of the I^{2}C peripherals.

Table 14-1: MAX78002 I^{2}C Peripheral Pins

I^{2}C Instance	Alternate Function	Alternate Function #
I2C0	I2C0_SCL	AF1
	I2C0_SDA	AF1
I2C1	I2C1_SCL	AF1
	I2C1_SDA	AF1
I2C2	I2C2_SCL	AF1
	I2C2_SDA	AF1
Note: Refer to the device datasheet's pin description table for alternate function assignments per package.		

14.3 I^{2}C Overview

14.3.1 I^{2}C Bus Terminology

Table 14-2 contains terms and definitions used in this chapter for the I^{2}C bus terminology.

Table 14-2: I^{2}C Bus Terminology

Term	Definition
Transmitter	The device sending data on the bus.
Receiver	The device receiving data from the bus.
Controller	The device that initiates a transfer, generates the clock signal, and terminates a transfer.
Peripheral	The device addressed by a controller.
Multi-controller	More than one controller can attempt to control the bus at the same time without corrupting the message.
Arbitration	Procedure to ensure that, if more than one controller simultaneously tries to control the bus, only one can do so, and the resulting message is not corrupted.
Synchronization	The procedure to synchronize the clock signals of two or more devices.
Clock Stretching	When a peripheral device holds SCL low to pause a transfer until it is ready. Clock stretching is an optional feature according to the I^{2}C specification; thus, a controller does not have to support peripheral clock stretching if none of the peripherals in the system are capable of clock stretching.

14.3.2 I^{2}C Transfer Protocol Operation

The I^{2}C protocol operates over a two-wire bus: a clock circuit (SCL) and a data circuit (SDA). I^{2}C 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 controller sends a START or repeated START condition, followed by the I^{2}C peripheral address of the targeted peripheral device plus a read/write bit. The controller can transmit data to the peripheral (a 'write' operation) or receive data from the peripheral (a 'read' operation). Information is sent most-significant-bit (MSB) first. Following the peripheral address, the controller indicates a read or write operation and then exchanges data with the addressed peripheral. 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 controller 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 controller begins a new transmission. The number of bytes that can be transmitted per transfer is unrestricted.

14.3.3 START and STOP Conditions

A START condition occurs when a bus controller pulls SDA from high to low while SCL is high, and a STOP condition occurs when a bus controller 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.

14.3.4 Controller Operation

I^{2}C transmit and receive data transfer operations occur through the *I2Cn_FIFO* register. Writes to the register load the transmit FIFO and reads of the register return data from the receive FIFO. If a peripheral sends a NACK in response to a write operation, the I^{2}C controller generates an interrupt. The I^{2}C 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 I^{2}C controller stops the clock to allow time to read bytes from the receive FIFO or load bytes into the transmit FIFO.

14.3.5 Acknowledge and Not Acknowledge

An acknowledge bit (ACK) is generated by the receiver, whether I^{2}C controller or peripheral, 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 I^{2}C controller can then either generate a STOP condition to abort the transfer or generate a repeated START condition (that is, send a START condition without an intervening STOP condition) to start a new transfer.

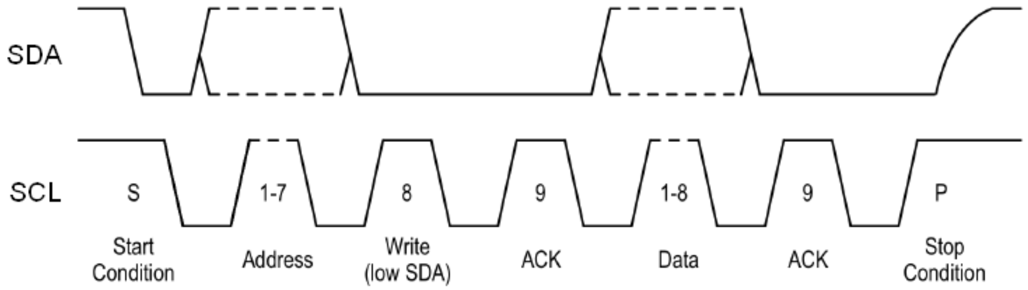
A receiver can generate a NACK after a byte transfer if any of the following conditions occur:

- No receiver is present on the bus with the transmitted address. In that case, no device responds with an acknowledge signal.
- The receiver cannot receive or transmit because it is busy and is not ready to start communication with the controller.
- 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 I^{2}C controller has requested data from a peripheral, it signals the peripheral to stop transmitting by sending a NACK following the last byte it requires.

14.3.6 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 I^{2}C 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 14-1*.

Figure 14-1: I^{2}C Write Data Transfer



An example of an I^{2}C data transfer is as follows:

1. A bus controller indicates a data transfer to a peripheral with a START condition.
2. The controller then transmits one byte with a 7-bit peripheral 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 controller releases SDA. During this clock period, the addressed peripheral responds with an ACK by pulling SDA low.
4. The controller senses the ACK condition and begins transferring data. If reading from the peripheral, it floats SDA and allows the peripheral to drive SDA to send data. After each byte, the controller drives SDA low to acknowledge the byte. If writing to the peripheral, the controller 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 peripheral to reply with the ACK indication.
5. After the last byte is transferred, the controller indicates the transfer is complete by generating a STOP condition. A STOP condition is generated when the controller pulls SDA from low to high while SCL is high.

14.4 Configuration and Usage

14.4.1 SCL and SDA Bus Drivers

SCL and SDA are open-drain signals. In this device, once the I^{2}C 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 I^{2}C peripheral device can hold SCL low, such as for clock stretching. Push-pull operation is enabled by setting *I2Cn_CTRL.scl* to 1. SDA, on the other hand, always operates in open-drain mode.

14.4.2 SCL Clock Configurations

The SCL frequency depends on the values of the I^{2}C 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 affects the target SCL frequency calculation.

14.4.3 SCL Clock Generation for Standard, Fast and Fast-Plus Modes

The controller generates the I^{2}C clock on the SCL line. When operating as a controller, the software must configure the *I2Cn_CLKHI* and *I2Cn_CLKLO* registers for the desired I^{2}C operating frequency.

The SCL high time is configured in the I^{2}C Clock High Time register field *I2Cn_CLKHI.hi* using *Equation 14-2*. The SCL low time is configured in the I^{2}C Clock Low Time register field *I2Cn_CLKLO.lo* using *Equation 14-3*. Each of these fields is 8-bits. The I^{2}C frequency value is shown in *Equation 14-1*.

Equation 14-1: I^{2}C Clock Frequency

$$f_{I2C\_CLK} = \frac{1}{t_{I2C\_CLK}}\ is\ either\ f_{PCLK}\ or\ f_{IBRO}$$

Equation 14-2: I^{2}C Clock High Time Calculation

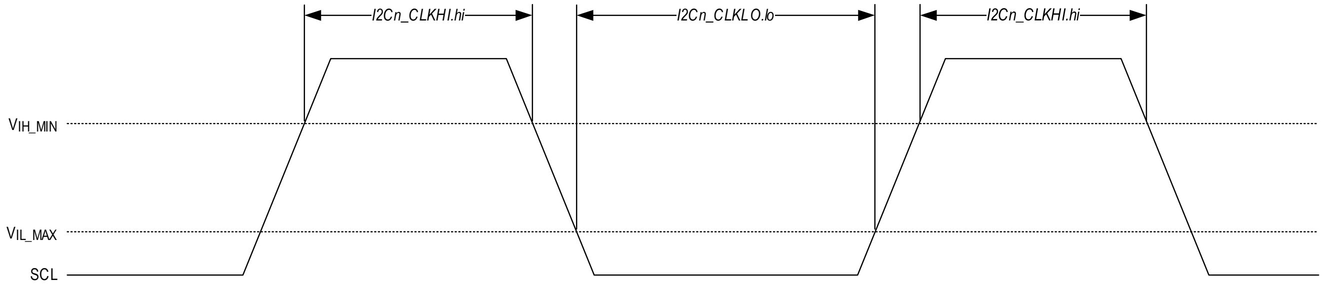
$$t_{SCL\_HI} = t_{I2C\_CLK} \times (\mathrm{I2Cn\_CLKHI}.\,hi + 1)$$

Equation 14-3: I^{2}C Clock Low Time Calculation

$$t_{SCL\_LO} = t_{I2C\_CLK} \times (\mathrm{I2Cn\_CLKLO}.\,lo + 1)$$

Figure 14-2 shows the association between the SCL clock low and high times for Standard Mode, Fast Mode, and Fast Mode Plus I^{2}C frequencies.

Figure 14-2: I²C SCL Timing for Standard, Fast and Fast-Plus Modes



During synchronization, external controllers or external peripherals can drive SCL simultaneously. This affects the SCL duty cycle. By monitoring SCL, the controller determines if an external controller or peripheral 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 controller pulls SCL low before the controller has finished counting SCL high cycles, the controller starts counting SCL low cycles and releases SCL once the time period, *I2Cn_CLKLO.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, including bus loading, other devices on the bus holding SCL low, and the filter delay time of this device.

14.4.4 SCL Clock Generation for Hs-Mode

The values programmed into the *I2Cn_HSCLK.lo* register and *I2Cn_HSCLK.hi* register must be determined to operate the I²C interface in Hs-Mode at its maximum speed (~3.4MHz). Since the Hs-Mode operation is entered by first using one of the lower speed modes for a preamble, a relevant lower speed mode must also be configured. See *SCL Clock Generation for Standard, Fast and Fast-Plus Modes* for information regarding the configuration of lower speed modes.

14.4.4.1 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.

14.4.4.2 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 14-4: I²C Target SCL Frequency

$$\textit{Desired Target Maximum I}^2\textit{C Frequency: } f_{SCL} = \frac{1}{t_{SCL}}.$$

In Hs-Mode, the analog glitch filter within the device adds a minimum delay of t_{AF_MIN} = 10ns.

Equation 14-5: Determining the I2Cn_HSCLK.lo Register Value

$$I2Cn\_HS\_CLK.lo = MAX\left\{\left\lceil\left(\frac{t_{LOW\_MIN} + t_{FCL\_MAX} + t_{I2C\_CLK} - t_{AF\_MIN}}{t_{I2C\_CLK}}\right)\right\rceil - 1, \quad \frac{t_{SCL}}{t_{I2C\_CLK}} - 1\right\}$$

Equation 14-6: Determining the I2Cn_HSCLK.hi Register Value

$$I2Cn\_HS\_CLK.hi = \left\lfloor \left(\frac{t_{HIGH\_MIN} + t_{rCL\_MAX} + t_{I2C\_CLK} - t_{AF\_MIN}}{t_{I2C\_CLK}} \right) \right\rfloor - 1$$

Equation 14-7: The Calculated Frequency of the I²C Bus Clock Using the Results of Equation 14-5 *and* Equation 14-6

$$Calculated\ Frequency = \left((I2Cn\_HS\_CLK.hi + 1) + (I2Cn\_HS\_CLK.lo + 1) \right) * \ t_{I2C\_CLK}$$

Table 14-3 shows the I²C bus clock calculated frequencies given different f_{SYS_CLK} frequencies.

Table 14-3: Calculated I²C Bus Clock Frequencies

f_{SYS_CLK} (MHz)	*I2Cn_HSCLK.hi*	*I2Cn_HSCLK.lo*	Calculated Frequency (MHz)
100	4	9	3.3
50	2	4	3.125
25	1	2	2.5

14.4.5 Controller Mode Addressing

After a START condition, the I²C peripheral address byte is transmitted by the hardware. The I²C peripheral address is composed of a peripheral address followed by a read/write bit.

Table 14-4: I²C Peripheral Address Format

Peripheral 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	1*xx*	*x*	HS-mode controller code
1111	1*xx*	*x*	Reserved for future purposes
1111	0*xx*	*x*	10-bit peripheral addressing

In 7-bit addressing mode, the controller sends one address byte. First, to address a 7-bit address peripheral, clear the *I2Cn_MSTCTRL.ex_addr_en* field to 0, then write the address to the transmit FIFO formatted as follows where A*n* is address A6:A0.

Controller writing to peripheral: 7-bit address : [A6 A5 A4 A3 A2 A1 A0 0]

Controller reading from peripheral: 7-bit address : [A6 A5 A4 A3 A2 A1 A0 1]

In 10-bit addressing mode (*I2Cn_MSTCTRL.ex_addr_en* = 1), the first byte the controller sends is the 10-bit peripheral Addressing byte that 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 peripheral. If the operation is a read, it is followed by a repeated START. The software then writes the 10-bit address again with a 1 for the R/W bit. This I²C then starts receiving data from the peripheral device.

14.4.6 Controller Mode Operation

The peripheral operates in controller mode when controller mode enable (*I2Cn_CTRL.mst_mode*) is set to 1. To initiate a transfer, the controller generates a START condition by setting *I2Cn_MSTCTRL.start* = 1. If the bus is busy, it does not generate a START condition until the bus is available.

A controller can communicate with multiple peripheral devices without relinquishing the bus. Instead of generating a STOP condition after communicating with the first peripheral, the controller generates a Repeated START condition, or RESTART, by setting *I2Cn_MSTCTRL.restart* = 1. If a transaction is in progress, the peripheral finishes the transaction before generating a RESTART. The peripheral then transmits the peripheral address stored in the transmit FIFO. The *I2Cn_MSTCTRL.restart* bit is automatically cleared to 0 as soon as the controller begins a RESTART condition.

I2Cn_MSTCTRL.start is automatically cleared to 0 after the controller has completed a transaction and sent a STOP condition.

The controller can also generate a STOP condition by setting *I2Cn_MSTCTRL.stop* = 1.

If both START and RESTART conditions are enabled simultaneously, 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 simultaneously, 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 simultaneously, a START condition is first generated. At the end of the first transaction, a RESTART condition is generated. The *I2Cn_MSTCTRL.stop* bit is cleared and ignored.

A peripheral cannot generate START, RESTART, or STOP conditions. Therefore, when controller mode is disabled, the *I2Cn_MSTCTRL.start*, *I2Cn_MSTCTRL.restart*, and *I2Cn_MSTCTRL.stop* bits are all cleared to 0.

For controller mode operation, the following registers should only be configured when either:

1. The I^{2}C peripheral is disabled,

   or

2. The I^{2}C bus is guaranteed to be idle/free.

If this peripheral is the only controller on the bus, then changing the registers outside of a transaction (*I2Cn_MSTCTRL.start* = 0) satisfies this requirement:

- *I2Cn_CTRL.mst_mode*
- *I2Cn_CTRL.irxm_en*
- *I2Cn_CTRL.one_mst_mode*
- *I2Cn_CTRL.hs_en*
- *I2Cn_RXCTRL1.cnt*
- *I2Cn_MSTCTRL.ex_addr_en*
- *I2Cn_CLKLO.lo*
- *I2Cn_CLKHI.hi*
- *I2Cn_HSCLK.lo*
- *I2Cn_HSCLK.hi*

In contrast to the above set of register fields, the register fields below can be safely (re)programmed at any time:

- All interrupt flags and interrupt enable bits
- *I2Cn_TXCTRL0.thd_val*
- *I2Cn_RXCTRL0.thd_lvl*
- *I2Cn_TIMEOUT*.scl_to_val
- *I2Cn_DMA.rx_en*
- *I2Cn_DMA.tx_en*
- *I2Cn_FIFO.data*
- *I2Cn_MSTCTRL.start*
- *I2Cn_MSTCTRL.restart*
- *I2Cn_MSTCTRL.stop*

14.4.6.1 I²C Controller Mode Receiver Operation

When in controller mode, initiating a controller receiver operation begins with the following sequence:

1. Write the number of data bytes to receive to the I²C receive count field (*I2Cn_RXCTRL1.cnt*).
2. Write the I²C peripheral address byte to the *I2Cn_FIFO* register with the R/W bit set to 1.
3. Send a START condition by setting *I2Cn_MSTCTRL.start* = 1.
4. The peripheral address is transmitted by the controller from the *I2Cn_FIFO* register.
5. The I²C controller receives an ACK from the peripheral, and the controller sets the address ACK interrupt flag (*I2Cn_INTFL0.addr_ack* = 1).
6. The I²C controller receives data from the peripheral and automatically ACKs each byte. The software must retrieve this data by reading the *I2Cn_FIFO* register.
7. Once *I2Cn_RXCTRL1.cnt* data bytes have been received, the I²C controller sends a NACK to the peripheral and sets the Transfer Done Interrupt Status Flag (*I2Cn_INTFL0.done* = 1).
8. If *I2Cn_MSTCTRL.restart* or *I2Cn_MSTCTRL.stop* is set, then the I²C controller sends a repeated START or STOP, respectively.

14.4.6.2 I²C Controller Mode Transmitter Operation

When in controller mode, initiating a controller transmitter operation begins with the following sequence:

1. Write the I²C peripheral 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 transmit FIFO. (e.g., If the transmit FIFO size is 8 bytes, the software can write one address byte and seven data bytes before starting the transaction.)
3. Send a START condition by setting *I2Cn_MSTCTRL.start* = 1.
4. The controller transmits the peripheral address byte written to the *I2Cn_FIFO* register.
5. The I²C controller receives an ACK from the peripheral, and the controller sets the address ACK interrupt flag (*I2Cn_INTFL0.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 peripheral after each data byte.
   b. As the transfer proceeds, the software should refill the transmit FIFO by writing to the *I2Cn_FIFO* register as needed.
   c. If the transmit FIFO goes empty during this process, the controller pauses at the beginning of the byte and waits for the software to either write more data or instruct the controller to send a RESTART or STOP condition.
7. Once the software writes all the desired bytes to the *I2Cn_FIFO* register; the software should set either *I2Cn_MSTCTRL.restart* or *I2Cn_MSTCTRL.stop*.
8. Once the controller sends all the remaining bytes and empties the transmit FIFO, it sets *I2Cn_INTFL0.done* and proceeds to send out either a RESTART condition if *I2Cn_MSTCTRL.restart* was set or a STOP condition if *I2Cn_MSTCTRL.stop* was set.

14.4.6.3 I²C Multi-Controller Operation

The I²C protocol supports multiple controllers on the same bus. When the bus is free, two (or more) controllers might try to initiate communication simultaneously. This is a valid bus condition. If this occurs and the two controllers want to transmit different data and/or address different peripherals, only one controller can remain in controller mode and complete its transaction. The other controller must back off the transmission and wait until the bus is idle. This process by which the winning controller is determined is called bus arbitration.

The controller compares the data being transmitted on SDA to the value observed on SDA to determine which controller wins the arbitration for each address or data bit. If a controller attempts to transmit a 1 on SDA (that is, the controller lets SDA float) but senses a 0 instead, then that controller loses arbitration, and the other controller that sent a zero continues with the transaction. The losing controller cedes the bus by switching off its SDA and SCL drivers.

Note: This arbitration scheme works with any number of bus controllers: if more than two controllers begin transmitting simultaneously, the arbitration continues as each controller cedes the bus until only one controller remains transmitting. Data is not corrupted because as soon as each controller realizes it has lost the arbitration, it stops transmitting on SDA, leaving the following data bits sent on SDA intact.

If the I^{2}C controller peripheral detects it has lost the arbitration, it stops generating SCL; sets *I2Cn_INTFL0.arb_err*; sets *I2Cn_INTFL0.tx_lockout*, flushing any remaining data in the transmit FIFO; and clears *I2Cn_MSTCTRL.start*, *I2Cn_MSTCTRL.restart*, and *I2Cn_MSTCTRL.stop* to 0. As long as the peripheral is not addressed by the winning controller, the I^{2}C peripheral stays in controller mode (*I2Cn_CTRL.mst_mode = 1)*. If, *at any time,* another controller addresses this peripheral using the address programmed in I2Cn_SLAVE0.*addr*, then the I^{2}C peripheral clears *I2Cn_CTRL.mst_mode* to 0 and begins responding as a peripheral. 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, sets I2Cn_INTFL0.tx_lockout. Therefore, after an arbitration loss, the software needs to clear I2Cn_INTFL0.tx_lockout and reload the transmit FIFO.

Also, in a multi-controller environment, the software does *not* need to wait for the bus to become free before attempting to start a transaction (writing 1 to *I2Cn_MSTCTRL.start*). If the bus is free when *I2Cn_MSTCTRL.start* is set to 1, the transaction begins immediately. If, instead, the bus is busy, then the peripheral:

1. Waits for the other controller to complete the transaction(s) by sending a STOP,
2. Counts out the bus free time using $t_{BUF} = t_{SCL\_LO}$ (see *Equation 14-3), and then*
3. Sends a START condition and begin transmitting the peripheral address byte(s) in the transmit FIFO, followed by the rest of the transfer.

The I^{2}C controller peripheral is compliant with all bus arbitration and clock synchronization requirements of the I^{2}C specification; this operation is automatic, and no additional programming is required.

14.4.7 Peripheral Mode Operation

When in peripheral mode, the I^{2}Cn peripheral operates as a peripheral device on the I^{2}C bus and responds to an external controller's requests to transmit or receive data. To configure the I^{2}Cn peripheral as a peripheral, write the *I2Cn_CTRL.mst_mode* bit to zero. The controller drives the I2Cn clock on the bus, so the SCL device pin is driven by the external controller, and *I2Cn_STATUS.mst_busy* remains a zero. The desired peripheral address must be set by writing to the *I2Cn_SLAVE0.addr* register.

For peripheral mode operation, the following register fields should be configured with the I2Cn peripheral disabled:

- *I2Cn_CTRL.mst_mode* = 0 for peripheral operation.
- Set the *I2Cn_SLAVE0.addr* for to a valid 7-bit or 10-bit I^{2}C address.
- Set the *I2Cn_SLAVE0.ext_addr_en* field to select either 7-bit or 10-bit addressing.
- *I2Cn_CTRL.gc_addr_en*
- *I2Cn_CTRL.irxm_en*
    - The recommended value for this field is 0. *Note that a setting of 1 is incompatible with peripheral mode operation with clock stretching disabled (I2Cn_CTRL.clkstr_dis = 1).*
- *I2Cn_CTRL.clkstr_dis*
- *I2Cn_CTRL.hs_en*
- *I2Cn_RXCTRL0.dnr*
    - SMBus/PMBus applications should set this to 0, while other applications should set this to 1.
- *I2Cn_TXCTRL0.nack_flush_dis*
- *I2Cn_TXCTRL0.rd_addr_flush_dis*
- *I2Cn_TXCTRL0.wr_addr_flush_dis*
- *I2Cn_TXCTRL0.gc_addr_flush_dis*
- *I2Cn_TXCTRL0.preload_mode*
    - The recommended value is 0 for applications that can tolerate peripheral clock stretching (*I2Cn_CTRL.clkstr_dis* = 0).
    - The recommended value is 1 for applications that do not allow peripheral clock stretching (*I2Cn_CTRL.clkstr_dis* = 1).
- *I2Cn_CLKHI.hi*
    - Applies to peripheral mode when clock stretching is enabled (*I2Cn_CTRL.clkstr_dis* = 0)
        - This is used to satisfy $t_{SU;DAT}$ after clock stretching; program it so that the value defined by *Equation 14-2* is >= $t_{SU;DAT(min)}$.
- *I2Cn_HSCLK.hi*
    - Applies to peripheral mode in Hs Mode when clock stretching is enabled (*I2Cn_CTRL.clkstr_dis* = 0)
        - This is used to satisfy $t_{SU;DAT}$ after clock stretching during Hs-Mode operation; program it so that the value defined by *Equation 14-6* is >= $t_{SU;DAT(min)}$.

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_TXCTRL0.thd_val* and *I2Cn_RXCTRL0.thd_lvl*
    - Transmit and receive FIFO threshold levels.
- *I2Cn_TXCTRL0.tx_ready_mode*
    - Transmit ready (can only be cleared by hardware).
- *I2Cn_TIMEOUT*.scl_to_val
    - Timeout control.
- *I2Cn_DMA.rx_en* and *I2Cn_DMA.tx_en*
    - Transmit and receive DMA enables.
- *I2Cn_FIFO.data*
    - FIFO access register.

14.4.7.1 Peripheral Transmitter

The device operates as a peripheral transmitter when the received address matches the device peripheral address with the R/W bit set to 1. The controller is then reading from the device peripheral. There two main modes of peripheral transmitter operation: just-in-time mode and preload mode.

14.4.7.1.1 Just-in-Time Peripheral Transmitter

In just-in-time mode, the software waits to write the transmit data to the transmit FIFO until after the controller addresses it for a READ transaction, just in time, to send the data to the controller. This allows the software to defer the determination of what data should be sent until the time of the address match. For example, the transmit data could be based on an immediately preceding I2C write transaction that requests a certain block of data to be sent. The data could represent the latest, most up-to-date value of a sensor reading. Clock stretching *must* be enabled (*I2Cn_CTRL*.*clkstr_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.en* = 0, initialize all relevant registers, including:

   a. Set the *I2Cn_SLAVE0.addr* field with the desired I^{2}C peripheral address.

   b. Set the *I2Cn_SLAVE0.ext_addr_en* for either 7-bit or 10-bit addressing.

   c. Just-in-time mode specific settings:

      i) *I2Cn_CTRL. clkstr_dis* = 0
      ii) *I2Cn_TXCTRL0*[5:2] = 0x8
      iii) *I2Cn_TXCTRL0.preload_mode* = 0.

   d. Program *I2Cn_CLKHI.hi* and *I2Cn_HSCLK.hi* with appropriate values satisfying $t_{SU;DAT}$ (and HS $t_{SU;DAT}$).

2. The software sets *I2Cn_CTRL.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 responds to its address with an ACK.

   b. When the address match occurs, the hardware sets *I2Cn_INTFL0.addr_match* and *I2Cn_INTFL0.tx_lockout*.

3. The software waits for *I2Cn_INTFL0.addr_match* to read 1, either through polling the interrupt flag or setting *I2Cn_INTEN0.addr_match* to interrupt the CPU.

4. After reading *I2Cn_INTFL0.addr_match* =1, the software reads *I2Cn_CTRL.read* to determine whether the transaction is a transmit (read = 1) or receive (read = 0) operation. In this case, assume read = 1, indicating transmit.

   a. The hardware holds SCL low until the software clears *I2Cn_INTFL0.tx_lockout* and loads data into the FIFO.

5. The software clears *I2Cn_INTFL0.addr_match* and *I2Cn_INTFL0.tx_lockout*. Now that *I2Cn_INTFL0.tx_lockout* is 0, the software can begin loading the transmit data into *I2Cn_FIFO*.

6. As soon as there is data in the FIFO, the hardware releases SCL (after counting out *I2Cn_CLKHI.hi*) and sends out the data on the bus.

7. While the controller keeps requesting data and sending ACKs, *I2Cn_INTFL0.done* remains 0, and the software should continue to monitor the transmit FIFO and refill it as needed.

   a. The FIFO level can be monitored synchronously through the transmit FIFO status/interrupt flags or asynchronously by setting *I2Cn_TXCTRL0.thd_val* and setting the *I2Cn_INTEN0.tx_thd* interrupt.

   b. If the transmit FIFO ever empties during the transaction, the hardware starts clock stretching and waits for it to be refilled.

8. The controller ends the transaction by sending a NACK. Once this happens, the *I2Cn_INTFL0.done* interrupt flag is set, and the software can stop monitoring the transmit FIFO.

   a. If the software needs to know how many data bytes were transmitted to the controller, it should check the transmit FIFO level as soon as *I2Cn_INTFL0.done* = 1 and use it to determine how many data bytes were successfully sent.

      1) Note: Any data remaining in the transmit FIFO is discarded before the next transmit operation; it is NOT necessary for the software to manually flush the transmit FIFO.

9. The transaction is complete. The software should clear the *I2Cn_INTFL0.done* interrupt flag and clear the *I2Cn_INTFL0.tx_thd* interrupt flag. Return to step 3, waiting on an address match.

14.4.7.1.2 Preload Mode Peripheral Transmit

The other mode of operation for peripheral transmit is preload mode. In this mode, it is assumed that the software knows before the transmit operation what data it should send to the controller. This data is then "preloaded" into the transmit 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 peripheral transmit preload mode:

1. With *I2Cn_CTRL.en* = 0, initialize all relevant registers, including:
    - a. Set the *I2Cn_SLAVE0.addr* field with the desired I^{2}C peripheral address.
    - b. Set the *I2Cn_SLAVE0.ext_addr_en* for either 7-bit or 10-bit addressing.
    - c. Preload mode specific settings:
        - i) *I2Cn_CTRL.clkstr_dis* = 1
        - ii) *I2Cn_TXCTRL0*[5:2] = 0xF
        - iii) *I2Cn_TXCTRL0.preload_mode* = 1.
2. The software sets *I2Cn_CTRL.en* = 1.
    - a. Even though the controller is enabled, it does not ACK an address match with R/W equal to 1 until the software sets the *I2Cn_TXCTRL1.preload_rdy* field to 1.
3. The software prepares for the transmit operation by loading data into the transmit FIFO, enabling DMA, setting *I2Cn_TXCTRL0.thd_val,* and setting *I2Cn_INTEN0.tx_thd* interrupt, etc.
    - a. If clock stretching is disabled, an empty transmit FIFO during the transmit operation causes a transmit underrun error. Therefore, the software should take any necessary steps to avoid an underrun *before* setting *I2Cn_TXCTRL1.preload_rdy* = 1.
    - b. If clock stretching is enabled, then an empty transmit FIFO does not cause a transmit underrun error. However, it is recommended to follow the same preparation steps to minimize the amount of time spent clock stretching, which lets the transaction complete as quickly as possible.
4. Once the software has prepared for the transmit operation; it sets *I2Cn_TXCTRL1.preload_rdy* = 1.
    - a. The controller is now fully enabled and responds with an ACK to an address match.
    - b. The hardware sets *I2Cn_INTFL0.addr_match* when an address match occurs. *I2Cn_INTFL0.tx_lockout* is NOT set to 1 and remains 0.
5. The software waits for *I2Cn_INTFL0.addr_match* = 1, either through polling the interrupt flag or setting *I2Cn_INTEN0.addr_match* to 1 to interrupt the CPU.
6. After seeing *I2Cn_INTFL0.addr_match* =1, the software reads *I2Cn_CTRL.read* to determine if the transaction is a transmit (read = 1) or receive (read = 0) operation. In this case, assume *I2Cn_CTRL.read*, indicating a transmit.
    - a. The hardware begins sending out the data that was preloaded into the transmit FIFO.
    - b. Once the first data byte is sent, the hardware automatically clears *I2Cn_TXCTRL1.preload_rdy* to 0.
7. While the controller keeps requesting data and sending ACKs, *I2Cn_INTFL0.done* remains 0, and the software should continue to monitor the transmit FIFO and refill it as needed.
    - a. The FIFO level can be monitored synchronously through the transmit FIFO status/interrupt flags or asynchronously by setting *I2Cn_TXCTRL0.thd_val* and setting *I2Cn_INTEN0.tx_thd* interrupt.
    - b. If clock stretching is disabled and the transmit FIFO empties during the transaction, the hardware sets *I2Cn_INTFL1.tx_un* = 1 and sends 0xFF for all following data bytes requested by the controller.
8. The controller ends the transaction by sending a NACK, causing the hardware to set the *I2Cn_INTFL0.done* interrupt flag.
    - a. If the transmit FIFO empties simultaneously that the controller indicates the transaction is complete by sending a NACK, this is not considered an underrun event *I2Cn_INTFL1.tx_un* flag remains 0.
    - b. If the software needs to know how many data bytes were transmitted to the controller, check the transmit FIFO level when the *I2Cn_INTFL0.done* flag is set to 1.
9. The transaction is complete, the software should "clean up," which should include clearing *I2Cn_INTFL0.done*. Return to step 3 and prepare for the next transaction.
    - a. Any data remaining in the transmit FIFO is not discarded; it is reused for the next transmit operation.
        - 1) If this is not desired, the software can flush the transmit FIFO. Flush the transmit and receive FIFOs by writing 0 to *I2Cn_CTRL.en* and the writing 1 to *I2Cn_CTRL.en*.

Once a peripheral starts transmitting from the *I2Cn_FIFO*, detecting out of sequence STOP, START, or RESTART conditions terminates the current transaction. When a transaction is terminated due to an out of sequence error, *I2Cn_INTFL0.start_err* or *I2Cn_INTFL0.stop_err* is set to 1.

If the transmit FIFO is not ready (*I2Cn_TXCTRL1.preload_rdy* = 0) and the I^{2}C controller receives a data read request from the controller, 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 I^{2}C read transaction is after the address byte.

14.4.7.2 Peripheral Receivers

The device operates as a peripheral receiver when the received address matches the device peripheral address with the R/W bit set to 0. The external controller is writing to the peripheral.

Program flow for a receive operation is as follows:

1. With *I2Cn_CTRL.en* = 0, initialize all relevant registers, including:
   a. Set the *I2Cn_SLAVE0.addr* field with the desired I^{2}C peripheral address.
   b. Set the *I2Cn_SLAVE0.ext_addr_en* for either 7-bit or 10-bit addressing.
2. Set *I2Cn_CTRL.en* = 1.
   a. If an address match with the R/W bit equal to zero occurs, and the receive FIFO is empty, the peripheral responds with an ACK, and the *I2Cn_INTFL0.addr_match* flag is set.
   b. If the receive FIFO is not empty, then depending on the value of *I2Cn_RXCTRL0.dnr*, the peripheral NACKs either the address byte (*I2Cn_RXCTRL0.dnr* = 1) or the first data byte (*I2Cn_RXCTRL0.dnr* = 0).
3. Wait for *I2Cn_INTFL0.addr_match* = 1, either by polling or by enabling the *wr_addr_match* interrupt. Once a successful address match occurs, the hardware sets *I2Cn_INTFL0.addr_match* = 1.
4. Read *I2Cn_CTRL.read* to determine if the transaction is a transmit (*I2Cn_CTRL.read* = 1) or a receive (*I2Cn_CTRL.read* = 0) operation. In this case, assume *I2Cn_CTRL.read* = 0, indicating receive. The device begins receiving data into the receive FIFO.
5. Clear *I2Cn_INTFL0.addr_match*, and while the controller keeps sending data, *I2Cn_INTFL0.done* remains 0, and the software should continue to monitor the receive FIFO and empty it as needed.
   a. The FIFO level can be monitored synchronously through the receive FIFO status/interrupt flags or asynchronously by setting *I2Cn_RXCTRL0.thd_lvl* and enabling the *I2Cn_INTFL0.rx_thd* interrupt.
   b. If the receive FIFO ever fills up during the transaction, then the hardware sets *I2Cn_INTFL1.rx_ov* and then either:
      i. If *I2Cn_CTRL.clkstr_dis* = 0, start clock stretching and wait until the software reads from the receive FIFO, *or*
      ii. If *I2Cn_CTRL.clkstr_dis* = 1, respond to the controller with a NACK, and the last byte is discarded.
6. The controller ends the transaction by sending a RESTART or STOP. Once this happens, the *I2Cn_INTFL0.done* interrupt flag is set, and the software can stop monitoring the receive FIFO.
7. Once a peripheral starts receiving into its receive FIFO, detection of an out of sequence STOP, START, or RESTART condition releases the I^{2}C bus to the Idle state, and the hardware sets the *I2Cn_INTFL0.start_err* field or *I2Cn_INTFL0.stop_err* field to 1 based on the specific condition.

If the software has not emptied the data in the receive FIFO from the previous transaction by the time a controller addresses it for another write (i.e., receive) transaction, then the controller does *not* participate in the transaction, and no additional data is written into the FIFO. Although a NACK *is* sent to the controller, the software can control if the NACK is sent with the initial address match or sent at the end of the first data byte. Setting *I2Cn_RXCTRL0.dnr* to 1 chooses the former while setting *I2Cn_RXCTRL0.dnr* to 0 chooses the latter.

14.4.8 Interrupt Sources

The I^{2}C controller has a very flexible interrupt generator that generates an interrupt signal to the interrupt controller on any of several events. On recognizing the I^{2}C interrupt, the software determines the cause of the interrupt by reading the I^{2}C interrupt flags registers *I2Cn_INTFL0* and *I2Cn_INTFL1*. Interrupts can be generated for the following events:

- Transaction Complete (controller/peripheral).
- Address NACK received from peripheral (controller).
- Data NACK received from peripheral (controller).
- Lost arbitration (controller).
- Transaction timeout (controller/peripheral).
- FIFO is empty, not empty, or full to a configurable threshold level (controller/peripheral).
- Transmit FIFO locked out because it is being flushed (controller/peripheral)
- Out of sequence START and STOP conditions (controller/peripheral).
- Sent a NACK to an external controller because the transmit or receive FIFO was not ready (peripheral).
- Address ACK or NACK received (controller).
- Incoming address match (peripheral)
- Transmit underflow or receive overflow (peripheral).

Interrupts for each event can be enabled or disabled by setting or clearing the corresponding bit in the *I2Cn_INTEN0* or *I2Cn_INTEN1* interrupt enable register.

Note: Disabling the interrupt does not prevent the corresponding flag from being set by the hardware but does prevent an interrupt when the interrupt flag is set.

Note: Before enabling an interrupt, the status of the corresponding interrupt flag should be checked and, if necessary, serviced or cleared, preventing a previous interrupt event from interfering with a new I^{2}C communications session.

14.4.9 Transmit FIFO and Receive FIFO

There are separate transmit and receive FIFOs. Both are accessed using the FIFO data register *I2Cn_FIFO*. Writes to this register enqueue data into the transmit FIFO. Writes to a full transmit FIFO has no effect. Reads from *I2Cn_FIFO* dequeue data from the receive FIFO. Writes to a full transmit FIFO has no effect and reads from an empty receive FIFO return 0xFF.

The transmit and receive FIFO only read or write one byte at a time. Transactions greater than 8 bits can still be performed, however. A 16- or 32-bit write to the transmit FIFO stores just the lowest 8 bits of the write data. A 16- or 32-bit read from the receive FIFO has the valid data in the lowest 8 bits and zeros in the upper bits. In any case, the transmit and receive FIFOs only accept 8 bits at a time for either read or write.

To offload work from the CPU, the DMA can read and write to each FIFO. See *DMA Control* for more information on configuring the DMA.

During a receive transaction (which during controller operation is a READ, and during peripheral operation is a WRITE), received bytes are automatically written to the receive FIFO. The software should monitor the receive FIFO level and unload data from it as needed by reading *I2Cn_FIFO*. If the receive FIFO becomes full during a controller mode transaction, then the hardware sets the *I2Cn_INTFL1.rx_ov* the *I2Cn_INTFL1.rx_ov* bit, and one of two things occur depending on the value of *I2Cn_CTRL.clkstr_dis*:

- If clock stretching is enabled (*I2Cn_CTRL.clkstr_dis* = 0), then the hardware stretches the clock until the software makes space available in the receive FIFO by reading *I2Cn_FIFO*. Once space is available, the hardware moves the data byte from the shift register into the receive FIFO, the SCL device pin is released, and the controller is free to continue the transaction.

- If clock stretching is disabled (*I2Cn_CTRL.clkstr_dis* = 1), the hardware responds to the controller with a NACK, and the data byte is lost. The controller 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 controller operation is a WRITE, and during peripheral operation is a READ), either the software or the DMA can provide data to be transmitted by writing to the transmit FIFO. Once the peripheral finishes transmitting each byte, it removes it from the transmit FIFO and, if available, begins transmitting the next byte.

Interrupts can be generated for the following FIFO status:

- Transmit FIFO level less than or equal to the threshold.
- Receive FIFO level greater than or equal to the threshold.
- Transmit FIFO underflow.
- Receive FIFO overflow.
- Transmit FIFO locked for writing.

Both the receive FIFO and transmit FIFO are flushed when the I2Cn port is disabled by clearing *I2Cn_CTRL.en* to 0. While the peripheral is disabled, writes to the transmit FIFO have no effect and reads from the receive FIFO return 0xFF.

The transmit FIFO and receive FIFO can be flushed by setting the transmit FIFO flush bit (*I2Cn_TXCTRL0.flush*=1) or the receive FIFO flush bit (*I2Cn_RXCTRL0.flush*=1), respectively. In addition, under certain conditions, the transmit FIFO is automatically locked by the hardware and flushed so stale data is not unintentionally transmitted. The transmit FIFO is automatically flushed and writes locked out from the software under the following conditions:

- General Call Address Match: Automatic flushing and lockout can be disabled by setting *I2Cn_TXCTRL0.gc_addr_flush_dis*.
- Peripheral Address Match Write: Automatic flushing and lockout can be disabled by setting *I2Cn_TXCTRL0.wr_addr_flush_dis*.
- Peripheral Address Match Read: Automatic flushing and lockout can be disabled by setting *I2Cn_TXCTRL0.rd_addr_flush_dis*.
- During operation as a peripheral transmitter, a NACK is received. Automatic flushing and lockout can be disabled by setting *I2Cn_TXCTRL0.nack_flush_dis*.
- Any of the following interrupts:
  - Arbitration error, timeout error, controller mode address NACK error, controller mode data NACK error, start error, and stop error. Automatic flushing cannot be disabled for these conditions.

When the above conditions occur, the transmit FIFO is flushed so that data intended for a previous transaction is not transmitted unintentionally for a new transaction. In addition to flushing the transmit FIFO, the transmit lockout flag is set (*I2Cn_INTFL0.tx_lockout* = 1) and writes to the transmit FIFO are ignored until the software acknowledges the external event by clearing *I2Cn_INTFL0.tx_lockout*.

14.4.10 Transmit FIFO Preloading

There can be situations during peripheral mode operation where the software wants to preload the transmit FIFO before a transmission, such as when clock stretching is disabled. In this scenario, rather than responding to an external controller requesting data with an ACK and clock stretching while the software writes the data to the transmit FIFO, the hardware responds with a NACK until the software has preloaded the requested data into the transmit FIFO.

When transmit FIFO preloading is enabled, the software controls ACKs to the external controller using the transmit ready (*I2Cn_TXCTRL1.preload_rdy*) bit. When *I2Cn_TXCTRL1.preload_rdy* is set to 0, the hardware automatically NACKs all read transactions from the controller. Setting *I2Cn_TXCTRL1.preload_rdy* to 1 sends an ACK to the controller on the next read transaction and transmits the data in the transmit FIFO. Preloading the transmit FIFO should be complete before setting the *I2Cn_TXCTRL1.preload_rdy* field to 1.

The required steps for implementing transmit FIFO preloading in software are as follows:

1. Enable the transmit FIFO preloading by setting the field *I2Cn_TXCTRL0.preload_mode* to 1. The hardware automatically clears the *I2Cn_TXCTRL1.preload_rdy* field to 0.
2. If the transmit FIFO lockout flag (*I2Cn_INTFL0.tx_lockout*) is set to 1, write 1 to clear the flag and enable writes to the transmit FIFO.
3. Enable DMA or interrupts if required.
4. Load the transmit FIFO with the data to send when the controller sends the next read request.
5. Set *I2Cn_TXCTRL1.preload_rdy* to 1 to automatically let the hardware send the preloaded FIFO on the next read from a controller.
6. *I2Cn_TXCTRL1.preload_rdy* is cleared by the hardware once it finishes transmitting the first byte, and data is transmitted from the transmit FIFO. Once cleared, the software can repeat the preloading process or disable transmit FIFO preloading.

Note: To prevent the preloaded data from being cleared when the controller tries to read it, the software must at least set I2Cn_TXCTRL0.rd_addr_flush_dis to 1, disabling auto flush on READ address match. The software determines if the other auto flush disable bits should be set. For example, if a controller uses I^{2}C WRITE transactions to determine what data the peripheral should send in the following READ transactions, the software can clear I2Cn_TXCTRL0.wr_addr_flush_dis to 0. When a WRITE occurs, the transmit FIFO is flushed, giving the software time to load the new data. For the READ transaction, the external controller can poll the peripheral address until the new data has been loaded and I2Cn_TXCTRL1.preload_rdy is set, at which point the peripheral responds with an ACK.

14.4.11 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. IRXM is enabled by setting *I2Cn_CTRL.irxm_en* = 1. If IRXM is enabled, it must occur before any I^{2}C transfer is initiated.

When 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_INTFL0.irxm* = 1). Software must read the data and generate a response (ACK or NACK) by setting the IRXM Acknowledge (*I2Cn_CTRL.irxm_ack*) bit accordingly. Send an ACK by clearing the *I2Cn_CTRL.irxm_ack* bit to 0. Send a NACK by setting the *I2Cn_CTRL.irxm_ack* bit to 1.

After setting the *I2Cn_CTRL.irxm_ack* bit, clear the IRXM interrupt flag. Write 1 to *I2Cn_INTFL0.irxm* 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.irxm_ack* on the SDA line.

While the I2Cn peripheral is waiting for the software to clear the *I2Cn_INTFL0.irxm* flag, the software can disable IRXM and, if operating as a controller, load the remaining number of bytes to be received for the transaction. This allows the software to examine the initial bytes of a transaction, which might be a command, and then disable IRXM to receive the remaining bytes in normal operation.

During IRXM, received data is not placed in the receive FIFO. Instead, the *I2Cn_FIFO* address is repurposed to directly read the receive shift register, bypassing the receive FIFO. Therefore, before disabling IRXM, the software 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 receive FIFO returns 0xFF.

Note: IRXM only applies to data bytes and does not apply to address bytes, general call address responses, or START byte responses.

Note: When enabling IRXM and operating as a peripheral, clock stretching must remain enabled (I2Cn_CTRL.clkstr_dis = 0).

14.4.12 Clock Stretching

When the I2Cn peripheral requires some response or intervention from the software to continue with a transaction, it holds SCL low, preventing the transfer from continuing. This is called 'clock stretching' or 'stretching the clock.' While the I^{2}C Bus Specification defines the term 'clock stretching' to only apply to a peripheral device holding the SCL line low, this section describes situations where the I2Cn peripheral holds the SCL line low in either peripheral *or* controller 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 receive FIFO during a receive operation or an empty transmit 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 IRXM (*I2Cn_CTRL.irxm_en* = 1), the peripheral can also clock stretch *before* the ACK bit, allowing the software to decide if to send an ACK or NACK.

For a transmit operation (as either controller or peripheral), when the transmit FIFO is empty, SCL is automatically held low after the ACK bit and before the next data byte begins. The software must write data to *I2Cn_FIFO.data* to stop clock stretching and continue the transaction. However, if operating in controller mode instead of sending more data, the software can also set either *I2Cn_MSTCTRL.stop* or *I2Cn_MSTCTRL.restart* to send a STOP or RESTART condition, respectively.

For a receive operation (as either controller or peripheral), when both the receive FIFO and the receive shift register are full, SCL is automatically held low until at least one data byte is read from the receive FIFO. The software must read data from *I2Cn_FIFO.data* to stop clock stretching and continue the transaction. If operating in controller mode and this is the final byte of the transaction, as determined by *I2Cn_RXCTRL1.cnt*, the software must also set either *I2Cn_MSTCTRL.stop* or *I2Cn_MSTCTRL.restart* to send a STOP or RESTART condition, respectively. This must be done in addition to reading from the receive FIFO since the peripheral cannot start sending the STOP or RESTART until the last data byte has been moved from the receive shift register into the receive FIFO. This occurs automatically once there is space in the receive FIFO.

Note: Since some controllers do not support other devices stretching the clock, it is possible to completely disable all clock stretching during peripheral mode by setting I2Cn_CTRL.clkstr_dis to 1 and clearing I2Cn_CTRL.irxm_en 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 controller or peripheral devices stretching the clock is built into the peripheral and requires no intervention from the software to operate correctly.

14.4.13 Bus Timeout

The timeout field, *I2Cn_TIMEOUT.scl_to_val,* is used to detect bus errors. *Equation 14-8* and *Equation 14-9* show equations for calculating the maximum and minimum timeout values based on the value loaded into the *I2Cn_TIMEOUT.scl_to_val* field.

Equation 14-8: I^{2}C Timeout Maximum

$$t_{TIMEOUT} \le \left(\frac{1}{f_{I2C\_CLK}}\right) \times \left((I2Cn\_TIMEOUT.scl\_to\_val \times 32) + 3\right)$$

Due to clock synchronization, the timeout is guaranteed to meet the following minimum time calculation shown in *Equation 14-9*.

Equation 14-9: I^{2}C Timeout Minimum

$$t_{TIMEOUT} \le \left(\frac{1}{f_{I2C\_CLK}}\right) \times \left((I2Cn\_TIMEOUT.scl\_to\_val \times 32) + 2\right)$$

The timeout feature is disabled when *I2Cn_TIMEOUT.scl_to_val* = 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_INTFL0.to_err* = 1).

For applications where the device can 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.scl_to_val* = 0).

14.4.14 DMA Control

There are independent DMA channels for each transmit FIFO, and each receive FIFO. DMA activity is triggered by the transmit FIFO (*I2Cn_TXCTRL0.thd_val*) and receive FIFO (*I2Cn_RXCTRL0.thd_lvl*) threshold levels.

When the transmit FIFO byte count (*I2Cn_TXCTRL1.lvl*) is less than or equal to the transmit FIFO threshold level *I2Cn_TXCTRL0.thd_val*, then the DMA transfers data into the transmit FIFO according to the DMA configuration.

The DMA burst size should be set as follows to ensure the DMA does not overflow the transmit FIFO:

Equation 14-10: DMA Burst Size Calculation for I^{2}C Transmit

$$DMA\ Burst\ Size\ \leq TX\ FIFO\ Depth - I2Cn\_TXCTRL0.thd\_val = 8 - I2Cn\_TXCTRL0.thd\_val$$

$$where\ 0\ \leq I2Cn\_TXCTRL0.thd\_val\ \leq 7$$

Software trying to avoid transmit underflow and/or clock stretching should use a smaller burst size and higher *I2Cn_TXCTRL0.thd_val* setting. This fills up the FIFO more frequently but increases internal bus traffic.

When the receive FIFO count (*I2Cn_RXCTRL1.lvl*) is greater than or equal to the receive FIFO threshold level *I2Cn_RXCTRL0.thd_lvl*, the DMA transfers data out of the receive FIFO according to the DMA configuration. The DMA burst size should be set as follows to ensure the DMA does not underflow the receive FIFO:

Equation 14-11: DMA Burst Size Calculation for I^{2}C Receive

$$DMA\ Burst\ Size \leq I2Cn\_RXCTRL0.thd\_lvl$$

$$where\ 1 \leq I2Cn\_RXCTRL0.thd\_lvl \leq 8$$

Applications trying to avoid receive overflow and/or clock stretching should use a smaller burst size and lower *I2Cn_RXCTRL0.thd_lvl*. 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_RXCTRL0.thd_lvl. Otherwise, the receive transaction ends with some data still in the receive 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_RXCTRL0.thd_lvl = 1).

Enable the transmit DMA channel (*I2Cn_DMA.tx_en*) and/or the receive DMA channel (*I2Cn_DMA.rx_en*) to enable DMA transfers.

14.5 Registers

See *Table 3-3* 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 14-1*. Register names for a specific instance are defined by replacing "n" with the instance number. For example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.
See *Table 1-1* 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 14-5: Register Summary

Offset	Register	Description
[0x0000]	*I2Cn_CTRL*	I^2C Control Register
[0x0004]	*I2Cn_STATUS*	I^2C Status Register
[0x0008]	*I2Cn_INTFL0*	I^2C Interrupt Flags 0 Register
[0x000C]	*I2Cn_INTEN0*	I^2C Interrupt Enable 0 Register
[0x0010]	*I2Cn_INTFL1*	I^2C Interrupt Flags 1 Register
[0x0014]	*I2Cn_INTEN1*	I^2C Interrupt Enable 1 Register
[0x0018]	*I2Cn_FIFOLEN*	I^2C FIFO Length Register
[0x001C]	*I2Cn_RXCTRL0*	I^2C Receive Control 0 Register
[0x0020]	*I2Cn_RXCTRL1*	I^2C Receive Control 1 Register
[0x0024]	*I2Cn_TXCTRL0*	I^2C Transmit Control 0 Register
[0x0028]	*I2Cn_TXCTRL1*	I^2C Transmit Control 1 Register
[0x002C]	*I2Cn_FIFO*	I^2C Transmit and Receive FIFO Register
[0x0030]	*I2Cn_MSTCTRL*	I^2C Controller Control Register
[0x0034]	*I2Cn_CLKLO*	I^2C Clock Low Time Register
[0x0038]	*I2Cn_CLKHI*	I^2C Clock High Time Register
(0x003C)	*I2Cn_HSCLK*	I^2C Hs-Mode Clock Control Register
[0x0040]	*I2Cn_TIMEOUT*	I^2C Timeout Register
[0x0048]	*I2Cn_DMA*	I^2C DMA Enable Register
[0x004C]	*I2Cn_SLAVE0*	I^2C Peripheral Address 0 Register
[0x0050]	*I2Cn_SLAVE1*	I^2C Peripheral Address 1 Register
[0x0054]	*I2Cn_SLAVE2*	I^2C Peripheral Address 2 Register
[0x0058]	*I2Cn_SLAVE3*	I^2C Peripheral Address 3 Register

14.5.1 Register Details

Table 14-6: I^2C Control Register

<table>
<tr><th colspan="3">I²C Control</th><th colspan="2">I2Cn_CTRL</th><th>[0x0000]</th></tr>
<tr><th>Bits</th><th>Field</th><th>Access</th><th>Reset</th><th colspan="2">Description</th></tr>
<tr><td>31:16</td><td>-</td><td>RO</td><td>0</td><td colspan="2"><b>Reserved</b></td></tr>
<tr><td>15</td><td>hs_en</td><td>R/W</td><td>0</td><td colspan="2"><b>Hs-Mode Enable</b>
I²C high speed mode operation
0: Disabled.
1: Enabled.</td></tr>
<tr><td>14</td><td>-</td><td>RO</td><td>0</td><td colspan="2"><b>Reserved</b></td></tr>
</table>

I^{2}C Control			I2Cn_CTRL	[0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**
13	one_mst_mode	R/W	0	**Single Controller Only** When set to 1, the device MUST ONLY be used in a single controller application with peripheral devices that are NOT going to hold SCL low (i.e., the peripheral devices never clock stretch).
12	clkstr_dis	R/W	0	**Peripheral Mode Clock Stretching** 0: Enabled. 1: Disabled.
11	read	R	0	**Peripheral Read/Write Bit Status** Returns the logic level of the R/W bit on a received address match (*I2Cn_INTFL0.addr_match* = 1) or general call match (*I2Cn_INTFL0.gc_addr_match* = 1). This bit is valid for three system clock cycles after the address match status flag is set.
10	bb_mode	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 the hardware. 1: SDA and SCL are controlled by the software using the *I2Cn_CTRL.sda_out* and *I2Cn_CTRL.scl_out* 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. *Note: Only valid when I2Cn_CTRL.bb_mode=1*
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. *Note: Only valid when I2Cn_CTRL.bb_mode =1*
5	-	RO	0	**Reserved**
4	irxm_ack	R/W	0	**IRXM Acknowledge** If IRXM is enabled (*I2Cn_CTRL.irxm_en* = 1), this field determines if the hardware sends an ACK or a NACK to an IRXM transaction. 0: Respond to IRXM with ACK. 1: Respond to IRXM with NACK.
3	irxm_en	R/W	0	**IRXM Enable** When receiving data, this field allows for an IRXM 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 the *Interactive Receive Mode* section for detailed information. 0: Disabled. 1: Enabled. *Note: Only set this field when the I^{2}C bus is inactive.*

I^{2}C Control			I2Cn_CTRL	[0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**
2	gc_addr_en	R/W	0	**General Call Address Enable** 0: Ignore General Call Address. 1: Acknowledge General Call Address.
1	mst_mode	R/W	0	**Controller Mode Enable** 0: Peripheral mode enabled. 1: Controller mode enabled.
0	en	R/W	0	**I^{2}C Peripheral Enable** 0: Disabled. 1: Enabled.

Table 14-7: I^{2}C Status Register

I^{2}C Status			I2Cn_STATUS	[0x0004]
Bits	**Field**	**Access**	**Reset**	**Description**
31:6	-	RO	0	**Reserved**
5	mst_busy	RO	0	**Controller Mode I^{2}C Bus Transaction Active** The peripheral is operating in controller mode, and a valid transaction beginning with a START command is in progress on the I^{2}C bus. This bit reads 1 until the controller ends the transaction with a STOP command. This bit continues to read 1 while a peripheral performs clock stretching. 0: Device not actively driving SCL clock cycles. 1: Device operating as controller and actively driving SCL clock cycles.
4	tx_full	RO	0	**Transmit FIFO Full** 0: Not full. 1: Full.
3	tx_em	RO	1	**Transmit FIFO Empty** 0: Not empty. 1: Empty.
2	rx_full	RO	0	**Receive FIFO Full** 0: Not full. 1: Full.
1	rx_em	RO	1	**Receive FIFO Empty** 0: Not empty. 1: Empty.
0	busy	RO	0	**Controller or Peripheral Mode I^{2}C Busy Transaction Active** The peripheral is operating in controller or peripheral mode, and a valid transaction beginning with a START command is in progress on the I^{2}C bus. This bit reads 1 until the peripheral acting as a controller or an external controller ends the transaction with a STOP command. This bit continues to read 1 while a peripheral performs clock stretching. 0: I^{2}C bus is idle. 1: I^{2}C bus transaction in progress.

Table 14-8: I^{2}C Interrupt Flag 0 Register

I^{2}C Interrupt Flag 0			I2Cn_INTFL0	[0x0008]
Bits	**Field**	**Access**	**Reset**	**Description**
31:24	-	RO	0	**Reserved**

I^{2}C Interrupt Flag 0				I2Cn_INTFL0 [0x0008]
Bits	**Field**	**Access**	**Reset**	**Description**
23	wr_addr_match	R/W1C	0	**Peripheral Write Address Match Interrupt Flag** If set, the device has been accessed for a write (i.e., receive) transaction in peripheral mode, and the address received matches the device peripheral address. 0: No address match. 1: Address match.
22	rd_addr_match	R/W1C	0	**Peripheral Read Address Match Interrupt Flag** If set, the device has been accessed for a read (i.e., transmit) transaction in peripheral mode, and the address received matches the device peripheral address. 0: No address match. 1: Address match.
21:17	-	RO	0	**Reserved**
16	-	R/W1C	0	**MAMI Interrupt Flag**
15	tx_lockout	R/W1C	0	**Transmit FIFO Locked Interrupt Flag** If set, the transmit FIFO is locked, and writes to the transmit FIFO are ignored. When set, the transmit FIFO is automatically flushed. Writes to the transmit FIFO are ignored until this flag is cleared. Write 1 to clear. 0: Transmit FIFO not locked. 1: Transmit FIFO is locked, and all writes to the transmit FIFO are ignored.
14	stop_err	R/W1C	0	**Out of Sequence STOP Interrupt Flag** This flag is set if a STOP condition occurs out of the 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_err	R/W1C	0	**Out of Sequence START Interrupt Flag** This flag is set if a START condition occurs out of the 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	dnr_err	R/W1C	0	**Peripheral Mode Do Not Respond Interrupt Flag** This occurs if an address match is made, but the transmit FIFO or receive 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 transmit or receive FIFO is not configured.
11	data_err	R/W1C	0	**Controller Mode Data NACK from External Peripheral Interrupt Flag** The hardware sets this flag if a NACK is received from a peripheral. This flag is only valid if the I2Cn peripheral is configured for controller mode operation. Write 1 to clear. Write 0 has no effect. 0: Error condition has not occurred. 1: Data NACK received from a peripheral.
10	addr_nack_err	R/W1C	0	**Controller Mode Address NACK from Peripheral Error Flag** The hardware sets this flag if an Address NACK is received from a peripheral bus. This flag is only valid if the I2Cn peripheral is configured for controller mode operation. Write 1 to clear. Write 0 has no effect. 0: Error condition has not occurred. 1: Address NACK received from a peripheral.

I^{2}C Interrupt Flag 0			I2Cn_INTFL0	[0x0008]
Bits	**Field**	**Access**	**Reset**	**Description**
9	to_err	R/ W1C	0	**Timeout Error Interrupt Flag** This flag is set when this device holds SCL low longer than the programmed timeout value. This field's setting applies to both controller and peripheral mode. Write 1 to clear. Write 0 has no effect. 0: Timeout error has not occurred. 1: Timeout error occurred.
8	arb_err	R/ W1C	0	**Controller 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	**Controller Mode Address ACK from External Peripheral Interrupt Flag** This field is set when a peripheral address ACK is received. Write 1 to clear. Write 0 has no effect. 0: Condition has not occurred. 1: The peripheral device ACK for the address was received.
6	stop	R/ W1C	0	**Peripheral 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_thd	RO	1	**Transmit FIFO Threshold Level Interrupt Flag** The hardware sets this field 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 the hardware when the transmit FIFO contains fewer bytes than the transmit threshold level. 0: Transmit FIFO contains more bytes than the transmit threshold level. 1: Transmit FIFO contains the transmit threshold level or fewer bytes.
4	rx_thd	R/W1C	1	**Receive FIFO Threshold Level Interrupt Flag** The hardware sets this field 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 receive FIFO contains fewer bytes than the receive threshold setting. 0: Normal operation. 1: Receive FIFO contains at least receive threshold level of bytes.
3	addr_match	R/W1C	0	**Peripheral Mode Incoming Address Match Status Interrupt Flag** Write 1 to clear. Writing 0 has no effect. 0: Normal operation. 1: Peripheral address match occurred.
2	gc_addr_match	R/W1C	0	**Peripheral Mode General Call Address Match Received Interrupt Flag** Write 1 to clear. Writing 0 has no effect. 0: Normal operation. 1: General call address match occurred.
1	irxm	R/W1C	0	**Interactive Receive Mode Interrupt Flag** Write 1 to clear. Writing 0 is ignored. 0: Normal operation. 1: Interrupt condition occurred.

I^{2}C Interrupt Flag 0			I2Cn_INTFL0	[0x0008]
Bits	**Field**	**Access**	**Reset**	**Description**
0	done	R/W1C	0	**Transfer Complete Interrupt Flag** This flag is set for both controller and peripheral mode once a transaction completes. Write 1 to clear. Writing 0 has no effect. 0: Transfer is not complete. 1: Transfer complete.

Table 14-9: I^{2}C Interrupt Enable 0 Register

I^{2}C Interrupt Enable 0			I2Cn_INTEN0	[0x000C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:24	-	RO	0	**Reserved**
23	wr_addr_match	R/W	0	**Peripheral Write Address Match Interrupt Enable** This bit is set to enable interrupts when the device is accessed in peripheral mode, and the address received matches the device peripheral addressed for a write transaction. 0: Disabled. 1: Enabled.
22	rd_addr_match	R/W	0	**Peripheral Read Address Match Interrupt Enable** This bit is set to enable interrupts when the device is accessed in peripheral mode, and the address received matches the device peripheral addressed for a read transaction. 0: Disabled. 1: Enabled.
21:17	-	RO	0	**Reserved**
16	mami	R/W	0	**MAMI Interrupt Enable**
15	tx_lockout	R/W	0	**Transmit FIFO Lock Out Interrupt Enable** 0: Disabled. 1: Enabled.
14	stop_err	R/W	0	**Out of Sequence STOP Condition Detected Interrupt Enable** 0: Disabled. 1: Enabled.
13	start_err	R/W	0	**Out of Sequence START Condition Detected Interrupt Enable** 0: Disabled. 1: Enabled.
12	dnr_err	R/W	0	**Peripheral Mode Do Not Respond Interrupt Enable** Set this field to enable interrupts in peripheral mode when the "Do Not Respond" condition occurs. 0: Interrupt disabled. 1: Interrupt enabled.
11	data_err	R/W	0	**Controller Mode Received Data NACK from Peripheral Interrupt Enable** 0: Disabled. 1: Enabled.
10	addr_nack_err	R/W	0	**Controller Mode Received Address NACK from Peripheral Interrupt Enable** 0: Disabled. 1: Enabled.
9	to_err	R/W	0	**Timeout Error Interrupt Enable** 0: Disabled. 1: Enabled.

I^{2}C Interrupt Enable 0		I2Cn_INTEN0		[0x000C]
Bits	**Field**	**Access**	**Reset**	**Description**
8	arb_err	R/W	0	**Controller Mode Arbitration Lost Interrupt Enable** 0: Disabled. 1: Enabled.
7	addr_ack	R/W	0	**Received Address ACK from Peripheral Interrupt Enable** Set this field to enable interrupts for controller mode peripheral 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_thd	R/W	0	**Transmit FIFO Threshold Level Interrupt Enable** 0: Disabled. 1: Enabled.
4	rx_thd	R/W	0	**Receive FIFO Threshold Level Interrupt Enable** 0: Disabled. 1: Enabled.
3	addr_match	R/W	0	**Peripheral Mode Incoming Address Match Interrupt Enable** 0: Disabled. 1: Enabled.
2	gc_addr_match	R/W	0	**Peripheral Mode General Call Address Match Received Interrupt Enable** 0: Disabled. 1: Enabled.
1	irxm	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 14-10: I^{2}C Interrupt Flag 1 Register

I^{2}C Interrupt Status Flags 1		I2Cn_INTFL1		[0x0010]
Bits	**Field**	**Access**	**Reset**	**Description**
31:3	-	RO	0	**Reserved**
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.
1	tx_un	R/W1C	0	**Peripheral Mode Transmit FIFO Underflow Status Flag** In peripheral mode operation, the hardware sets this flag automatically if the transmit FIFO is empty and the controller requests more data by sending an ACK after the previous byte is transferred. 0: Peripheral mode transmit FIFO underflow condition has not occurred. 1: Peripheral mode transmit FIFO underflow condition occurred.
0	rx_ov	R/W1C	0	**Peripheral Mode Receive FIFO Overflow Status Flag** In peripheral mode operation, the hardware sets this flag automatically when a receive FIFO overflow occurs. Write 1 to clear. Writing 0 has no effect. 0: Peripheral mode receive FIFO overflow event has not occurred. 1: Peripheral mode receive FIFO overflow condition occurred (data lost).

Table 14-11: I²C Interrupt Enable 1 Register

I²C Interrupt Enable 1		I2Cn_INTEN1		[0x0014]
Bits	**Field**	**Access**	**Reset**	**Description**
31:3	-	RO	0	**Reserved**
2	start	R/W	0	**START Condition Interrupt Enable** 0: Disabled. 1: Enabled.
1	tx_un	R/W	0	**Peripheral Mode Transmit FIFO Underflow Interrupt Enable** 0: Disabled. 1: Enabled.
0	rx_ov	R/W	0	**Peripheral Mode Receive FIFO Overflow Interrupt Enable** 0: Disabled. 1: Enabled.

Table 14-12: I²C FIFO Length Register

I²C FIFO Length		I2Cn_FIFOLEN		[0x0018]
Bits	**Field**	**Access**	**Reset**	**Description**
31:16	-	RO	0	**Reserved**
15:8	tx_depth	RO	8	**Transmit FIFO Length** This field returns the depth of the transmit FIFO. 8: 8-bytes.
7:0	rx_depth	RO	8	**Receive FIFO Length** This field returns the depth of the receive FIFO. 8: 8-bytes.

Table 14-13: I²C Receive Control 0 Register

I²C Receive Control 0		I2Cn_RXCTRL0		[0x001C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:12	-	RO	0	**Reserved**
11:8	thd_lvl	R/W	0	**Receive FIFO Threshold Level** Set this field to the required number of bytes to trigger a receive FIFO threshold event. When the number of bytes in the receive FIFO is equal to or greater than this field, the hardware sets the *I2Cn_INTFL0.rx_thd* bit indicating a receive FIFO threshold level event. 0: 0 bytes or more in the receive FIFO causes a threshold event. 1: 1+ bytes in the receive FIFO triggers a receive threshold event (recommended minimum value). … 8: Receive FIFO threshold event only occurs when the receive FIFO is full.
7	flush	R/W1O	0	**Flush Receive FIFO** Write 1 to this field to initiate a receive FIFO flush, clearing all data in the receive FIFO. This field is automatically cleared by the hardware when the receive FIFO flush completes. Writing 0 has no effect. 0: Receive FIFO flush complete or not active. 1: Flush the receive FIFO.
6:1	-	RO	0	**Reserved**

I^{2}C Receive Control 0			I2Cn_RXCTRL0	[0x001C]
Bits	**Field**	**Access**	**Reset**	**Description**
0	dnr	R/W	0	**Peripheral Mode Do Not Respond** Peripheral mode operation only. If the device has been addressed for a write operation, and there is still data in the receive FIFO, then: 0: Always respond to an address match with an ACK but always respond to data bytes with a NACK. 1: NACK the address.

Table 14-14: I^{2}C Receive Control 1 Register

I^{2}C Receive Control 1			I2Cn_RXCTRL1	[0x0020]
Bits	**Field**	**Access**	**Reset**	**Description**
31:12	-	RO	0	**Reserved**
11:8	lvl	R	0	**Receive FIFO Byte Count Status** This field returns the number of bytes in the receive FIFO. 0: 0 bytes (No data). 1: 1 byte. 2: 2 bytes. 3: 3 bytes. 4: 4 bytes. 5: 5 bytes. 6: 6 bytes. 7: 7 bytes. 8: 8 bytes.
7:0	cnt	R/W	1	**Receive FIFO Transaction Byte Count Configuration** In controller 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. *This field is ignored when I2Cn_CTRL.irxm_en = 1. To receive more than 256 bytes, use I2Cn_CTRL.irxm_en = 1*

Table 14-15: I^{2}C Transmit Control 0 Register

I^{2}C Transmit Control 0			I2Cn_TXCTRL0	[0x0024]
Bits	**Field**	**Access**	**Reset**	**Description**
31:12	-	RO	0	**Reserved**
11:8	thd_val	R/W	0	**Transmit FIFO Threshold Level** This field sets the level for a transmit FIFO threshold event interrupt. If the number of bytes remaining in the transmit FIFO falls to this level or lower, the interrupt flag *I2Cn_INTFL0.tx_thd* is set, indicating a transmit FIFO threshold event occurred. 0: 0 bytes remaining in the transmit FIFO triggers a transmit FIFO threshold event. 1: 1 byte or fewer remaining in the transmit FIFO triggers a transmit FIFO threshold event (recommended minimum value). … 7: 7 or fewer bytes remaining in the transmit FIFO triggers a transmit FIFO threshold event

I^{2}C Transmit Control 0			I2Cn_TXCTRL0	[0x0024]
Bits	**Field**	**Access**	**Reset**	**Description**
7	flush	R/W1O	0	**Transmit FIFO Flush** A transmit FIFO flush clears all remaining data from the transmit FIFO. 0: Transmit FIFO flush is complete or not active. 1: Flush the transmit FIFO *Note: The hardware automatically clears this bit to 0 after it is written to 1 when the flush is completed.* If *I2Cn_INTFL0.tx_lockout* = 1, then *I2Cn_TXCTRL0.flush* = 1.
6	-	RO	0	**Reserved**
5	nack_flush_dis	R/W	0	**Transmit FIFO received NACK Auto Flush Disable** Various situations or conditions are described in this user guide, leading to the transmit FIFO being flushed and locked out (*I2Cn_INTFL0.tx_lockout* = 1). 0: Received NACK at the end of a peripheral transmit operation enabled. 1: Received NACK at the end of a peripheral transmit operation disabled. *Note: Upon entering transmit preload mode, the hardware automatically sets this bit to 0. The software can subsequently set this bit to any value desired (i.e., the hardware does not continuously force the bit to 0).*
4	rd_addr_flush_dis	R/W	0	**Transmit FIFO Peripheral Address Match Read Auto Flush Disable** Various situations or conditions are described in this user guide, leading to the transmit FIFO being flushed and locked out (*I2Cn_INTFL0.tx_lockout* = 1). 0: Enabled. 1: Disabled. *Note: Upon entering transmit preload mode, hardware automatically sets this bit to 1. The software can subsequently set this bit to any value desired (i.e., the hardware does not continuously force the bitfield to 1).*
3	wr_addr_flush_dis	R/W	0	**Transmit FIFO Peripheral Address Match Write Auto Flush Disable** Various situations or conditions are described in this user guide, leading to the transmit FIFO being flushed and locked out (*I2Cn_INTFL0.tx_lockout* = 1). 0: Enabled. 1: Disabled. *Note: Upon entering transmit preload mode, hardware automatically sets this bit to 1. The software can subsequently set this bit to any value desired (i.e., the hardware does not continuously force the bit to 1).*
2	gc_addr_flush_dis	R/W	0	**Transmit FIFO General Call Address Match Auto Flush Disable** Various situations or conditions are described in this user guide, leading to the transmit FIFO being flushed and locked out (*I2Cn_INTFL0.tx_lockout* = 1). 0: Enabled. 1: Disabled. *Note: Upon entering transmit preload mode, hardware automatically sets this bit to 1. The software can subsequently set this bit to any value desired (i.e., the hardware does not continuously force the bit to 1).*
1	tx_ready_mode	R/W	0	**Transmit FIFO Ready Manual Mode** 0: The hardware controls *I2Cn_TXCTRL1.preload_rdy.* 1: Software control of *I2Cn_TXCTRL1.preload_rdy.*

I^{2}C Transmit Control 0			I2Cn_TXCTRL0	[0x0024]
Bits	**Field**	**Access**	**Reset**	**Description**
0	preload_mode	R/W	0	**Transmit FIFO Preload Mode Enable** 0: Normal operation. An address match in peripheral mode, or a general call address match, flushes and locks the transmit FIFO so it cannot be written and set *I2Cn_INTFL0.tx_lockout.* 1: Transmit FIFO preload mode. An address match in peripheral mode, or a general call address match, does not lock the transmit FIFO and does not set *I2Cn_INTFL0.tx_lockout*. This allows the software to preload data into the transmit FIFO. The status of the I^{2}C is controllable at *I2Cn_TXCTRL1.preload_rdy.*

Table 14-16: I^{2}C Transmit Control 1 Register

I^{2}C Transmit Control Register 1			I2Cn_TXCTRL1	[0x0028]
Bits	**Field**	**Access**	**Reset**	**Description**
31:12	-	RO	0	**Reserved**
11:8	lvl	R	0	**Transmit FIFO Byte Count Status** 0: 0 bytes (No data). 1: 1 byte. 2: 2 bytes. 3: 3 bytes. 4: 4 bytes. 5: 5 bytes. 6: 6 bytes. 7: 7 bytes. 8: 8 bytes (max value).
7:1	-	RO	0	**Reserved**
0	preload_rdy	R/W1O	1	**Transmit FIFO Preload Ready Status** When transmit FIFO preload mode is enabled, *I2Cn_TXCTRL0.preload_mode* = 1, this bit is automatically cleared to 0. While this bit is 0, if the I2Cn hardware receives a peripheral address match, a NACK is sent. Once the I2Cn hardware is ready (the software has preloaded the transmit FIFO, configured the DMA, etc.), the software must set this bit to 1, so the I2Cn hardware sends an ACK on a peripheral address match. When transmit FIFO preload mode is disabled, *I2Cn_TXCTRL0.preload_mode* = 1, this bit is forced to 1, and the I2Cn hardware behaves normally.

Table 14-17: I^{2}C Data Register

I^{2}C Data			I2Cn_FIFO	[0x002C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
7:0	data	R/W	0xFF	**FIFO Data** Reads from this register pop data off the receive FIFO. Writes to this register push data onto the transmit FIFO. Reading from an empty receive FIFO returns 0xFF. Writes to a full transmit FIFO are ignored.

Table 14-18: I^{2}C Controller Control Register

I^{2}C Controller Control			I2Cn_MSTCTRL	[0x0030]
Bits	**Field**	**Access**	**Reset**	**Description**
31:11	-	RO	0	**Reserved**
10:8	-	RO	0	**Reserved**

I^{2}C Controller Control			**I2Cn_MSTCTRL**	**[0x0030]**
Bits	**Field**	**Access**	**Reset**	**Description**
7	ex_addr_en	R/W	0	**Peripheral Extended Addressing Enable** 0: Send a 7-bit address to the peripheral. 1: Send a 10-bit address to the peripheral.
6:3	-	RO	0	**Reserved**
2	stop	R/W1O	0	**Send STOP Condition** 1: Send a STOP Condition at the end of the current transaction. *Note: This bit is automatically cleared by the hardware when the STOP condition begins.*
1	restart	R/W1O	0	**Send Repeated START Condition** After sending data to a peripheral, the controller can send another START to retain control of the bus. 1: Send a repeated START condition to the peripheral instead of sending a STOP condition at the end of the current transaction. *Note: This bit is automatically cleared by the hardware when the repeated START condition begins.*
0	start	R/W1O	0	**Start Controller Mode Transfer** 1: Start controller mode transfer *Note: This bit is automatically cleared by the hardware when the transfer is completed or aborted.*

Table 14-19: I^{2}C SCL Low Control Register

I^{2}C Clock Low Control			**I2Cn_CLKLO**	**[0x0034]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:9	-	RO	0	**Reserved**
8:0	lo	R/W	0x001	**Clock Low Time** In controller mode, this configures the SCL low time. $t_{SCL\_LO} = f_{I2C\_CLK} \times (lo + 1)$ *Note: 0 is not a valid setting for this field.*

Table 14-20: I^{2}C SCL High Control Register

I^{2}C Clock High Control			**I2Cn_CLKHI**	**[0x0038]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:9	-	RO	0	**Reserved**
8:0	hi	R/W	0x001	**Clock High Time** In controller mode, this configures the SCL high time. $t_{SCL\_HI} = {}^{1}/_{f_{I2C\_CLK}} \times (hi + 1)$ In both controller and peripheral mode, this also configures the time SCL is held low after new data is loaded from the transmit FIFO or after the software clears *I2Cn_INTFL0.irxm* during IRXM. *Note: 0 is not a valid setting for this field.*

Table 14-21: I^{2}C Hs-Mode Clock Control Register

I^{2}C Hs-Mode Clock Control			**I2Cn_HSCLK**	**[0x003C]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:16	-	R/W	0	**Reserved**

I^{2}C Hs-Mode Clock Control			I2Cn_HSCLK	[0x003C]
Bits	**Field**	**Access**	**Reset**	**Description**
15:8	hi	R/W	0	**Hs-Mode Clock High Time** This field sets the Hs-Mode clock high count. In peripheral mode, this is the time SCL is held high after data is output on SDA. *Note: See SCL Clock Generation for Hs-Mode for details on the requirements for the Hs-Mode clock high and low times.*
7:0	lo	R/W	0	**Hs-Mode Clock Low Time** This field sets the Hs-Mode clock low count. In peripheral mode, this is the time SCL is held low after data is output on SDA. *Note: See SCL Clock Generation for Hs-Mode for details on the requirements for the Hs-Mode clock high and low times.*

Table 14-22: I^{2}C Timeout Register

I^{2}C Timeout			I2Cn_TIMEOUT	[0x0040]
Bits	**Field**	**Access**	**Reset**	**Description**
31:16	-	RO	0	**Reserved**
15:0	scl_to_val	R/W	0	**Bus Error SCL Timeout Period** Set this value to the number of I2C 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 before the timeout number of I2C clock cycles, a bus error condition is set (*I2Cn_INTFL0.to_err* = 1), and the peripheral releases the SCL and SDA lines 0: Timeout disabled. All other values result in a timeout calculation of: $t_{BUS\_TIMEOUT} = {}^{1}/_{f_{I2C\_CLK}} \times scl\_to\_val$ *Note: The timeout counter monitors the I2Cn peripheral's driving of the SCL pin, not an external I2C device driving the SCL pin.*

Table 14-23: I^{2}C DMA Register

I^{2}C DMA			I2Cn_DMA	[0x0048]
Bits	**Field**	**Access**	**Reset**	**Description**
31:2	-	RO	0	**Reserved**
1	rx_en	R/W	0	**Receive DMA Channel Enable** 0: Disable 1: Enable
0	tx_en	R/W	0	**Transmit DMA Channel Enable** 0: Disable 1: Enable

Table 14-24: I^{2}C Peripheral Address 0 Register

I^{2}C Peripheral Address 0			I2Cn_SLAVE0	[0x004C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:16	-	RO	0	**Reserved**
15	ext_addr_en	R/W	0	**Peripheral Mode Extended Address Length Select** 0: 7-bit addressing 1: 10-bit addressing

I^{2}C Peripheral Address 0			I2Cn_SLAVE0	[0x004C]
Bits	**Field**	**Access**	**Reset**	**Description**
14:10	-	RO	0	**Reserved**
9:0	addr	R/W	0	**Peripheral Mode Peripheral Address** In peripheral mode operation, (*I2Cn_CTRL.mst_mode* = 0), set this field to the peripheral 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. *Note: I2Cn_SLAVE0.ext_addr_en controls if this field is a 7-bit or 10-bit address.*

Table 14-25: I^{2}C Peripheral Address 1 Register

I^{2}C Peripheral Address 1			I2Cn_SLAVE1	[0x0050]
Bits	**Field**	**Access**	**Reset**	**Description**
31:16	-	RO	0	**Reserved**
15	ext_addr_en	R/W	0	**Peripheral Mode Extended Address Length Select** 0: 7-bit addressing. 1: 10-bit addressing.
14:10	-	RO	0	**Reserved**
9:0	addr	R/W	0	**Peripheral Mode Peripheral Address** In peripheral mode operation, (*I2Cn_CTRL.mst_mode* = 0), set this field to the peripheral 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. *Note: I2Cn_SLAVE1.ext_addr_en controls if this field is a 7-bit or 10-bit address.*

Table 14-26: I^{2}C Peripheral Address 2 Register

I^{2}C Peripheral Address 2			I2Cn_SLAVE2	[0x0054]
Bits	**Field**	**Access**	**Reset**	**Description**
31:16	-	RO	0	**Reserved**
15	ext_addr_en	R/W	0	**Peripheral Mode Extended Address Length Select** 0: 7-bit addressing 1: 10-bit addressing
14:10	-	RO	0	**Reserved**
9:0	addr	R/W	0	**Peripheral Mode Peripheral Address** In peripheral mode operation, (*I2Cn_CTRL.mst_mode* = 0), set this field to the peripheral 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. *Note: I2Cn_SLAVE2.ext_addr_en controls if this field is a 7-bit or 10-bit address.*

Table 14-27: I^{2}C Peripheral Address 3 Register

I^{2}C Peripheral Address 3			I2Cn_SLAVE3	[0x0058]
Bits	**Field**	**Access**	**Reset**	**Description**
31:16	-	RO	0	**Reserved**
15	ext_addr_en	R/W	0	**Peripheral Mode Extended Address Length Select** 0: 7-bit addressing 1: 10-bit addressing
14:10	-	RO	0	**Reserved**

I^{2}C Peripheral Address 3			**I2Cn_SLAVE3**	**[0x0058]**
Bits	**Field**	**Access**	**Reset**	**Description**
9:0	addr	R/W	0	**Peripheral Mode Peripheral Address** In peripheral mode operation, (*I2Cn_CTRL.mst_mode* = 0), set this field to the peripheral 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. *Note: I2Cn_SLAVE3.ext_addr_en controls if this field is a 7-bit or 10-bit address.*

15. Inter-Integrated Sound Interface (I^{2}S)

I^{2}S is a serial audio interface for communicating pulse-code modulation (PCM) encoded streams between devices. The peripheral supports both controller and peripheral modes.

Key features:

- Stereo (2 channel) and mono (left or right channel option) formats.
- Separate DMA channels for transmit and receive.
- Flexible timing
  - Configurable sampling rate from $^1/_{65536}$ to 1 of the I^{2}S input clock.
- Flexible data format
  - The number of bits per data word can be selected from 1 to 32, typically 8, 16, 24, or 32-bit width.
  - Feature enhancement not in the I^{2}S specification:
    - Word/Channel select polarity control.
    - First bit position selection.
    - Selectable FIFO data alignment to the MSB or the LSB of the sample.
    - Sample size less than the word size with adjustment to MSB or LSB of the word.
    - Optional sign extension.
- Full-duplex serial communication with separate I^{2}S serial data input and serial data output pins.

15.1 Instances

Table 15-1: MAX78002 I^{2}S Instances

Instance	Supported Channels	I2S_CLK Clock Options		Receive FIFO Depth	Transmit FIFO Depth
I2S	Stereo	I2S_CLKEXT (P0.14)	PCLK	8 × 32-bits	8 × 32-bits

Note: I2S_CLKEXT must be enabled for controller operation; in peripheral operation, external clocking is used for the LRCLK and BCLK input pins.

15.1.1 I^{2}S Bus Lines and Definitions

The I^{2}S peripheral includes support for the following signals:

1. Bit clock line
   - Continuous serial clock (SCK), referred to as bit clock (BCLK) in this document.
2. Word clock line
   - Word select (WS) referred to as left right clock (LRCLK) in this document.
3. Serial data input (SDI)
4. Serial data output (SDO)
5. I2S_CLKEXT input clock is required for operation in controller mode and must be enabled.

Detailed pin mapping is shown in *Table 15-2*. Refer to the device datasheet's pin description table for I^{2}S alternate function mapping.

Table 15-2: MAX78002 I^{2}S Pin Mapping

Instance	I^{2}S Signal	Pin Description	Notes
I2S	BCLK (SCK)	I^{2}S bit clock	Also referred to as serial clock
	LRCLK (WS)	I^{2}S left/right clock	Also referred to as word select
	SDI	I^{2}S serial data input	
	SDO	I^{2}S serial data output	

15.2 Details

The I^{2}S supports full-duplex serial communication with separate SDI and SDO pins. *Figure 15-1* shows an interconnect between a peripheral configured in host mode, communicating with an external I^{2}S peripheral receiver and an external I^{2}S transmitter. In controller mode, the peripheral hardware generates the BCLK and LRCLK, and each is output to each peripheral device.

Note: Controller operation requires the use of the I2S_CLKEXT input to generate the LRCLK and BCLK signals.

Figure 15-1: I^{2}S Controller Mode

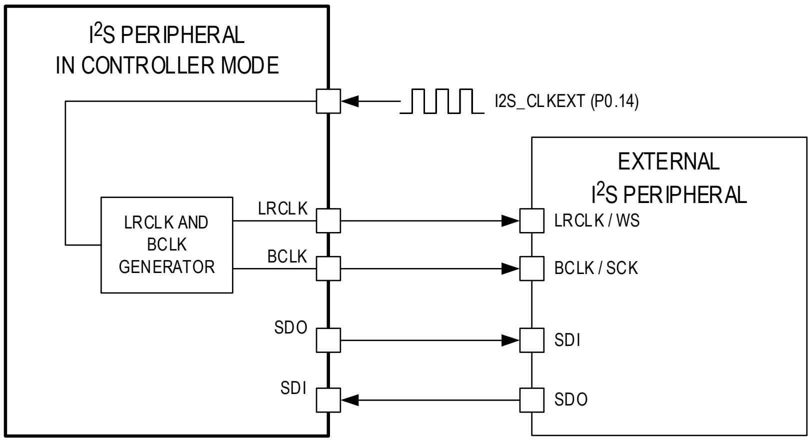
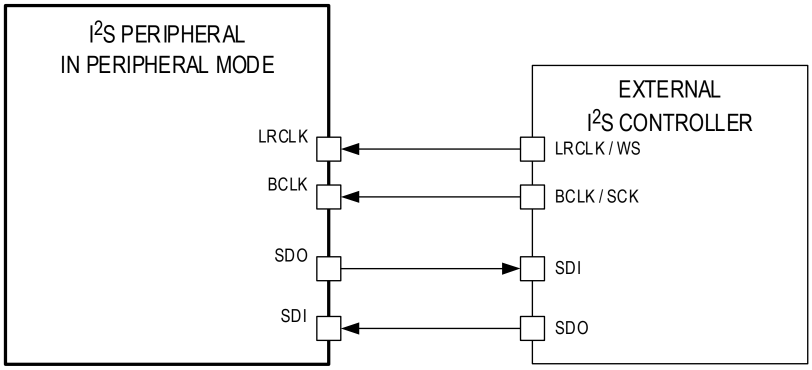


Figure 15-2 shows the I^{2}S peripheral configured for peripheral operation. The LRCLK and BCLK signals are generated externally and are inputs to the I^{2}S peripheral.

Figure 15-2: I^{2}S Peripheral Mode



15.3 Controller and Peripheral Mode Configuration

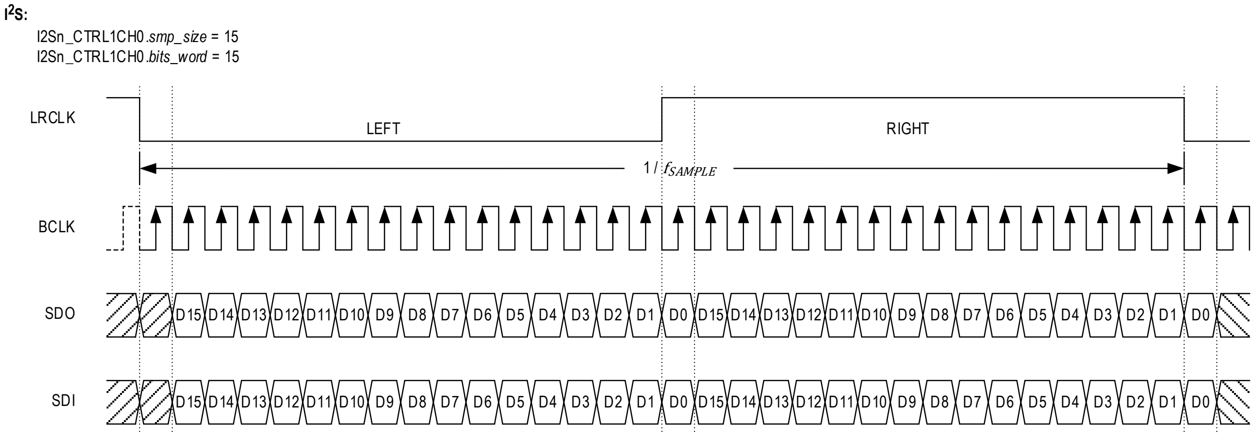
The device supports controller and peripheral modes. In controller mode, the BCLK and LRCLK signals are generated internally and output on the BCLK and LRCLK pins. In peripheral mode, the BCLK and LRCLK pins are configured as inputs, and the external clock source controls the peripheral timing.

Table 15-3: I^{2}S Mode Configuration

Device Mode	***I2S_CTRL0CH0.ch_mode***	LRCLK	BCLK
Controller	0	Output to peripheral	Output to peripheral
Peripheral	3	Input from controller	Input from controller

15.4 Clocking

Figure 15-3: Audio Interface I^{2}S Signal Diagram


I²S:
I2Sn_CTRL1CH0.smp_size = 15
I2Sn_CTRL1CH0.bits_word = 15
LRCLK
LEFT
RIGHT
1 / fSAMPLE
BCLK
SDO
SDI
D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0

I^{2}S communication is synchronized using two signals, the LRCLK and the BCLK. When the I^{2}S peripheral is configured as a controller, the BCLK and LRCLK signals are generated internally by the peripheral using the I2S_CLKEXT input clock. If using the I^{2}S peripheral in controller mode, the I2S_CLKEXT input clock must generate the BCLK and LRCLK signals.

When the I^{2}S peripheral is configured in peripheral mode, the BCLK and LRCLK pins must be configured as inputs. An external controller generates the BCLK and LRCLK signals, which the peripheral uses to synchronize itself to the I^{2}S bus. *Figure 15-3* shows the default I^{2}S signals and timing for I^{2}S communication.

The BCLK frequency is the product of the sample rate, the number of bits per channel (left and right), and the number of channels. For CD audio sampled at a frequency of 44.1kHz, with 16-bit sample width and stereo audio (left and right), the bit clock frequency, f_{BCLK}, is 1.4112MHz as shown in *Equation 15-1*.

Equation 15-1: CD Audio Bit Frequency Calculation

$$f_{BCLK} = 44.1\ kHz \times 16 \times 2 = 1.4112MHz$$

15.4.1 BCLK Generation for Controller Mode

As indicated by *Equation 15-1*, the requirements for determining the BCLK frequency are:

1. Audio sample frequency
2. Number of bits per sample, referred to as sample width

Equation 14-2 shows the formula to calculate the bit clock frequency for a given audio file using the above requirements.

Equation 15-2: Calculating the Bit Clock Frequency for Audio

$$f_{BCLK} = f_{SAMPLE} \times Sample\ Width \times 2$$

In controller mode, the I^{2}S external clock input is used to generate the BCLK frequency. The I^{2}S external clock is divided by the *I2S_CTRL1CH0.clkdiv* field to achieve the target BCLK frequency, as shown in *Equation 15-3*.

Equation 15-3: Controller Mode BCLK Generation Using the I^{2}S External Clock

$$f_{BCLK} = \frac{f_{ERFO}}{(I2Sn\_CTRL1CH0.clkdiv + 1) \times 2}$$

Use *Equation 15-4* to determine the I^{2}S clock divider for a target BCLK frequency.

Equation 15-4: Controller Mode Clock Divisor Calculation for a Target Bit Clock Frequency

$$I2Sn\_CTRL1CH0.clkdiv = \frac{f_{ERFO}}{2 \times f_{BCLK}} - 1$$

15.4.2 LRCLK Period Calculation

An I^{2}S data stream can carry mono (either left or right channel) or stereo (left and right channel) data. The LRCLK signal indicates which channel is currently being sent, either left or right channel data, as shown in *Figure 15-3*. The LRCLK is a 50% duty cycle signal and is the same frequency as the audio sampling frequency, f_{SAMPLE}.

The I^{2}S Peripheral uses the bits per word field, *I2S_CTRL1CH0.bits_word*, to define the audio's sample width, equivalent to the number of bit clocks per channel. This value should be set to the sample width of the audio minus 1. For example, the software should set the *I2S_CTRL1CH0.bits_word* field to 15 for audio sampled using a 16-bit width.

Equation 15-5: Bits Per Word Calculation

$$I2Sn\_CTRL1CH0.bits\_word = Sample\ Width - 1$$

The LRCLK frequency, or word select frequency, is automatically generated by the I^{2}S peripheral hardware is set to operate as a controller. The LRCLK frequency calculation is shown in *Equation 15-6*.

Equation 15-6: LRCLK Frequency Calculation

$$f_{LRCLK} = f_{BCLK} \times (I2Sn\_CTRL1CH0.bits\_word + 1)$$

15.5 Data Formatting

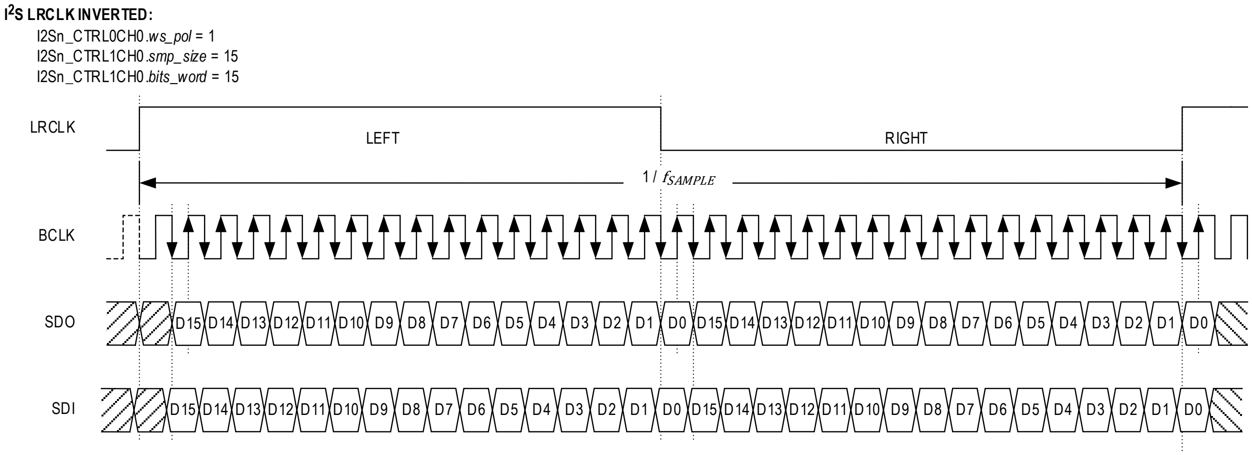
15.5.1 Sample Size

The sample size field, *I2S_CTRL1CH0.smp_size*, defines the number of desired samples within each channel, left, right or mono, for the peripheral. This field can be less than or equal to the *I2S_CTRL1CH0.bits_word* field. For example, for 16-bit sample width audio, the *I2S_CTRL1CH0.bits_word* field must be set to 15. However, the sample size field can be set from 0 to 15. Setting the sample size to 0 is equivalent to setting it to the value of the bits per word field. The sample size field determines how many of the bits per word are transmitted or saved per channel. The sample size field is a 0 based field; therefore, setting *I2S_CTRL1CH0.smp_size* to 15 collects 16 samples. See *Figure 15-6* for an example of the bits per word field's setting compared to the sample size field's setting.

15.5.2 Word Select Polarity

Left channel data, by default, is transferred when the LRCLK signal is low, and right channel data is transferred when the LRCLK signal is high. The polarity of the LRCLK is programmable, allowing left and right data to be swapped. The LRCLK polarity is controlled using the word select polarity field, *I2S_CTRL0CH0.ws_pol*. By default, LRCLK low is for the left channel, high is for the right channel as shown in *Figure 15-3*. Setting *I2S_CTRL0CH0.ws_pol* to 1 inverts the LRCLK polarity, using LRCLK high for the left channel and LRCLK low for the right channel as shown in *Figure 15-4*.

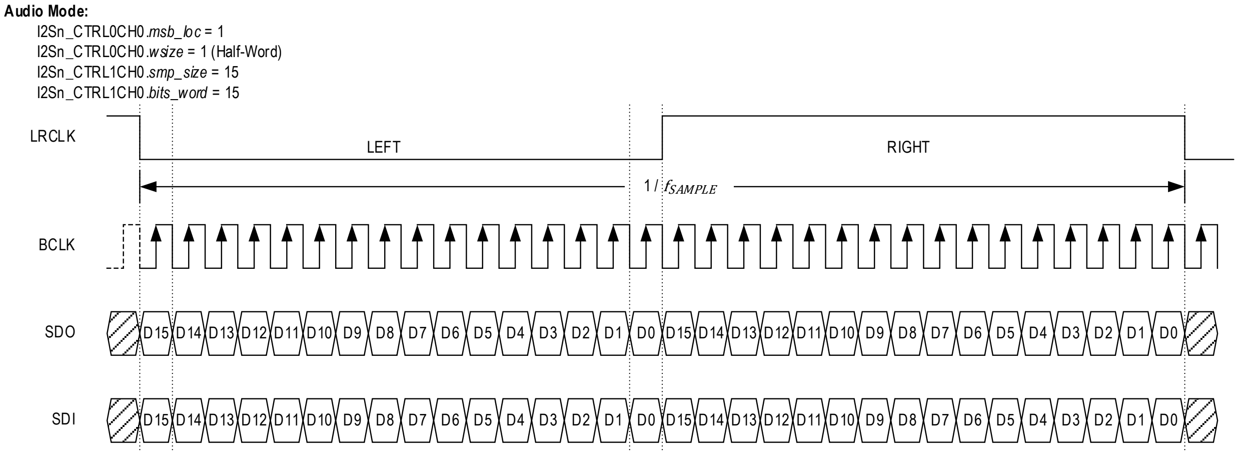
Figure 15-4: Audio Mode with Inverted Word Select Polarity


I2S LRCLK INVERTED:
I2Sn_CTRL0CH0.ws_pol = 1
I2Sn_CTRL1CH0.smp_size = 15
I2Sn_CTRL1CH0.bits_word = 15
LRCLK
LEFT
RIGHT
1 / fSAMPLE
BCLK
SDO
SDI
D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0

15.5.3 First Bit Location Control

The default setting is for the first bit of I^{2}S data to be located at the second complete BCLK cycle after the LRCLK transition required by the I^{2}S specification. See *Figure 15-3* for the standard data sampling configuration. Optionally, the first bit location can be left justified, resulting in the first bit of data being sampled on the first BCLK cycle after the LRCLK signal transitions as shown in *Figure 15-5*. Set *I2S_CTRL0CH0.msb_loc* to 1 to left justify the data with respect to the LRCLK.

Figure 15-5: Audio Controller Mode Left-Justified First Bit Location


Audio Mode:
I2Sn_CTRL0CH0.msb_loc = 1
I2Sn_CTRL0CH0.wsize = 1 (Half-Word)
I2Sn_CTRL1CH0.smp_size = 15
I2Sn_CTRL1CH0.bits_word = 15
LRCLK
LEFT
RIGHT
1 / fSAMPLE
BCLK
SDO
SDI
D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0 D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0

15.5.4 Sample Adjustment

When the sample size field, *I2S_CTRL1CH0.smp_size*, is less than the bits per word field, *I2S_CTRL1CH0.bits_word*, use the *I2S_CTRL1CH0.adjust* field to set which bits are stored in the receive FIFO or transmitted from the transmit FIFO, either from the first sample of the SDI/SDO line or the last sample of the SDI/SDO line for the left and right channels. *Figure 15-6* shows an example of the default adjustment, MSB, where *I2S_CTRL1CH0.smp_size* = 7 and *I2S_CTRL1CH0.bits_word* = 15. *Figure 15-7* shows the adjustment set to the LSB of the SDI/SDO data.

Figure 15-6: MSB Adjustment when Sample Size is Less Than Bits Per Word

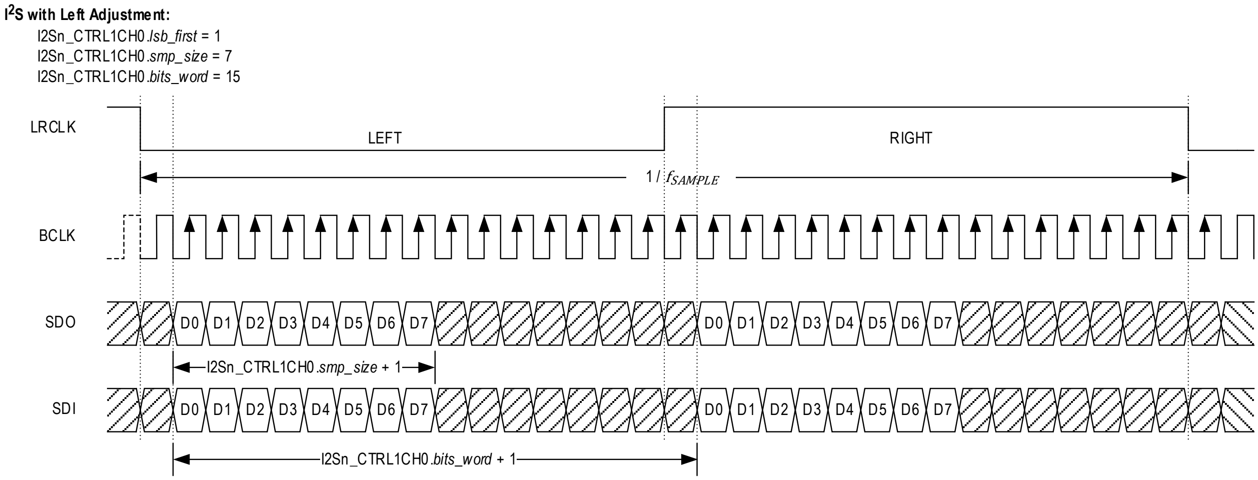
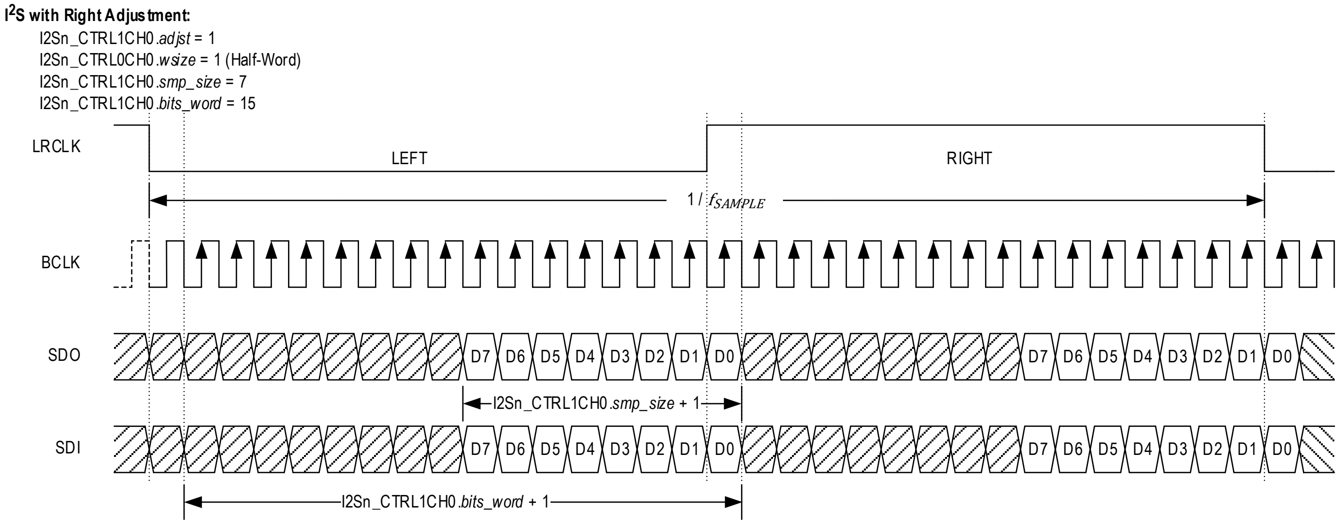

I²S with Left Adjustment:
I2Sn_CTRL1CH0.lsb_first = 1
I2Sn_CTRL1CH0.smp_size = 7
I2Sn_CTRL1CH0.bits_word = 15
LRCLK
LEFT
RIGHT
1 / fSAMPLE
BCLK
SDO
D0 D1 D2 D3 D4 D5 D6 D7
D0 D1 D2 D3 D4 D5 D6 D7
I2Sn_CTRL1CH0.smp_size + 1
SDI
D0 D1 D2 D3 D4 D5 D6 D7
D0 D1 D2 D3 D4 D5 D6 D7
I2Sn_CTRL1CH0.bits_word + 1

Figure 15-7: LSB Adjustment when Sample Size is Less Than Bits Per Word


I²S with Right Adjustment:
I2Sn_CTRL1CH0.adjst = 1
I2Sn_CTRL0CH0.wsize = 1 (Half-Word)
I2Sn_CTRL1CH0.smp_size = 7
I2Sn_CTRL1CH0.bits_word = 15
LRCLK
LEFT
RIGHT
1 / fSAMPLE
BCLK
SDO
D7 D6 D5 D4 D3 D2 D1 D0
D7 D6 D5 D4 D3 D2 D1 D0
I2Sn_CTRL1CH0.smp_size + 1
SDI
D7 D6 D5 D4 D3 D2 D1 D0
D7 D6 D5 D4 D3 D2 D1 D0
I2Sn_CTRL1CH0.bits_word + 1

15.5.5 Stereo/Mono Configuration

The I^{2}S can transfer stereo or mono data based on the *I2S_CTRL0CH0.stereo* field. In stereo mode, both the left and right channels hold data. In mono mode, only the left or right channel contain data. For stereo mode, set *I2S_CTRL0CH0.stereo* to 0. Set the *I2S_CTRL0CH0.stereo* field to 2 for left channel mono. Set the *I2S_CTRL0CH0.stereo* field to 3 for right channel mono.

Figure 15-8: I²S Mono Left Mode

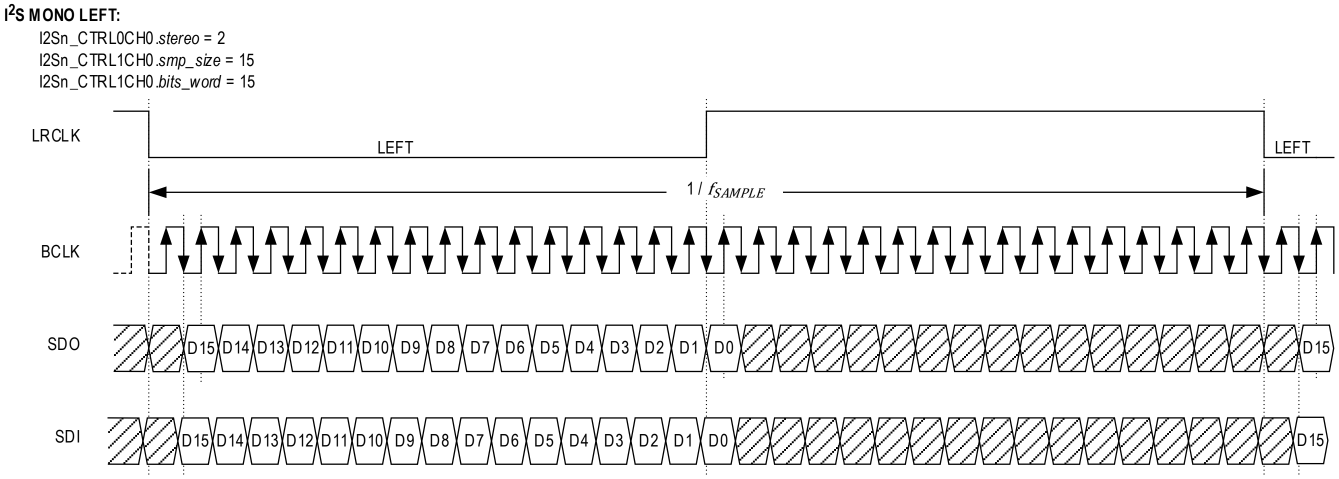
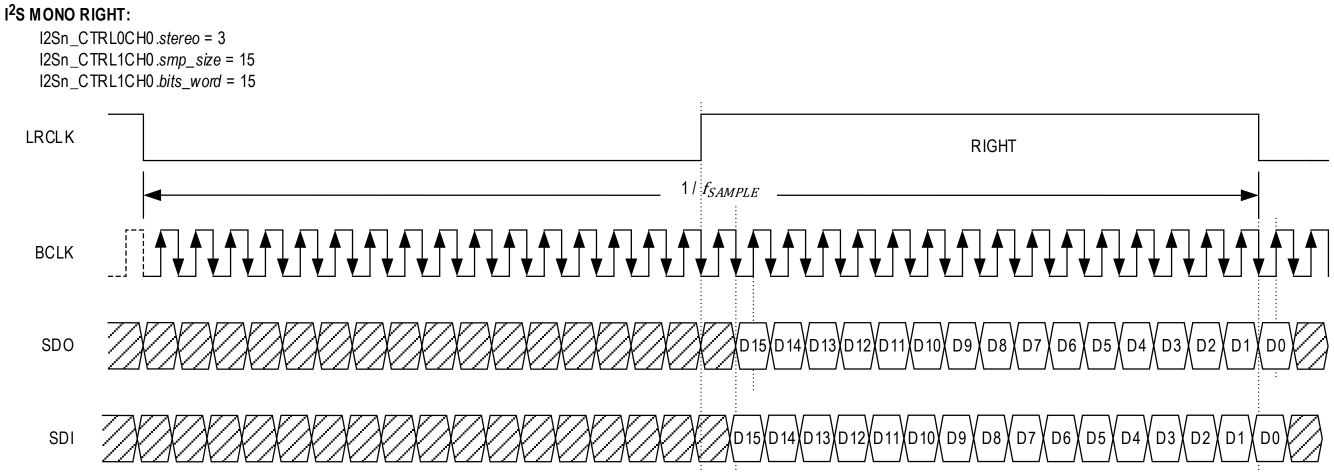

I²S MONO LEFT:
I2Sn_CTRL0CH0.stereo = 2
I2Sn_CTRL1CH0.smp_size = 15
I2Sn_CTRL1CH0.bits_word = 15
LRCLK
LEFT
LEFT
1 / fSAMPLE
BCLK
SDO
D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
D15
SDI
D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
D15

Figure 15-9: I²S Mono Right Mode


I²S MONO RIGHT:
I2Sn_CTRL0CH0.stereo = 3
I2Sn_CTRL1CH0.smp_size = 15
I2Sn_CTRL1CH0.bits_word = 15
LRCLK
RIGHT
1 / fSAMPLE
BCLK
SDO
D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
SDI
D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0

15.6 Transmit and Receive FIFOs

15.6.1 FIFO Data Width

I²S audio data is programmable from 1 to 32 bits using the *I2S_CTRL1CH0.bits_word* field. The software can set the FIFO width to either 8-bits (byte), 16-bits (half-word), or 32-bits (word). Set the FIFO width using the *I2S_CTRL0CH0.wsize* field. For FIFO word sizes less than 32-bits, the data frame, comprising a complete LRCLK cycle, can still be 64 bits; the unused bits are transmitted as zero by the hardware.

15.6.2 Transmit FIFO

An I²S transaction is started by writing data to the transmit FIFO using the *I2S_FIFOCH0.data* register, either directly or using a DMA channel. The data written is automatically transmitted out by the hardware, a FIFO word, as defined using the I2S_CTRL0CH0.*wsize* field, at a time, in the order it was written to the transmit FIFO. Use the I²S interrupt flags to monitor the transmit FIFO status and determine when the transfer cycle(s) have been completed.

If the transmit FIFO becomes empty, an error condition occurs and results in undefined behavior.

15.6.3 Receive FIFO

The received data is loaded into the receive FIFO, and it can then be unloaded by reading from the *I2S_FIFOCH0.data* register. An overrun event occurs if the receive FIFO is full and another word is shifted into the FIFO.

15.6.4 FIFO Word Control

The data width of the transmit and receive FIFOs can be configured using the *I2S_CTRL0CH0.wsize* field. The following tables describe the data ordering based on the *I2S_CTRL0CH0.wsize* setting.

The transmit and receive FIFOs must be flushed, and the peripheral reset by the software before reconfiguration. The software resets the peripheral by setting the *I2S_CTRL0CH0.rst* field to 1.

Table 15-4: Data Ordering for Byte Data Size (Stereo Mode)

Byte Data Width (*I2S_CTRL0CH0*.*wsize* = 0)				
FIFO Entry	**MS Byte**			**LS Byte**
FIFO 0	Right Channel Byte 1	Left Channel Byte 1	Right Channel Byte 0	Left Channel Byte 0
FIFO 1	Right Channel Byte 3	Left Channel Byte 3	Right Channel Byte 2	Left Channel Byte 2
…	…	…	…	…
FIFO 7	Right Channel Byte 14	Left Channel Byte 14	Right Channel Byte 13	Left Channel Byte 13

Table 15-5: Data Ordering for Half-Word Data Size (Stereo Mode)

Half-Word Data Width (*I2S_CTRL0CH0*.*wsize* = 1)		
FIFO Entry	**MS Half-Word**	**LS Half-Word**
FIFO 0	Right Channel Half-Word 0	Left Channel Half-Word 0
FIFO 1	Right Channel Half-Word 1	Left Channel Half-Word 1
…	…	…
FIFO 7	Right Channel Half Word 7	Left Channel Half-Word 7

Table 15-6: Data Ordering for Word Data Size (Stereo Mode)

Word Data Width (*I2S_CTRL0CH0*.*wsize* = 2 or 3)	
FIFO Entry	**Word**
FIFO 0	Left Channel Word 0
FIFO 1	Right Channel Word 0
FIFO 2	Left Channel Word 1
FIFO 3	Right Channel Word 1
…	…
FIFO 6	Left Channel Word 3
FIFO 7	Right Channel Word 3

15.6.5 FIFO Data Alignment

The I^{2}S data can be left aligned or right aligned using the *I2S_CTRL0CH0.align* field. The following conditions apply to each setting:
Left aligned: *I2S_CTRL0CH0.align* = 0

- If the number of bits per word is greater than the FIFO data width:
  - Receive: All bits after the LSB of the FIFO data width is discarded.
  - Transmit: All bits after the LSB of the FIFO data width are sent as 0.
- If the number of bits per word is less than the FIFO data width:
  - Receive: The data received is stored starting at the MSB of the FIFO entry up to the number of bits per word plus one bit.
  - Transmit: The transmit FIFO data is sent from the LSB to the number of bits plus 1.

Right aligned: *I2S_CTRL0CH0.align* = 1

- If the number of bits per word is greater than the FIFO data width:
  - Receive: The data received is stored in the receive FIFO starting with the LSB up to the FIFO data width, and any additional bits are discarded.
  - Transmit: 0 bits are transmitted for all bits greater than the FIFO data width. For example, if the bits per word field is set to 12 and the FIFO data width is 8, the first 4 bits are transmitted as 0, the 8-bits of data in the FIFO are transmitted.
- If the number of bits per word is less than the FIFO data width:
  - Receive: The data received is sign extended and saved to the receive FIFO.
  - Transmit: The transmit FIFO data is sent from the LSB to the number of bits plus 1.

15.6.6 Typical Audio Configurations

Table 15-7 shows the relationship between the bits per word field and the sample size field. *Equation 15-7* shows the required relationship between the sample size field and the bits per word field.

Equation 15-7: Sample Size Relationship Bits per Word

$$I2Sn\_CTRL1CH0.smp\_size \leq I2Sn\_CTRL1CH0.bits\_word$$

The *I2S_CTRL1CH0.bits_word* column in *Table 15-7* is set using the equation $\frac{\# BCLK}{Channel} - 1$. The *I2S_CTRL1CH0.smp_size* column is the number of samples per word captured from the I^{2}S bus and is calculated by the equation $\frac{\# Samples}{Channel} - 1$. Channel refers to the left and right channels of audio.

Table 15-7: Configuration for Typical Audio Width and Samples per WS Clock Cycle

Audio Sample Width/ Samples per WS Cycle	$\frac{\# BCLK}{Channel}$	$\frac{\# Samples}{Channel}$	*I2S_CTRL1CH0*			Sign extension (align = 1)[†]
			bits_word	smp_size	wsize	
8-bit / 16	8	8	7	7	0	
16-bit / 32	16	16	15	15	1	
20-bit / 40	20	20	19	19	2	sign
24-bit / 48	24	24	23	23	2	sign
24-bit / 64	32	24	31	23	2	sign
32-bit / 64	32	32	31	31	2	

Audio Sample Width/ Samples per WS Cycle	$\frac{\# BCLK}{Channel}$	$\frac{\# Samples}{Channel}$	I2S_CTRL1CH0			Sign extension (align = 1)†
			bits_word	smp_size	wsize	

† Sign Extension only applies when I2S_CTRL0CH0.align is set to 1 and I2S_CTRL1CH0.smp_size is less than the FIFO width size setting.

15.7 Interrupt Events

The I^{2}S peripheral generates interrupts for the events shown in *Table 15-8*. An interrupt is generated if the corresponding interrupt enable field is set. The interrupt flags stay set until cleared by the software by writing 1 to the interrupt flag field.

Table 15-8. I^{2}S Interrupt Events

Event	Interrupt Flag	Interrupt Enable
Receive FIFO overrun	*I2S_INTFL.rx_ov_ch0*	*I2S_INTEN.rx_ov_ch0*
Receive threshold	*I2S_INTFL.rx_thd_ch0*	*I2S_INTEN.rx_thd_ch0*
Transmit FIFO half-empty	*I2S_INTFL.tx_he_ch0*	*I2S_INTEN.tx_he_ch0*
Transmit FIFO one byte remaining	*I2S_INTFL.tx_ob_ch0*	*I2S_INTEN.tx_ob_ch0*

15.7.1 Receive FIFO Overrun

A receive FIFO overrun event occurs if the number of data words in the receive FIFO, *I2S_DMACH0.rx_lvl* is equal to the RX_FIFO_DEPTH, and another word has been shifted into the FIFO. The hardware automatically sets the *I2S_INTFL.rx_ov_ch0* field to 1 when this event occurs.

15.7.2 Receive FIFO Threshold

A receive FIFO threshold event occurs when a word is shifted in and the number of words in the receive FIFO, *I2S_DMACH0.rx_lvl*, exceeds the *I2S_CTRL0CH0.rx_thd_val*. The event does not occur if the opposite transition occurs. When this event occurs, hardware automatically sets the *I2S_INTFL.rx_thd_ch0* field to 1.

15.7.3 Transmit FIFO Half-Empty

A transmit FIFO half-empty event occurs when the number of words in the transmit FIFO, *I2S_DMACH0.tx_lvl*, is less than ½ of the TX_FIFO_DEPTH as shown in *Equation 15-8*. When this event occurs, the *I2S_INTFL.tx_he_ch0* flag is set to 1 by hardware.

Note: The transmit FIFO half empty interrupt flag is set by the hardware one BCLK cycle before the actual condition occurring. If the BCLK is much slower than the I^{2}S peripheral clock, the software can receive the interrupt while the actual transmit FIFO level is still equal to ½ of the TX_FIFO_DEPTH. The software should always read the transmit FIFO level before filling it to determine the correct number of words to write to the transmit FIFO. Read the level of the transmit FIFO using the I2S_DMACH0.tx_lvl field.

Equation 15-8: Transmit FIFO Half-Empty Condition

$$I2Sn\_DMACH0.tx\_lvl < \left(\frac{TX\ FIFO\ DEPTH}{2}\right)$$

15.7.4 Transmit FIFO One Entry Remaining

A transmit FIFO one entry remaining event occurs when the number of entries in the transmit FIFO is 1, *I2S_DMACH0.tx_lvl* = 1. When this event occurs, the *I2S_INTFL.tx_ob_ch0* flag is set to 1 by the hardware.

Note: The transmit FIFO one entry remaining interrupt flag is set by the hardware one BCLK cycle before the actual condition occurring. If the BCLK is much slower than the I^{2}S peripheral clock, the software can receive the interrupt while the actual transmit FIFO level is still equal to 2. The software should always read the transmit FIFO level before filling it to determine the correct number of words to write to the transmit FIFO. Read the level of the transmit FIFO using the I2S_DMACH0.tx_lvl field.

15.8 Direct Memory Access

The I^{2}S supports DMA for both transmit and receive; separate DMA channels can be connected to the receive and transmit FIFOs. The following describes the behavior of the receive and transmit DMA requests.

- A receive DMA request is asserted when the number of words in the receive FIFO is greater than or equal to the receive FIFO threshold.
- A transmit DMA request is asserted when the number of valid bytes in the transmit FIFO is less than ½ of the transmit FIFO's depth.

15.9 Block Operation

After exiting a power-on reset, the IP is disabled by default. It must be enabled and configured by the software to establish the I^{2}S serial communication. A typical software sequence is shown below.

1. Set *GCR_PCLKDIS1.i2s* to 0 to enable the I^{2}S peripheral clock source shown in *Table 15-1*.
2. Disable the I^{2}S clock by setting *I2S_CTRL1CH0.en* to 0.
3. Set *I2S_CTRL0CH0.rst* to 1 to reset the I^{2}S configuration.
4. Set *I2S_CTRL0CH0.flush* to 1 to flush the FIFO buffers.
5. Configure the *I2S_CTRL0CH0.ch_mode* to select the controller or peripheral configuration.
   a. For controller mode, configure the baud rate by programming the *I2S_CTRL1CH0.clkdiv* field to achieve the required bit rate, set the *I2S_CTRL1CH0.smp_size* field to the desired sample size of the data, and the *I2S_CTRL1CH0.adjust* field if the Sample Size is smaller than the number of bits per word.
6. Configure the threshold of the receive FIFO by programming the *I2S_CTRL0CH0.rx_thd_val*. The transmit FIFO threshold is a fixed value, which is half of the transmit FIFO depth.
7. If desired, configure DMA operation. See section *Direct Memory Access* for details.
8. Enable interrupt functionality by configuring the *I2S_INTEN* register if desired.
9. Program the *clkdiv* bits in the *I2S_CTRL1CH0* register for the new bit clock frequency.
10. For controller operation, load data in the transmit FIFO for transmit.
11. Re-enable the bit clock by setting *I2S_CTRL1CH0.en* to 1.

15.10 Registers

See *Table 3-3* 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 15-9*. Register names for a specific instance are defined by replacing "n" with the instance number. For example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See *Table 1-1* 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 15-9: I^{2}S Register Summary

Offset	Register Name	Description
[0x0000]	*I2S_CTRL0CH0*	I^{2}S Global Mode Control 0 Register

Offset	Register Name	Description
[0x0010]	*I2S_CTRL1CH0*	I^{2}S Controller Mode Configuration Register
[0x0030]	*I2S_DMACH0*	I^{2}S DMA Control Channel Register
[0x0040]	*I2S_FIFOCH0*	I^{2}S FIFO Register
[0x0050]	*I2S_INTFL*	I^{2}S Interrupt Status Register
[0x0054]	*I2S_INTEN*	I^{2}S Interrupt Enable Register

15.10.1 Register Details

Table 15-10: I^{2}S Control 0 Register

I^{2}S Control 0 Register			I2S_CTRL0CH0	[0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**
31:24	rx_thd_val	R/W	0	**Receive FIFO Interrupt Threshold** This field specifies the level of the receive FIFO for the threshold interrupt generation. Values of 0 or greater than the RX_FIFO_DEPTH are ignored.
23:21	-	RO	0	**Reserved**
20	fifo_lsb	R/W	0	**FIFO Bit Field Control** Only used if the FIFO size is larger than the sample size and *I2S_CTRL0CH0.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	R/W1O	0	**Reset** Write 1 to reset the I^{2}S peripheral. The hardware automatically clears this field to 0 when the reset is complete. 0: Reset not in process. 1: Reset peripheral.
18	flush	R/W1O	0	**FIFO Flush** Write 1 to start a flush of the receive FIFO and the transmit FIFO. The hardware automatically clears this field when the operation is complete. 0: Flush complete or not in process. 1: Flush receive and transmit FIFOs.
17	rx_en	R/W	0	**Receive Enable** Enable receive mode for the I^{2}S peripheral. 0: Disabled 1: Enabled
16	tx_en	R/W	0	**Transmit Enable** Enable transmit mode for the I^{2}S peripheral. 0: Disabled 1: Enabled
15:14	wsize	R/W	3	**Data Size When Reading/Writing FIFO** Set this field to the desired width for data writes and reads from the FIFO. 0: Byte 1: Half-word (16 bits) 2-3: Word (32 bits)

I^{2}S Control 0 Register		I2S_CTRL0CH0		[0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**
13:12	stereo	R/W	0	**I^{2}S Mode** Select the mode for the I^{2}S to stereo, mono left channel only, or mono right channel only. 0-1: Stereo 2: Mono left channel 3: Mono right channel
11	-	RO	0	**Reserved**
10	align	R/W	0	**FIFO Data Alignment** Set this field to control the alignment of the data in the FIFOs. This field is only used if the FIFO data width, *I2S_CTRL0CH0.wsize*, is not equal to the bits per word field. 0: MSB 1: LSB
9	msb_loc	R/W	0	**First Bit Location Sampling** This field controls when the first bit is transmitted/received in relation to the LRCLK. The first bit is transmitted/received on SDO/SDI on the second complete LRCLK cycle by default. Set this field to 1 to transmit/receive the first bit of data on the first complete LRCLK cycle. 0: Second complete LRCLK cycle is the first bit of the data 1: First complete LRCLK cycle is the first bit of the data
8	ws_pol	R/W	0	**LRCLK Polarity Select** This field determines the polarity of the LRCLK signal associated with the left channel data. Set this field to 1 to associate the left channel with the LRCLK high state. The default setting is the standard I^{2}S association. 0: LRCLK low for the left channel 1: LRCLK high for the left channel
7:6	ch_mode	R/W	0	**Mode** Set this field to indicate controller or peripheral I^{2}S operation. When using controller mode, the I2S_CLKEXT input clock must be used to generate the LRCLK/BCLK signals. 0: Controller mode, internal generation of LRCLK/BCLK using the I2S_CLKEXT input clock. 1 - 2: Reserved 3: Peripheral mode, external generation of LRCLK/BCLK
5:2	-	DNM	0	**Reserved, Do Not Modify**
1	lsb_first	R/W	0	**LSB First** Setting this field to 1 indicates the least significant bit of the data is transmitted/received first on the SDI/SDO pins. The default setting, 0, indicates the most significant bit of the data is received first. 0: Disabled 1: Enabled
0	-	RO	0	**Reserved**

Table 15-11: I^{2}S Controller Mode Configuration Register

I^{2}S Controller Mode Configuration		I2S_CTRL1CH0		[0x0010]
Bits	**Field**	**Access**	**Reset**	**Description**
31:16	clkdiv	R/W	0	**I^{2}S Frequency Divisor** Set this field to the required divisor to achieve the desired frequency for the I^{2}S BCLK. See *BCLK Generation for Controller Mode* for detailed information. *Note: This field only applies when the I^{2}S peripheral is set to controller mode, I2S_CTRL0CH0.ch_mode = 0.*

I^{2}S Controller Mode Configuration			**I2S_CTRL1CH0**	**[0x0010]**
Bits	**Field**	**Access**	**Reset**	**Description**
15	adjust	R/W	0	**Data Justification When Sample Size is Less than Bits Per Word** This field is used to determine which bits are used if the sample size is less than the bits per word. 0: Left adjustment 1: Right adjustment
14	-	RO	0	**Reserved**
13:9	smp_size	R/W	0	**Sample Size** This field is the desired sample size of the data received or transmitted with respect to the Bits per Word field. In most use cases, the sample size is equal to the bits per word. However, in some situations, fewer bits are required by the application, which allows flexibility. An example use case would be for 16-bit audio being received, and the application only needs 8-bits of resolution. See *Sample Size* for additional details. *Note: The sample size is equal to I2S_CTRL1CH0.bits_word when I2S_CTRL1CH0.smp_size = 0 or I2S_CTRL1CH0.smp_size > I2S_CTRL1CH0.bits_word.*
8	en	R/W	0	**I^{2}S Enable** For controller mode operation, this field is used to start generating the I^{2}S LRCLK and BCLK outputs. In peripheral mode, this field enables the peripheral to begin receiving signals on the I^{2}S interface. 0: Disabled. 1: Enabled
7:5	-	RO	0	**Reserved**
4:0	bits_word	R/W	0	**I^{2}S Word Length** This field is defined as the I^{2}S data bits per left and right channel. *Example: If the bit clocks is 16 per half frame, bits_word is 15.*

Table 15-12: I^{2}S DMA Control Register

I^{2}S DMA Control			**I2S_DMACH0**	**[0x0030]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:24	rx_lvl	RO	0	**Receive FIFO Level** This field is the number of data words in the receive FIFO.
23:16	tx_lvl	RO	0	**Transmit FIFO Level** This field is the number of data words in the transmit FIFO.
15	dma_rx_en	R/W	0	**DMA Receive Channel Enable** 0: Disabled 1: Enabled
14:8	dma_rx_thd_val	R/W	0	**DMA Receive FIFO Event Threshold** If the receive FIFO level is greater than this value, then the receive FIFO DMA interface sends a signal to the system DMA indicating the receive FIFO has characters to transfer to memory.
7	dma_tx_en	R/W	0	**DMA Transmit Channel Enable** 0: Disabled 1: Enabled
6:0	dma_tx_thd_val	RO	0	**DMA Transmit FIFO Event Threshold** If the transmit FIFO level is less than this value, then the transmit FIFO DMA interface sends a signal to system DMA, indicating the transmit FIFO is ready to receive data from memory.

Table 15-13: I²S FIFO Register

I²S FIFO Register				**I2S_FIFOCH0**	**[0x0040]**
Bits	**Field**	**Access**	**Reset**	**Description**	
31:0	data	R/W	0	**I²S FIFO** Writing to this field loads the next character into the transmit FIFO and increments the *I2S_DMACH0.tx_lvl*. Writes are ignored if the transmit FIFO is full. Reads of this field return the next character available from the receive FIFO and decrement the *I2S_DMACH0.rx_lvl*. The value 0 is returned if *I2S_DMACH0.rx_lvl* = 0.	

Table 15-14: I²S Interrupt Flag Register

I²S Interrupt Flag				**I2S_INTFL**	**[0x0050]**
Bits	**Field**	**Access**	**Reset**	**Description**	
31:4	-	DNM	0	**Reserved, Do Not Modify**	
3	tx_he_ch0	W1C	0	**Transmit FIFO Half-Empty Event Interrupt Flag** 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	**Transmit FIFO One Entry Remaining Event Interrupt Flag** 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	**Receive FIFO Threshold Event Interrupt Flag** 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	**Receive FIFO Overrun Event Interrupt Flag** If this field is set to 1, the event has occurred. Write 1 to clear. 0: No event 1: Event occurred	

Table 15-15: I²S Interrupt Enable Register

I²S Interrupt Enable				**I2S_INTEN**	**[0x0054]**
Bits	**Field**	**Access**	**Reset**	**Description**	
31:4	-	DNM	0	**Reserved, Do Not Modify**	
3	tx_he_ch0	R/W	0	**Transmit FIFO Half-Empty Event Interrupt Enable** Set this field to 1 to enable interrupts for this event. 0: Disabled 1: Enabled	
2	tx_ob_ch0	R/W	0	**Transmit FIFO One Entry Remaining Event Interrupt Enable** Set this field to 1 to enable interrupts for this event. 0: Disabled 1: Enabled	
1	rx_thd_ch0	R/W	0	**Receive FIFO Threshold Event Interrupt Enable** Set this field to 1 to enable interrupts for this event. 0: Disabled 1: Enabled	
0	rx_ov_ch0	R/W	0	**Receive FIFO Overrun Event Interrupt Enable** Set this field to 1 to enable interrupts for this event. 0: Disabled 1: Enabled	

16. Camera Interface (CAMERAIF)

The CAMERAIF 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

16.1 Instances

There is one instance of the CAMERAIF shown in *Table 16-1*. The alternate function names for the CAMERAIF are shown in *Table 16-2*. Refer to the device data sheet's pin description table for alternate function mapping to device pins.

Table 16-1: MAX78002 CAMERAIF Instances

Instance	CAMERAIF Clock Options	Receive FIFO Depth
CAMERAIF	PCLK	8

Table 16-2: MAX78002 CAMERAIF Signals

Signal Name	Signal Direction	Description
PCIF_PCLK	Input	Pixel Clock Input
PCIF_HSYNC	Input	Horizontal Synchronization Input
PCIF_VSYNC	Input	Vertical Synchronization Input
PCIF_D0	Input	Pixel Data Input 0
PCIF_D1	Input	Pixel Data Input 1
PCIF_D2	Input	Pixel Data Input 2
PCIF_D3	Input	Pixel Data Input 3
PCIF_D4	Input	Pixel Data Input 4
PCIF_D5	Input	Pixel Data Input 5
PCIF_D6	Input	Pixel Data Input 6
PCIF_D7	Input	Pixel Data Input 7
PCIF_D8	Input	Pixel Data Input 8
PCIF_D9	Input	Pixel Data Input 9
PCIF_D10	Input	Pixel Data Input 10
PCIF_D11	Input	Pixel Data Input 11

16.2 Capture Modes

The CAMERAIF supports either single image capture mode or continuous capture mode. Each mode and the CAMERAIF configuration are described in the following sections.

16.2.1 Single Image Capture

In this mode, the CAMERAIF waits for one image from the sensor, then stops reading data. Configure the CAMERAIF for this mode by setting the *CAMERAIF_CTRL.read_mode* field to 1. The *CAMERAIF_CTRL.read_mode* field remains set to 1 before and while receiving image data from the camera. Once the image is complete, the hardware automatically sets the *CAMERAIF_CTRL.read_mode* field to 0 and sets the *CAMERAIF_INT_FL.img_done* status to 1.

16.2.2 Continuous Capture

In this mode, the CAMERAIF continues to read image data as long as the connected camera sensor continues to provide image data. Configure the CAMERAIF for continuous capture mode by setting the *CAMERAIF_CTRL.read_mode* field to 2. Disable continuous mode capture by setting the *CAMERAIF_CTRL.read_mode* field to 0.

16.3 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.

16.3.1 Horizontal and Vertical Synchronization Timing Mode

In this timing mode, the CAMERAIF uses the PCIF_HSYNC and the PCIF_VSYNC input pins to determine the beginning and end of image data. The CAMERAIF 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 to detect the start of the 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 CAMERAIF that the image is complete.

Set the bit *CAMERAIF_CTRL.ds_timing_en* to 0 to configure the CAMERAIF for horizontal and vertical synchronization mode.

16.3.2 Data Stream Timing Mode

In this timing mode, the PCIF_HSYNC and PCIF_VSYNC input pins are ignored. The CAMERAIF 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 SAV code (*CAMERAIF_DS_TIMING_CODES.sav*) and the EAV code (*CAMERAIF_DS_TIMING_CODES.eav*). These two codes must match the codes sent by the connected camera respectively and cannot be identical. Set *CAMERAIF_CTRL.ds_timing_en* to 1 to configure the CAMERAIF for embedded timing codes mode.

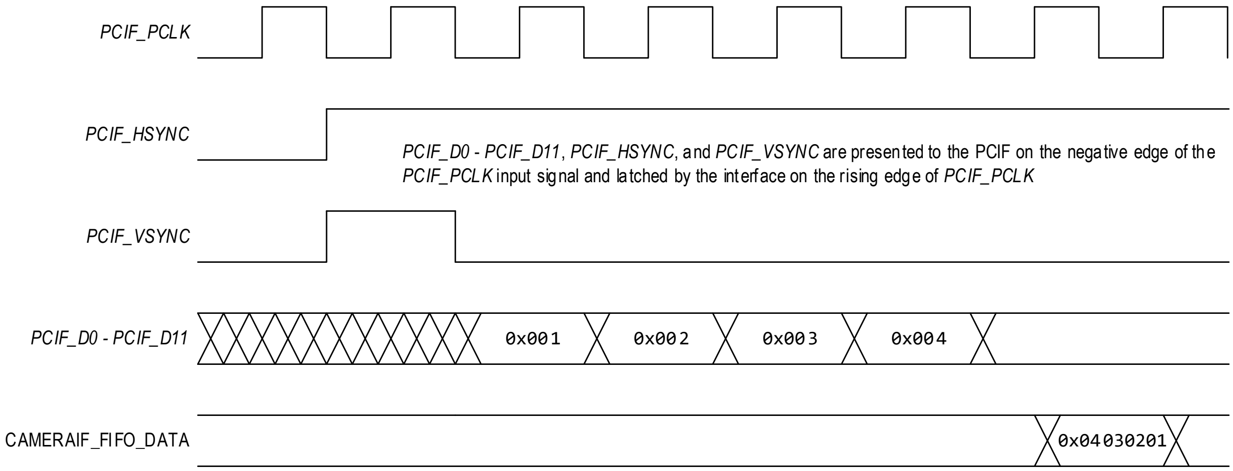
16.4 Data Width

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 changes on the negative edge of PCIF_PCLK.

16.4.1 8-Bit Width

Setting *CAMERAIF_CTRL.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 is framed as 32-bit words before these words are transferred to the 32-bit wide data FIFO and made ready to be read. The 32-bit data FIFO word is oriented with the most significant byte most recently received 8-bit PCIF_Dx data. See *Figure 16-1* and *Figure 16-2* examples.

Figure 16-1: Horizontal and Vertical Synchronization Timing Mode with 8-Bit Data Width

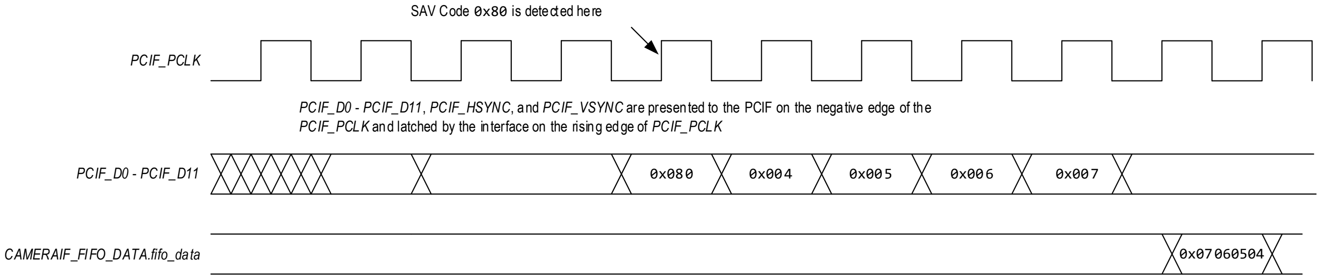


PCIF_D0 - PCIF_D11, *PCIF_HSYNC*, and *PCIF_VSYNC* are presented to the PCIF on the negative edge of the *PCIF_PCLK* input signal and latched by the interface on the rising edge of *PCIF_PCLK*

CAMERAIF_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
CAMERAIF_CTRL.data_width = 0
CAMERAIF_CTRL.timing_sel = 0

Figure 16-2: Data Stream Timing Mode with 8-Bit Data Width



PCIF_D0 - PCIF_D11, *PCIF_HSYNC*, and *PCIF_VSYNC* are presented to the PCIF on the negative edge of the *PCIF_PCLK* and latched by the interface on the rising edge of *PCIF_PCLK*

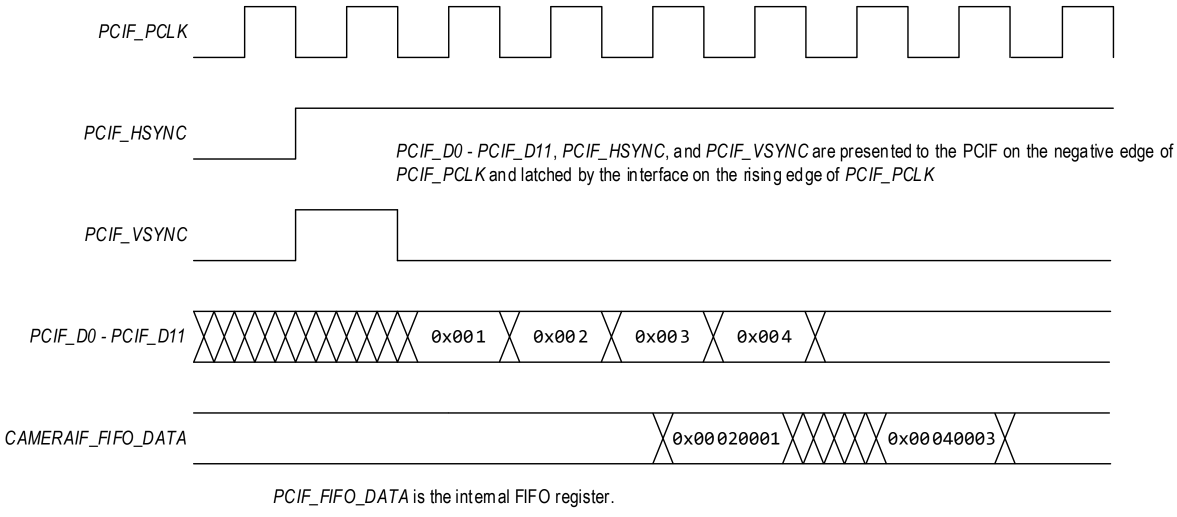
CAMERAIF_FIFO_DATA.data is the internal FIFO register. Data on *PCIF_Dn* is ignored until the SAV Code 0x80 is detected.

PCIF Register Configuration
CAMERAIF_DS_TIMING_CODES.sav = 0x80
CAMERAIF_CTRL.data_width = 0
CAMERAIF_CTRL.timing_sel = 1

16.4.2 10 and 12-bit Width

Setting *CAMERAIF_CTRL.data_width* to 1 sets the recognized pixel width on the PCIF_Dx bus to 10-bits. Set *CAMERAIF_CTRL.data_width* to 2 to set the recognized pixel width on the PCIF_Dx bus to 12-bits. As with the 8-bit width setting, the pixel data is framed as 32-bit words before these words are transferred to the 32-bit wide data FIFO *CAMERAIF_FIFO_DATA* and made ready to be read. These pixel widths are MSB zero-padded to 16-bits, and 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 16-3* for a PCIF_VSYNC/PCIF_HSYNC timing example.

Figure 16-3: 10 or 12-bit PCIF_VSYNC/PCIF_HSYNC



PCIF_D0 - PCIF_D11, *PCIF_HSYNC*, and *PCIF_VSYNC* are presented to the PCIF on the negative edge of *PCIF_PCLK* and latched by the interface on the rising edge of *PCIF_PCLK*

PCIF_FIFO_DATA is the internal FIFO register.

PCIF Register Configuration

CAMERAIF_CTRL.data_width = 1 *or* 2

CAMERAIF_CTRL.timing_sel = 0

16.5 Data FIFO

The data FIFO *CAMERAIF_FIFO_DATA* 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 *CAMERAIF_CTRL.fifo_thrsh*. The *CAMERAIF_INT_FL.fifo_thresh* is set if the data FIFO depth becomes greater than or equal to *CAMERAIF_CTRL.fifo_thrsh*. An interrupt can be generated when this condition happens if *CAMERAIF_INT_EN.fifo_thresh* is set. The data FIFO also provides status flags for FIFO full (*CAMERAIF_INT_FL.fifo_full*)and FIFO not empty (*CAMERAIF_INT_FL.fifo_not_empty*). Both status flags have associated interrupts (*CAMERAIF_INT_EN.fifo_full* and *CAMERAIF_INT_EN.fifo_not_empty*) that can be enabled and triggered when the status flags are set.

16.6 Usage

16.6.1 DMA

1. Set *CAMERAIF_CTRL.data_width* and *CAMERAIF_CTRL.ds_timing_en* as required by the camera sensor attached.
2. Enable the *CAMERAIF_INT_EN.img_done* to generate an interrupt once the image is complete.
3. Set *CAMERAIF_CTRL.read_mode* for a single image or continuous capture. Triggering the camera sensor to output an image starts the PCI automatically.
4. Set the *CAMERAIF_CTRL.rx_dma_thrsh* field to the desired FIFO level required to trigger a DMA threshold event.
5. Enable the receive DMA by setting the *CAMERAIF_CTRL.rx_dma* field to 1.
6. Enable the CAMERAIF by setting the *CAMERAIF_CTRL.pcif_sys* field to 1.
7. As data is read from the camera sensor by the CAMERAIF, it triggers 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 CAMERAIF triggers the *CAMERAIF_INT_EN.img_done* interrupt.
8. The interrupt handler can then reset the interrupt flag by writing 1 to *CAMERAIF_INT_FL.img_done.*

16.6.2 Interrupts

1. Set *CAMERAIF_CTRL.data_width* and *CAMERAIF_CTRL.ds_timing_en* as required by the camera sensor attached.
2. Set *CAMERAIF_CTRL.fifo_thrsh* to the desired level to allow the interrupt to service the FIFO before it fills.
3. Enable the *CAMERAIF_INT_EN.img_done* and the *CAMERAIF_INT_EN.fifo_thresh* interrupts to generate an interrupt when the image is complete or the FIFO is filled to the threshold level set in the threshold field (*CAMERAIF_CTRL.fifo_thrsh*).
4. Set *CAMERAIF_CTRL.read_mode* for a single image or continuous capture. When the camera sensor is triggered to output an image, the CAMERAIF automatically starts receiving data.
5. Enable the CAMERAIF by setting the *CAMERAIF_CTRL.pcif_sys* field to 1.
6. As data is read from the camera sensor by the PCIF, the hardware triggers an interrupt when the FIFO threshold *CAMERAIF_CTRL.fifo_thrsh* is met. The interrupt handler should perform a burst read from the FIFO (*CAMERAIF_FIFO_DATA.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 a *CAMERAIF_INT_EN.img_done* interrupt.
7. After servicing an image done interrupt, the interrupt handler must reset the image done interrupt flag by writing 1 to the *CAMERAIF_INT_FL.img_done.*
8. The software should check *CAMERAIF_INT_FL.fifo_not_empty* and perform a read of *CAMERAIF_FIFO_DATA.data* to receive the remainder of the words of data that occupy the FIFO less than *CAMERAIF_CTRL.fifo_thrsh*. When all of the data is read from the FIFO, hardware clears the *CAMERAIF_INT_FL.fifo_not_empty* flag automatically.

16.7 Registers

See *Table 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 16-3: Parallel Camera Interface Register Summary

Offset	Register	Name
[0x0000]	*CAMERAIF_VER*	*CAMERAIF Revision Register*
[0x0004]	*CAMERAIF_FIFO_SIZE*	*CAMERAIF FIFO Size Register*
[0x0008]	*CAMERAIF_CTRL*	*CAMERAIF Configuration Register*
[0x000C]	*CAMERAIF_INT_EN*	*CAMERAIF Interrupt Enable Register*
[0x0010]	*CAMERAIF_INT_FL*	*CAMERAIF Status Flag Register*
[0x0014]	*CAMERAIF_DS_TIMING_CODES*	*CAMERAIF Timing Code Register*
[0x0030]	*CAMERAIF_FIFO_DATA*	*CAMERAIF FIFO Data Register*

16.7.1 Parallel Camera Register Details

Table 16-4: CAMERAIF Version Register

Version			CAMERAIF_VER		[0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**	
31:16	-	RO	0	**Reserved**	
15:8	major	RO	*	**Major Revision** This field returns the major revision number of the CAMERAIF.	
7:0	minor	RO	*	**Minor Revision** This field returns the minor revision number of the CAMERAIF.	

Table 16-5: CAMERAIF FIFO Size Register

FIFO Size		**CAMERAIF_FIFO_SIZE**		**[0x0004]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
7:0	fifo_size	RO	8	**FIFO Size** This field returns the size of the CAMERAIF FIFO in words. 8: FIFO size is 8 words

Table 16-6: CAMERAIF Configuration Register

Configuration		**CAMERAIF_CTRL**		**[0x0008]**
Bits	**Field**	**Access**	**Reset**	**Description**
31	pcif_sys	R/W	0	**Camera Interface Enable** Set this field to 1 to enable the Camera interface. 0: Camera interface disabled 1: Camera interface enabled
30	three_ch_en	R/W	0	**CNN Mode Enable** Enabling CNN mode pads 5:6:5 and similar camera modes into 8:8:8, left aligns the pixels, and pads the top byte, resulting in 32-bit data. This mode unpacks 15/16 bits of camera data into 32 bits enabling the pushing of camera data into the CNN without any additional byte shuffling. 0: CNN mode disabled 1: CNN mode enabled
29:16	-	RO	0	**Reserved**
30:17	rx_dma_thrsh	R/W	1	**DMA Threshold** Set this field to the value of the receive FIFO level to trigger a DMA request. The DMA threshold event occurs when the FIFO level is equal to or greater than the setting in this field. *Note: This field is only used if the CAMERAIF_CTRL.rx_dma is set to 1.* 0: Invalid, do not set this field to 0 1: The receive DMA threshold event occurs when the FIFO level is greater than or equal to 1. … … 8: The receive DMA threshold event occurs when the FIFO level is equal to 8.
16	rx_dma	R/W	0	**Receive DMA Enable** Write this field to 1 to enable receive DMA requests 0: Receive DMA events are disabled, and any pending events are cleared 1: Receive DMA events are enabled
15:10	-	RO	0	**Reserved**

Configuration		**CAMERAIF_CTRL**		**[0x0008]**
Bits	**Field**	**Access**	**Reset**	**Description**
9:5	fifo_thrsh	R/W	1	**Data FIFO Threshold Setting** If the number of words in the FIFO is greater than or equal to this value, the *CAMERAIF_INT_FL.fifo_thresh* field is set to 1. 0: Invalid, do not set this field to 0. 1: FIFO threshold equals 1 word … … 8: FIFO threshold equals 8 words 9 - 31: Reserved
4	ds_timing_en	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 through the SAV and EAV codes.
3:2	data_width	R/W	0	**Camera Data Width** Set this field to the width of the camera's data. 0: 8-bit data 1: 10-bit data 2: 12-bit data 3: Reserved *Note: Unused PCIF_Dx pins are ignored.*
1:0	read_mode	R/W	0	**Camera Read Mode** Set this field to the required camera read mode. Setting this field to 0 disables the CAMERAIF. 0: Disabled 1: Single image capture 2: Continuous capture 3: Reserved

Table 16-7: CAMERAIF Interrupt Enable Register

Interrupt Enable		**CAMERAIF_INT_EN**		**[0x000C]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved**
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 (*CAMERAIF_INT_FL.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 (*CAMERAIF_INT_FL.fifo_thresh* = 1). 0: Interrupt Disabled 1: Interrupt Enabled

Interrupt Enable		CAMERAIF_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 (*CAMERAIF_INT_FL.fifo_full* = 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 (*CAMERAIF_INT_FL.img_done* = 1). 0: Interrupt Disabled 1: Interrupt Enabled

Table 16-8: CAMERAIF Status Flags Register

Status Flags		CAMERAIF_INT_FL		[0x0010]
Bits	**Field**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved**
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 *CAMERAIF_CTRL.fifo_thrsh* field. When the level in the FIFO falls below the set threshold, this field is automatically cleared to 0 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, *CAMERAIF_DS_TIMING_CODES.eav*, is detected. 0: End of the image not detected 1: End of the image detected

Table 16-9: CAMERAIF Timing Codes Register

Camera Timing Codes		CAMERAIF_DS_TIMING_CODES		[0x0014]
Bits	**Field**	**Access**	**Reset**	**Description**
31:16	-	RO	0	**Reserved**

Camera Timing Codes		CAMERAIF_DS_TIMING_CODES		[0x0014]
Bits	**Field**	**Access**	**Reset**	**Description**
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 *CAMERAIF_DS_TIMING_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 *CAMERAIF_DS_TIMING_CODES.eav*. Set this field to the camera's start active video field, which may differ from the reset default of 0x80.

Table 16-10: CAMERAIF FIFO Data Register

FIFO Data		CAMERAIF_FIFO_DATA		[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.

17. MIPI CSI-2 Camera Interface (CSI2)

Placeholder content for the MIPI CSI-2 chapter. Additional details will be provided in a future release of the MAX78002 user guide.

17.1 CSI-2 Registers

See *Table 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 17-1: MIPI CSI2 Register Summary

Offset	Register	Description
[0x0000]	*CSI_CFG_NUM_LANES*	Number of Lanes Configuration Register
[0x0004]	*CSI_CFG_CLK_LANE_EN*	Clock Lane Configuration Register
[0x0008]	*CSI_CFG_DATA_LANE_EN*	Data Lane Enable Register
[0x000C]	*CSI_CFG_FLUSH_COUNT*	Flush Count Register
[0x0010]	*CSI_CFG_BIT_ERR*	Bit Error Register
[0x0014]	*CSI_IRQ_STATUS*	Interrupt Status Register
[0x0018]	*CSI_IRQ_ENABLE*	Interrupt Enable Register
[0x001C]	*CSI_IRQ_CLR*	Interrupt Clear Register
[0x0020]	*CSI_ULPS_CLK_STATUS*	Ultra-Low-Power State (ULPS) Clock Register
[0x0024]	*CSI_ULPS_STATUS*	Receive D-PHY ULPS Status Register
[0x0028]	*CSI_ULPS_CLK_MARK_STATUS*	Mark Status of Receive ULPS State Register
[0x002C]	*CSI_ULPS_MARK_STATUS*	Mark Status of Receive D-PHY ULPS State Register
[0x0030]	*CSI_PPI_ERRSOT_HS*	
[0x0034]	*CSI_PPI_ERRSOTSYNC_HS*	
[0x0038]	*CSI_PPI_ERRESC*	
[0x003C]	*CSI_PPI_ERRSYNCESC*	
[0x0040]	*CSI_PPI_ERRCONTROL*	PPI Control Error Register
[0x0044]	*CSI_CFG_CPHY_EN*	C-PHY Enable Register
[0x0048]	*CSI_CFG_PPI_16_EN*	PPI 16 Bit Enable Register
[0x004C]	*CSI_CFG_PACKET_INTERFACE_EN*	Packet Interface Configuration Register
[0x0050]	*CSI_CFG_VCX_EN*	Virtual Channel Extension Configuration Register
[0x0054]	*CSI_CFG_BYTE_DATA_FORMAT*	Byte Data Configuration Register
[0x0058]	*CSI_CFG_DISABLE_PAYLOAD_0*	Disable Payload 0 Configuration Register
[0x005C]	*CSI_CFG_DISABLE_PAYLOAD_1*	Disable Payload 1 Configuration Register
[0x0080]	*CSI_CFG_VID_IGNORE_VC*	CSI-2 RX Controller IGNORE_VC Configuration Register
[0x0084]	*CSI_CFG_VID_VC*	CSI-2 RX Controller VC Configuration Register
[0x0088]	*CSI_CFG_P_FIFO_SEND_LEVEL*	CSI-2 RX Controller P_FIFO Send Level Configuration Register
[0x008C]	*CSI_CFG_VID_VSYNC*	CSI-2 RX Controller VSYNC Configuration Register
[0x0090]	*CSI_CFG_VID_HSYNC_FP*	CSI-2 RX Controller HSYNC_FP Configuration Register
[0x0094]	*CSI_CFG_VID_HSYNC*	CSI-2 RX Controller HSYNC Configuration Register
[0x0098]	*CSI_CFG_VID_HSYNC_BP*	CSI-2 RX Controller HSYNC_BP Configuration Register
[0x0400]	*CSI_CFG_DATABUS16_SEL*	High Speed Mode Data Bus Configuration Register
[0x0404]	*CSI_CFG_D0_SWAP_SEL*	Data Lane 0 Configuration Register
[0x0408]	*CSI_CFG_D1_SWAP_SEL*	Data Lane 1 Configuration Register

Offset	Register	Description
[0x040C]	CSI_CFG_D2_SWAP_SEL	Data Lane 2 Configuration Register
[0x0410]	CSI_CFG_D3_SWAP_SEL	Data Lane 3 Configuration Register
[0x0414]	CSI_CFG_C0_SWAP_SEL	Clock Lane Control Configuration Register
[0x0418]	CSI_CFG_DPDN_SWAP	Data Lane Swap Configuration Register
[0x041C]	CSI_RG_CFGCLK_1US_CNT	Reference Clock Counter Configuration Register
[0x0420]	CSI_RG_HSRX_CLK_PRE_TIME_GRP0	Pre-Zero Timing Clock Lane 0 Configuration Register
[0x0424]	CSI_RG_HSRX_DATA_PRE_TIME_GRP0	Pre-Zero Timing Data Lanes Configuration Register
[0x0428]	CSI_RESET_DESKEW	Reset De-Skew Configuration Register
[0x042C]	CSI_PMA_RDY	Physical Medium Attachment (PMA) Circuit Ready Register
[0x0490]	CSI_RG_CDRX_DSIRX_EN	DSI Receive Enable Register
[0x0494]	CSI_RG_CDRX_L012_SUBLVDS_EN	Sub-Low-Voltage Differential Signaling Mode Enable Register
[0x0498]	CSI_RG_CDRX_L012_HSRT_CTRL	High-Speed Receive Termination Enable Register
[0x04A8]	CSI_DBG1_MUX_SEL	Debug MUX Selection Register
[0x04AC]	CSI_DBG2_MUX_SEL	Debug MUX Selection Register
[0x04B0]	CSI_DBG1_MUX_DOUT	Debug MUX Output Register
[0x04B4]	CSI_DBG2_MUX_DOUT	Debug MUX Output Register
[0x04B8]	CSI_AON_POWER_READY_N	Power Ready Signal to DPHY Register
[0x04BC]	CSI_DPHY_RST_N	Reset Control to DPHY Register
[0x04C0]	CSI_RXBYTECLKHS_INV	Invert PPI Input Clock from DPHY Register
[0x0500]	CSI_VFIFO_CFG0	Video FIFO Configuration Register 0
[0x0504]	CSI_VFIFO_CFG1	Video FIFO Configuration Register 1
[0x0508]	CSI_VFIFO_CTRL	Video FIFO Control Register
[0x050C]	CSI_VFIFO_STS	Video FIFO Status Register
[0x0510]	CSI_VFIFO_LINE_NUM	Video FIFO CSI Line Number Per Frame Register
[0x0514]	CSI_VFIFO_PIXEL_NUM	Video FIFO CSI Pixel Number Per Line Register
[0x0518]	CSI_VFIFO_LINE_CNT	Video FIFO CSI Line Count Register
[0x051C]	CSI_VFIFO_PIXEL_CNT	Video FIFO CSI Pixel Count Register
[0x0520]	CSI_VFIFO_FRAME_STS	Video FIFO Frame Status Register
[0x0524]	CSI_VFIFO_RAW_CTRL	Video FIFO RAW-to-RGB Control Register
[0x0528]	CSI_VFIFO_RAW_BUF0_ADDR	Video FIFO RAW-to-RGB Line Buffer 0 Address Register
[0x052C]	CSI_VFIFO_RAW_BUF1_ADDR	Video FIFO RAW-to-RGB Line Buffer 1 Address Register
[0x0530]	CSI_VFIFO_AHBM_CTRL	Video FIFO AHB Master Control Register
[0x0534]	CSI_VFIFO_AHBM_STS	Video FIFO AHB Master Status Register
[0x0538]	CSI_VFIFO_AHBM_START_ADDR	Video FIFO AHB Master Start Address Register
[0x053C]	CSI_VFIFO_AHBM_ADDR_RANGE	Video FIFO AHB Master Address Range Register
[0x0540]	CSI_VFIFO_AHBM_MAX_TRANS	Video FIFO AHB Master Maximal Transfer Number Register
[0x0544]	CSI_VFIFO_AHBM_TRANS_CNT	Video FIFO AHB Master Transfer Count Register
[0x0600]	CSI_RX_EINT_VFF_IE	CSI2 Video FIFO Interrupt Enable Register
[0x0604]	CSI_RX_EINT_VFF_IF	CSI2 Video FIFO Interrupt Flag Register
[0x0608]	CSI_RX_EINT_PPI_IE	CSI2 DPHY Interrupt Enable Register
[0x060C]	CSI_RX_EINT_PPI_IF	CSI2 DPHY FIFO Interrupt Flag Register
[0x0610]	CSI_RX_EINT_CTRL_IE	CSI2 RX Controller Interrupt Enable Register
[0x0614]	CSI_RX_EINT_CTRL_IF	CSI2 RX Controller Interrupt Flag Register
[0x0700]	CSI_PPI_STOPSTATE	DPHY PPI Stop State Register

Offset	Register	Description
[0x0704]	*CSI_PPI_TURNAROUND_CFG*	DPHY PPI Turn-Around Configuration Register

17.1.1 Register Details

Table 17-2: Number of Lanes Configuration Register

Number of Lanes Configuration			CSI_CFG_NUM_LANES	[0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved**
3:0	lanes	R/W	0	**Number of Lanes**

Table 17-3: Configuration Clock Lane Enable Register

Configuration Clock Lane Enable			CSI_CFG_CLK_LANE_EN	[0x0004]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
0	en	R/W	0	**Lane Clock En** 0: Disabled 1: Enabled

Table 17-4: Configuration Data Lane Enable Register

Configuration Data Lane Enable			CSI_CFG_DATA_LANE_EN	[0x0008]
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
7:0	en	R/W	0	**Data Lane En**

Table 17-5: Configuration Flush Count Register

Configuration Flush Count			CSI_CFG_FLUSH_COUNT	[0x000C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved**
3:0	count	R/W	0	**Flush Count**

Table 17-6: Configuration Bit Error Register

Configuration Bit Error			CSI_CFG_BIT_ERR	[0x0010]
Bits	**Field**	**Access**	**Reset**	**Description**
31:10	-	RO	0	**Reserved**
9	vid_err_fifo_wr_ov	RC	0	**Video Error Fifo Write Overflow Error**
8	vid_err_send_lvl	RC	0	**Video Error Send Level Error**
7	crc	RC	0	**CRC Error**
6:2	header	RC	0	**Header Bit Location for Single Bit ECC Error**
1	sbe	RC	0	**Single-Bit ECC Error**
0	mbe	RC	0	**Multiple-Bit ECC Error**

Table 17-7: Interrupt Status Register

Interrupt Status			**CSI_IRQ_STATUS**	**[0x0014]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:7	-	RO	0	**Reserved**
6	vid_err_fifo_wr_ov	R	0	**Video Error FIFO Write Overflow Error**
5	vid_err_send_lvl	R	0	**Video Error Send Level Error**
4	ulps_mark_active	R	0	**ULPS Mark Active Status Change**
3	ulps_active	R	0	**ULPS Active Status Change**
2	mbe	R	0	**Two Bit ECC Error**
1	sbe	R	0	**One Bit ECC Error**
0	crc	R	0	**CRC Error**

Table 17-8: Interrupt Enable Register

Interrupt Enable			**CSI_IRQ_ENABLE**	**[0x0018]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:7	-	RO	0	**Reserved**
6	vid_err_fifo_wr_ov	R/W	0	**Video Error FIFO Write Overflow Error**
5	vid_err_send_lvl	R/W	0	**Video Error Send Level Error**
4	ulps_mark_active	R/W	0	**ULPS Mark Active Status Change**
3	ulps_active	R/W	0	**ULPS Active Status Change**
2	mbe	R/W	0	**Two Bit ECC Error**
1	sbe	R/W	0	**One Bit ECC Error**
0	crc	R/W	0	**CRC Error**

Table 17-9: Interrupt Clear Register

Interrupt Clear			**CSI_IRQ_CLR**	**[0x001C]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:7	-	RO	0	**Reserved**
6	vid_err_fifo_wr_ov	R/W1C	0	**Video Error FIFO Write Overflow Error** Write 1 to clear the corresponding interrupt flag in the *CSI_IRQ_STATUS* register.
5	vid_err_send_lvl	R/W1C	0	**Video Error Send Level Error** Write 1 to clear the corresponding interrupt flag in the *CSI_IRQ_STATUS* register.
4	ulps_mark_active	R/W1C	0	**ULPS Mark Active Status Change** Write 1 to clear the corresponding interrupt flag in the *CSI_IRQ_STATUS* register.
3	ulps_active	R/W1C	0	**ULPS Active Status Change** Write 1 to clear the corresponding interrupt flag in the *CSI_IRQ_STATUS* register.
2	mbe	R/W1C	0	**Two Bit ECC Error** Write 1 to clear the corresponding interrupt flag in the *CSI_IRQ_STATUS* register.

Interrupt Clear		CSI_IRQ_CLR		[0x001C]
Bits	**Field**	**Access**	**Reset**	**Description**
1	sbe	R/W1C	0	**One Bit ECC Error** Write 1 to clear the corresponding interrupt flag in the *CSI_IRQ_STATUS* register.
0	crc	R/W1C	0	**CRC Error** Write 1 to clear the corresponding interrupt flag in the *CSI_IRQ_STATUS* register.

Table 17-10: ULPS Clock Status Register

ULPS Clock Status		CSI_ULPS_CLK_STATUS		[0x0020]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
0	fifo	R	0	**FIFO Read/Write**

Table 17-11: ULPS Status Register

ULPS Status		CSI_ULPS_STATUS		[0x0024]
Bits	**Field**	**Access**	**Reset**	**Description**
31:2	-	RO	0	**Reserved**
1	data_lane1	R	0	**Data Lane 1** 0: Data lane 1 not in ULPS 1: Data lane 1 in ULPS
0	data_lane0	R	0	**Data Lane 0** 0: Data lane 0 not in ULPS 1: Data lane 0 in ULPS

Table 17-12: ULPS Clock Mark Status Register

ULPS Clock Mark Status		CSI_ULPS_CLK_MARK_STATUS		[0x0028]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
0	clk_lane	R	0	**Mark Status of RX ULPS State** 0: Clock lane not in mark state 1: Clock lane in mark state

Table 17-13: ULPS Mark Status Register

ULPS Mark Status		CSI_ULPS_MARK_STATUS		[0x002C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:2	-	RO	0	**Reserved**
1	data_lane1	R	0	**Mark Status of RX ULPS State for Data Lane 1** 0: Data lane 1 not in mark state 1: Data lane 1 in mark state
0	data_lane0	R	0	**Mark Status of RX ULPS State for Data Lane 0** 0: Data lane 0 not in mark state 1: Data lane 0 in mark state

Table 17-14: PHY Protocol Interface(PPI) Start of Transmission (SoT) Error Register

PPI SoT Error		**CSI_PPI_ERRSOT_HS**		**[0x0030]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:2	-	RO	0	**Reserved**
1:0	err	R	0	**SoT Error** This field is set to 1 by hardware if the high-speed SoT leader sequence is corrupted in such a way that proper synchronization can still be achieved. This is considered a “soft error” in the leader sequence and confidence in the payload data is reduced. 0: Normal operation 1: Error occurred

Table 17-15: PPI SoT Synchronization Error Register

PPI SoT Synchronization Error		**CSI_PPI_ERRSOTSYNC_HS**		**[0x0034]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:2	-	RO	0	**Reserved**
1:0	err	R	0	**SoT Synchronization Error** This field is set to 1 by hardware if the high-speed SoT leader sequence is corrupted in such a way that proper synchronization can not achieved. 0: Normal operation 1: Error occurred

Table 17-16: PPI Escape Entry Error Register

PPI Escape Entry Error		**CSI_PPI_ERRESC**		**[0x0038]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:2	-	RO	0	**Reserved**
1:0	err	R	0	**Escape Entry Error** This field is set to 1 by hardware if an unrecognized escape entry command is received. The only escape entry command supported by the receiver is the ULPS. 0: Normal operation 1: Error occurred

Table 17-17: PPI Escape Synchronization Error Register

PPI Escape Synchronization Error		**CSI_PPI_ERRSYNCESC**		**[0x003C]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:2	-	RO	0	**Reserved**
1:0	err	R	0	**Escape Entry Error** This field is set to 1 by hardware if an unrecognized escape entry command is received. The only escape entry command supported by the receiver is the ULPS. 0: Normal operation 1: Error occurred

Table 17-18: PPI Control Error Register

PPI Control Error			**CSI_PPI_ERRCONTROL**	**[0x0040]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:2	-	RO	0	**Reserved**
1:0	err	R	0	**Control Error** This field is set to 1 by hardware if an incorrect line state sequence is detected. For example, if a turn-around request or escape mode request is immediately followed by a stop state instead of the required bridge state, this signal is asserted and remains asserted until the next change in line state. 0: Normal operation 1: Error occurred

Table 17-19: Configuration C-PHY Enable Register

C-PHY Enable			**CSI_CFG_CPHY_EN**	**[0x0044]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
0	en	R/W	0	**C-PHY Enable** 0: Disabled 1: Enabled

Table 17-20: Configuration PPI 16 Bit Enable Register

PPI 16 Bit Enable			**CSI_CFG_PPI_16_EN**	**[0x0048]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
0	en	R/W	0	**PPI 16-Bit Enable** 0: Disabled (8-bit mode) 1: Enabled

Table 17-21: Packet Interface Configuration Register

Packet Interface Configuration			**CSI_CFG_PACKET_INTERFACE_EN**	**[0x004C]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
0	en	R/W	0	**Configuration Packet Interface Enable**

Table 17-22: Virtual Channel Extension Configuration Register

Virtual Channel Extension Configuration			**CSI_CFG_VCX_EN**	**[0x0050]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
0	en	R/W	0	**Virtual Channel Extension Enable**

Table 17-23: Byte Data Configuration Register

Byte Data Configuration		CSI_CFG_BYTE_DATA_FORMAT		[0x0054]
Bits	**Field**	**Access**	**Reset**	**Description**
31:2	-	RO	0	**Reserved**
1:0	format	R/W	0	**Reserved** This field must be set to 0 to maintain compatibility with future devices.

Table 17-24: Disable Payload 0 Configuration Register

Disable Payload 0 Configuration		CSI_CFG_DISABLE_PAYLOAD_0		[0x0058]
Bits	**Field**	**Access**	**Reset**	**Description**
31	raw20	RO	0	**Reserved**
30	raw16			**RAW16 Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
29	raw14			**RAW14 Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
28	raw12			**RAW12 Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
27	raw10			**RAW10 Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
26	raw8			**RAW8 Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
25	raw7			**RAW7 Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
24	raw6	R/W	0	**RAW6 Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
23:21	-	RO	0	**Reserved**
20	rgb888	R/W	0	**RGB888 Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.

Disable Payload 0 Configuration			CSI_CFG_DISABLE_PAYLOAD_0	[0x0058]
Bits	**Field**	**Access**	**Reset**	**Description**
19	rgb666	R/W	0	**RGB666 Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
18	rgb565	R/W	0	**RGB565 Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
17	rgb555	R/W	0	**RGB555 Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
16	rgb444	R/W	0	**RGB444 Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
15	yuv422_10bit	R/W	0	**YUV422 10-Bit Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
14	yuv422_8bit	R/W	0	**YUV422 8-Bit Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
13	yuv420_10bit_csp	R/W	0	**YUV420 10-Bit CSP Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
12	yuv420_8bit_csp	R/W	0	**YUV420 8-Bit CSP Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
11	-	RO	0	**Reserved**
10	yuv420_8bit_leg	R/W	0	**YUV420 8-Bit Legacy Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
9	yuv420_10bit	R/W	0	**YUV420 10-Bit Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.

Disable Payload 0 Configuration			CSI_CFG_DISABLE_PAYLOAD_0	[0x0058]
Bits	**Field**	**Access**	**Reset**	**Description**
8	yuv420_8bit	R/W	0	**YUV420 8-Bit Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
7:3	-	RO	0	**Reserved**
2	embedded	R/W	0	**Embedded Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
1	blank	R/W	0	**Blank Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
0	null	R/W	0	**NULL Disable** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.

Table 17-25: Disable Payload 1 Configuration Register

Disable Payload 1 Configuration			CSI_CFG_DISABLE_PAYLOAD_1	[0x005C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
7	usr_def_type37	R/W	0	**User Defined Type 37** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
6	usr_def_type36	R/W	0	**User Defined Type 36** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
5	usr_def_type35	R/W	0	**User Defined Type 35** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
4	usr_def_type34	R/W	0	**User Defined Type 34** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
3	usr_def_type33	R/W	0	**User Defined Type 33** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.

Disable Payload 1 Configuration			CSI_CFG_DISABLE_PAYLOAD_1	[0x005C]
Bits	**Field**	**Access**	**Reset**	**Description**
2	usr_def_type32	R/W	0	**User Defined Type 32** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
1	usr_def_type31	R/W	0	**User Defined Type 31** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.
0	usr_def_type30	R/W	0	**User Defined Type 30** Setting this field to 0 disables payload data for the data type. When this field is set to 1 and this type of packet is received, only the packet header is presented, along with the SOP and EOP indication where the payload data would have appeared.

Table 17-26: CSI-2 RX Controller IGNORE_VC Configuration Register

CSI-2 RX Controller IGNORE_VC Configuration			CSI_CFG_VID_IGNORE_VC	[0x0080]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
0	en	R/W	0	**Ignore Video Interface Control**

Table 17-27: CSI-2 RX Controller VC Configuration Register

CSI-2 RX Controller VC Configuration			CSI_CFG_VID_VC	[0x0084]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-28: CSI-2 RX Controller P_FIFO Send Level Register

CSI-2 RX Controller P_FIFO Send Level			CSI_CFG_P_FIFO_SEND_LEVEL	[0x0088]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-29: CSI-2 RX Controller VSYNC Configuration Register

CSI-2 RX Controller VSYNC Configuration			CSI_CFG_VID_VSYNC	[0x008C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-30: CSI-2 RX Controller HSYNC_FP Configuration Register

CSI-2 RX Controller HSYNC_FP Configuration			CSI_CFG_VID_HSYNC_FP	[0x0090]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-31: CSI-2 RX Controller HSYNC Configuration Register

CSI-2 RX Controller HSYNC Configuration			CSI_CFG_VID_HSYNC	[0x0094]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-32: CSI-2 RX Controller HSYNC_BP Configuration Register

CSI-2 RX Controller HSYNC_BP Configuration			CSI_CFG_VID_HSYNC_BP	[0x0098]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-33: High Speed Mode Data Bus Configuration Register

High Speed Mode Data Bus Configuration			CSI_CFG_DATABUS16_SEL	[0x0400]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	RO	0	**Reserved**

Table 17-34: Data Lane 0 Configuration Register

Data Lane 0 Configuration			CSI_CFG_D0_SWAP_SEL	[0x0404]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
0	cdrx	R/W	0	**Pad Swap Select**

Table 17-35: Data Lane 1 Configuration Register

Data Lane 1 Configuration			CSI_CFG_D1_SWAP_SEL	[0x0408]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
0	cdrx	R/W	0	**Pad Swap Select**

Table 17-36: Data Lane 2 Configuration Register

Data Lane 2 Configuration			CSI_CFG_D2_SWAP_SEL	[0x040C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	RO	0	**Reserved**

Table 17-37: Data Lane 3 Configuration Register

Data Lane 3 Configuration			CSI_CFG_D3_SWAP_SEL	[0x0410]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	RO	0	**Reserved**

Table 17-38: Clock Lane Control Configuration Register

Clock Lane Control Configuration			CSI_CFG_C0_SWAP_SEL	[0x0414]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-39: Data Lane Swap Configuration Register

Data Lane Swap Configuration			CSI_CFG_DPDN_SWAP	[0x0418]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-40: Reference Clock Counter Configuration Register

Reference Clock Counter Configuration			CSI_RG_CFGCLK_1US_CNT	[0x041C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-41: Pre-Zero Timing Clock Lane 0 Configuration Register

Pre-Zero Timing Clock Lane 0 Configuration			CSI_RG_HSRX_CLK_PRE_TIME_GRP0	[0x0420]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-42: Pre-Zero Timing Data Lanes Configuration Register

Pre-Zero Timing Data Lanes Configuration			CSI_RG_HSRX_DATA_PRE_TIME_GRP0	[0x0424]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-43: Reset De-Skew Configuration Register

Reset De-Skew Configuration			CSI_RESET_DESKEW	[0x0428]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-44: PMA Circuit Ready Register

PMA Circuit Ready			CSI_PMA_RDY	[0x042C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-45: DSI Receive Enable Register

DSI Receive Enable			CSI_RG_CDRX_DSIRX_EN	[0x0490]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-46: Sub-LVDS Mode Enable Register

Sub-LVDS Mode Enable			CSI_RG_CDRX_L012_SUBLVDS_EN	[0x0494]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-47: High-Speed Receive Termination Enable Register

High-Speed Receive Termination Enable			CSI_RG_CDRX_L012_HSRT_CTRL	[0x0498]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-48: Debug MUX Selection Register

Debug MUX Selection			CSI_DBG1_MUX_SEL	[0x04A8]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-49: Debug MUX Selection Register

Debug MUX Selection			CSI_DBG2_MUX_SEL	[0x04AC]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-50: Debug MUX Output Register

Debug MUX Output			CSI_DBG1_MUX_DOUT	[0x04B0]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-51: Debug MUX Output Register

Debug MUX Output			CSI_DBG2_MUX_DOUT	[0x04B4]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-52: Power Ready Signal to DPHY Register

Power Ready Signal to DPHY			CSI_AON_POWER_READY_N	[0x04B8]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-53: Reset Control to DPHY Register

Reset Control to DPHY			CSI_DPHY_RST_N	[0x04BC]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-54: Invert PPI Input Clock from DPHY Register

Invert PPI Input Clock from DPHY			CSI_RXBYTECLKHS_INV	[0x04C0]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-55: Video FIFO Configuration 0 Register

Video FIFO Configuration Register 0		CSI_VFIFO_CFG0		[0x0500]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-56: Video FIFO Configuration 1 Register

Video FIFO Configuration Register 1		CSI_VFIFO_CFG1		[0x0504]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-57: Video FIFO Control Register

Video FIFO Control		CSI_VFIFO_CTRL		[0x0508]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-58: Video FIFO Status Register

Video FIFO Status		CSI_VFIFO_STS		[0x050C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-59: Video FIFO CSI Line Number Per Frame Register

Video FIFO CSI Line Number Per Frame		CSI_VFIFO_LINE_NUM		[0x0510]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-60: Video FIFO CSI Pixel Number Per Line Register

Video FIFO CSI Pixel Number Per Line		CSI_VFIFO_PIXEL_NUM		[0x0514]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-61: Video FIFO CSI Line Count Register

Video FIFO CSI Line Count		CSI_VFIFO_LINE_CNT		[0x0518]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-62: Video FIFO CSI Pixel Count Register

Video FIFO CSI Pixel Count		CSI_VFIFO_PIXEL_CNT		[0x051C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-63: Video FIFO Frame Status Register

Video FIFO Frame Status			**CSI_VFIFO_FRAME_STS**	**[0x0520]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-64: Video FIFO RAW-to-RGB Control Register

Video FIFO RAW-to-RGB Control			**CSI_VFIFO_RAW_CTRL**	**[0x0524]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-65: Video FIFO RAW-to-RGB Line Buffer 0 Address Register

Video FIFO RAW-to-RGB Line Buffer 0 Address			**CSI_VFIFO_RAW_BUF0_ADDR**	**[0x0528]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-66: Video FIFO RAW-to-RGB Line Buffer 1 Address Register

Video FIFO RAW-to-RGB Line Buffer 1 Address			**CSI_VFIFO_RAW_BUF1_ADDR**	**[0x052C]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-67: Video FIFO AHB Master Control Register

Video FIFO AHB Master Control			**CSI_VFIFO_AHBM_CTRL**	**[0x0530]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-68: Video FIFO AHB Master Status Register

Video FIFO AHB Master Status			**CSI_VFIFO_AHBM_STS**	**[0x0534]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-69: Video FIFO AHB Master Start Address Register

Video FIFO AHB Master Start Address			**CSI_VFIFO_AHBM_START_ADDR**	**[0x0538]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-70: Video FIFO AHB Master Address Range Register

Video FIFO AHB Master Address Range			**CSI_VFIFO_AHBM_ADDR_RANGE**	**[0x053C]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-71: Video FIFO AHB Master Maximal Transfer Number Register

Video FIFO AHB Master Max Transfer Number		CSI_VFIFO_AHBM_MAX_TRANS		[0x0540]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-72: Video FIFO AHB Master Transfer Count Register

Video FIFO AHB Master Transfer Count		CSI_VFIFO_AHBM_TRANS_CNT		[0x0544]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-73: CSI2 Video FIFO Interrupt Enable Register

CSI2 Video FIFO Interrupt Enable		CSI_RX_EINT_VFF_IE		[0x0600]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-74: CSI2 Video FIFO Interrupt Flag Register

CSI2 Video FIFO Interrupt Flag		CSI_RX_EINT_VFF_IF		[0x0604]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-75: CSI2 DPHY Interrupt Enable Register

CSI2 DPHY Interrupt Enable		CSI_RX_EINT_PPI_IE		[0x0608]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-76: CSI2 DPHY FIFO Interrupt Flag Register

CSI2 DPHY FIFO Interrupt Flag		CSI_RX_EINT_PPI_IF		[0x060C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-77: CSI2 RX Controller Interrupt Enable Register

CSI2 RX Controller Interrupt Enable		CSI_RX_EINT_CTRL_IE		[0x0610]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-78: CSI2 RX Controller Interrupt Flag Register

CSI2 RX Controller Interrupt Flag		CSI_RX_EINT_CTRL_IF		[0x0614]
Bits	**Field**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

Table 17-79: DPHY PPI Stop State Register

DPHY PPI Stop State				**CSI_PPI_STOPSTATE**	**[0x0700]**
Bits	**Field**	**Access**	**Reset**	**Description**	
31:1	-	RO	0	**Reserved**	

Table 17-80: DPHY PPI Turn-Around Configuration Register

DPHY PPI Turn-Around Configuration				**CSI_PPI_TURNAROUND_CFG**	**[0x0704]**
Bits	**Field**	**Access**	**Reset**	**Description**	
31:1	-	RO	0	**Reserved**	

18. 1-Wire Master (OWM)

The device provides a 1-Wire master (OWM) that the software 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.

18.1 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 derived from the current system clock source
- The OWM module clock can be pre-scaled 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 software 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 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 Analog Devices 1-Wire protocol and supporting devices, refer to the following resources:

- *AN937: Book of iButton® Standards*
  - *www.maximintegrated.com/AN937*
- *AN1796: Guide to 1-Wire Communication*
  - *www.maximintegrated.com/AN1796*
- *AN187: 1-Wire Search Algorithm*
  - *www.maximintegrated.com/AN187*

iButton is a registered trademark of Analog Devices, Inc.

18.2 1-Wire Pins and Configuration

The single instance of the peripheral is shown in *Table 18-1* and lists the alternate function names for the 1-Wire peripheral. Refer to the device data sheet's pin description table for the pin mapping of the alternate functions.

Table 18-1: MAX78002 1-Wire Master Peripheral Pins

OWM Instance	Alternate Function Name
OWM	OWM_IO
	OWM_PE

18.2.1 1-Wire I/O (OWM_IO)

The OWM_IO pin is a bidirectional I/O that is used to drive the external 1-Wire bus directly. 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.

18.2.2 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.

18.2.3 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 18-1* where $f_{PCLK} = {f_{SYSCLK}}/{2}$.

Equation 18-1: OWM 1MHz Clock Frequency

$$f_{owmclk} = 1MHz = \frac{f_{PCLK}}{OWM\_CLK\_DIV\_1US.divisor}$$

18.3 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 presence 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 software 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.

18.3.1 Networking Layers

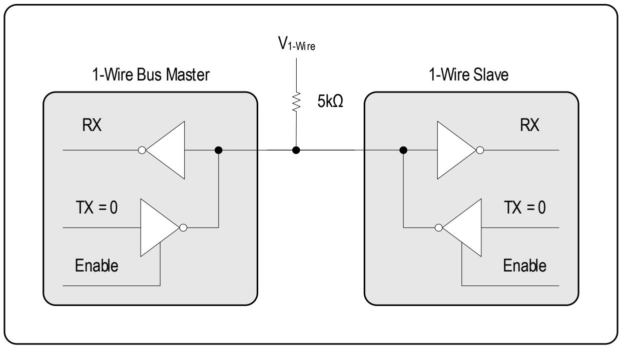
In the *Book of iButton Standards*, the 1-Wire communication protocol is described in terms of the ISO-OSI model (International Organization of Standardization (ISO) Open System Interconnection (OSI) Network Layer model). Network layers that apply to this description are the Physical, Link, Network, 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 corresponds 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 regarding the OSI Network Layer model. The Transport and Presentation layers are beyond the scope of this document.

18.3.1.1 Physical Layer

The 1-Wire communication bus consists of a single data/power line plus ground. Devices (either master or slave) interface to the 1-Wire communication bus using an open-drain (active low) connection, meaning 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 18-1*, and avoids bus contention if more than one device attempts to drive the 1-Wire bus at the same time.

Figure 18-1: 1-Wire Signal Interface



18.3.1.2 Link Layer

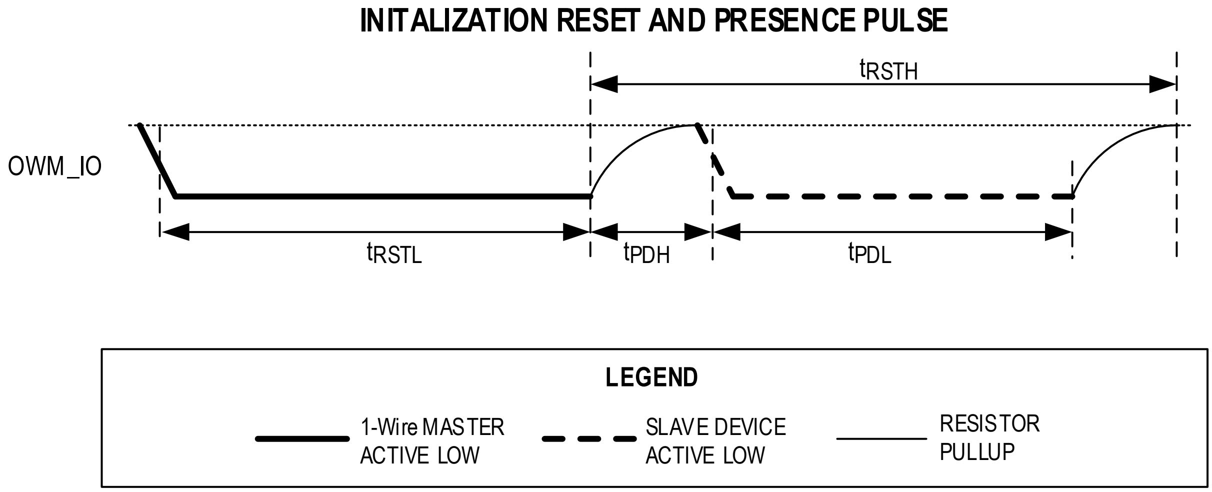
The 1-Wire Bus supports a single master and one or more slave devices (multidrop). Slave devices can connect to and disconnect from the 1-Wire Bus dynamically (as is typically the case with iButton devices that operate using an intermittent touch contact interface), which means that it is the master's responsibility to poll the bus as needed to determine the number and types of 1-Wire devices that are connected to the bus.

The OWM initiates all communication sequences on the 1-Wire Bus. The OWM determines when 1-Wire data transmissions begin and 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.

18.3.1.2.1 OWM Reset and Presence Detect

The OWM begins each communication sequence by sending a reset pulse, as shown in *Figure 18-2*. 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 18-2: 1-Wire Reset Pulse



In general, the 1-Wire line must idle in a high state when communication is not taking place. The master can 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.

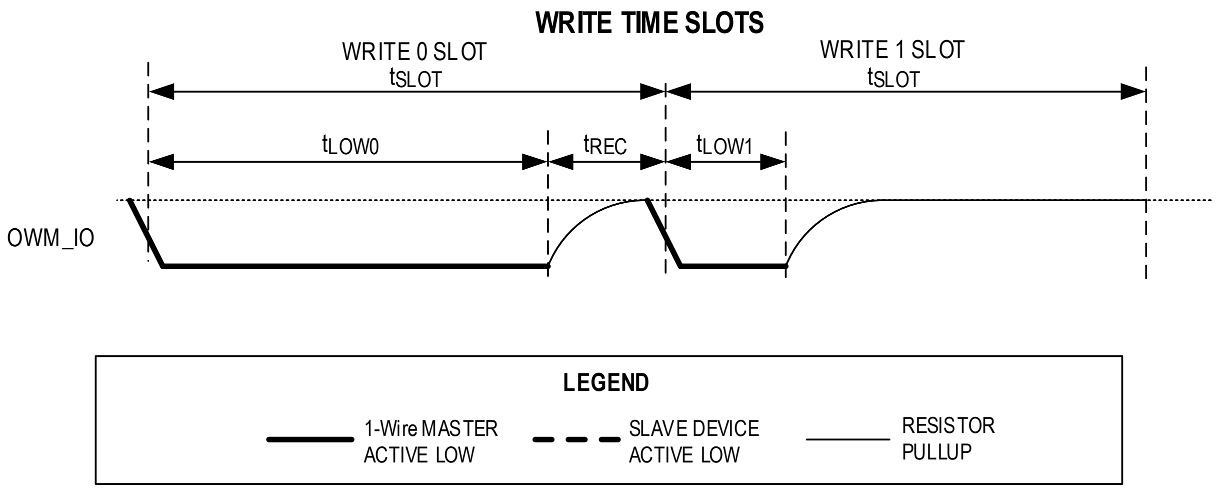
18.3.1.2.2 OWM Write Time Slot

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 if 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 18-3*, 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 18-3: 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.

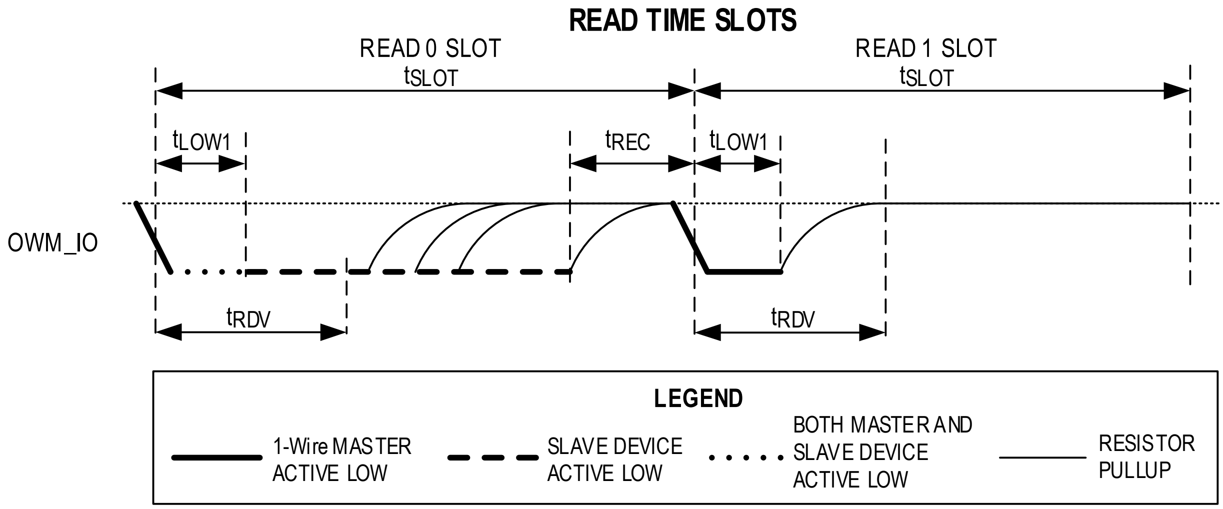
18.3.1.2.3 OWM Read Time Slot

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 18-3*. 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.

For example, *Figure 18-4* shows a sequence in which the slave device transmits data back to the 1-Wire bus master upon request. The slave device does not need to do anything to transmit a 1 bit. It simply leaves the line alone (to float high) and waits for the next time slot. The slave device holds the line low until the end of the time slot to transmit a 0 bit.

Figure 18-4: 1-Wire Read Time Slot



18.3.1.2.4 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.

18.3.1.3 Network Layer

18.3.1.3.1 ROM Commands

Following the initial 1-Wire reset pulse on the bus, all slave 1-Wire devices are active, which means 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 it intends to communicate with and deactivate all others. This is the purpose of the ROM commands (network layer) shown in *Table 18-2*.

Table 18-2: 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.

18.3.1.3.2 ROM ID

Figure 18-5 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 18-5: 1-Wire ROM ID Fields

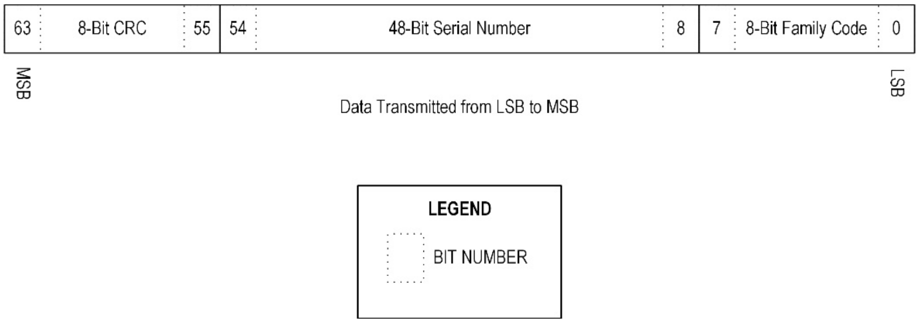


Table 18-3 provides a detailed description of each of the ROM ID fields.

Table 18-3: 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 *Book of iButton Standards*. 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 following descriptions assume, however, that the master is communicating with only one slave device at a time because this is the method 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 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.

18.3.2 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.

18.3.3 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.

18.3.4 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 is already determined. When transmitting this command, the master sends the command byte (that is, 0x55 for standard speed and 0x69 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 selected by the command to shift communication speed from standard speed to overdrive speed. The Overdrive Match ROM command byte (0x69) and the 64-bit ROM ID bits are transmitted at standard speed. All subsequent communication is sent at overdrive speed.

18.3.5 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 if 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 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.

18.3.6 Search ROM Accelerator Operation

The OWM module provides a special accelerator mode for use with the Search ROM command to allow the Search ROM command to process more quickly. 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 is 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 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 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.

18.3.7 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.

18.4 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.

18.4.1 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 the 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.

18.5 1-Wire Data Reads

18.5.1 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, the 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_data_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.

18.5.2 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 0x0FF.
3. Once the 8-bit transmission completes, the 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_data_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 0x0FF is the same as the transmitted value.*

18.6 Registers

See *Table 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 18-4: 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

18.6.1 Register Details

Table 18-5: OWM Configuration Register

OWM Configuration Register		**OWM_CFG**		**[0x0000]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
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.

OWM Configuration Register		OWM_CFG		[0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**
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.
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 18-6: OWM Clock Divisor Register

OWM Clock Divisor		OWM_CLK_DIV_1US		[0x0004]
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**

OWM Clock Divisor			OWM_CLK_DIV_1US	[0x0004]
Bits	**Field**	**Access**	**Reset**	**Description**
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 18-7: OWM Control Status Register

OWM Control Status			OWM_CTRL_STAT	[0x0008]
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
7	presence_detect	R	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**
4	od_spec_mode	R	0	**Overdrive Spec Mode** Returns the version of the overdrive spec.
3	ow_input	R	-	**OWM_IO 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 (*OWM_CFG.bit_bang_en* = 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 18-8: OWM Data Buffer Register

OWM Data		**OWM_DATA**		**[0x000C]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:8	-	RO	0	**Reserved**
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 18-9: OWM Interrupt Flag Register

OWM Interrupt Flag		**OWM_INTFL**		**[0x0010]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:5	-	RO	0	**Reserved**
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	**Receive Data Ready** Data received from the 1-Wire bus and is available in the *OWM_DATA.tx_rx* field. Write 1 to clear this flag. 0: Receive data not available. 1: Data received and is available in the *OWM_DATA.tx_rx* field.
1	tx_data_empty	R/W1C	0	**Transmit 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 *OWM_DATA.tx_rx* field has not completed transmission. 1: Data in the *OWM_DATA.tx_rx* 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 *OWM_CTRL_STAT.start_ow_reset*. Write 1 to clear this flag. 0: 1-Wire reset sequence not complete or bus idle. 1: 1-Wire reset sequence complete.

Table 18-10: OWM Interrupt Enable Register

OWM Interrupt Enable		**OWM_INTEN**		**[0x0014]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:5	-	RO	0	**Reserved**
4	line_low	R/W	0	**Line Low Interrupt Enable** Set this field to 1 to enable the I/O pin low detected interrupt. 0: Interrupt disabled. 1: Interrupt enabled.
3	line_short	R/W	0	**Line Short Interrupt Enable** Set this field to 1 to enable the I/O pin short detected interrupt. 0: Interrupt disabled. 1: Interrupt enabled.

OWM Interrupt Enable		OWM_INTEN		[0x0014]
Bits	**Field**	**Access**	**Reset**	**Description**
2	rx_data_ready	R/W	0	**Receive Data Ready Interrupt Enable** Set this field to 1 to enable the receive data ready interrupt. 0: Interrupt disabled. 1: Interrupt enabled.
1	tx_data_empty	R/W	0	**Transmit Data Empty Interrupt Enable** Set this field to 1 to enable the transmit data empty interrupt. 0: Interrupt disabled. 1: Interrupt enabled.
0	ow_reset_done	R/W	0	**1-Wire Reset Sequence Complete Interrupt Enable** Set this field to 1 to enable the 1-Wire reset sequence completed interrupt. 0: Interrupt disabled. 1: Interrupt enabled.

19. Real-Time Clock (RTC)

19.1 Overview

The 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 device data sheet for the required electrical characteristics of the external crystal.

A user-configurable, digital frequency trim is provided for applications requiring higher accuracy.

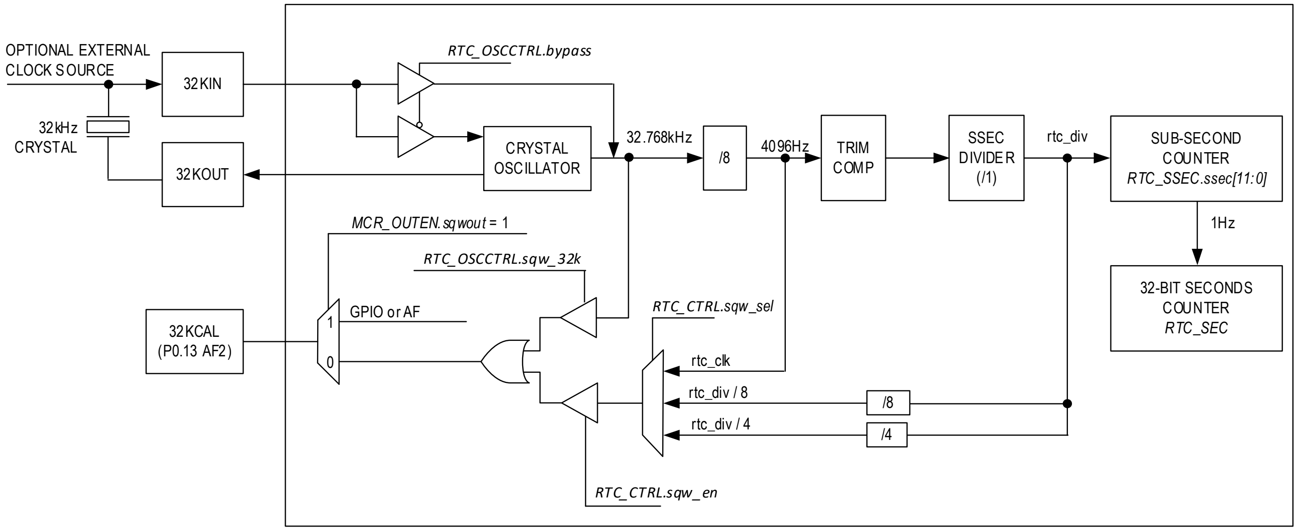
The 32-bit seconds counter register *RTC_SEC* is incremented every time there is a rollover of the *RTC_SSEC.ssec* sub-second counter field.

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.tod_alarm_ie* field.
2. A programmable sub-second alarm provides a recurring alarm using the RTC sub-second alarm register, *RTC_SSECA*, and the *RTC_CTRL.ssec_alarm* field.

The RTC is powered in the AoD. Disabling the RTC, *RTC_CTRL.en* cleared to 0, stops incrementing the *RTC_SSEC* and *RTC_SEC*, but preserves their current values. The 32kHz oscillator is not affected by the *RTC_CTRL.en* field. While the RTC is enabled (*RTC_CTRL.en* = 1), the *RTC_TRIM.vrtc_tmr* field is also incremented every 32 seconds.

Figure 19-1: MAX78002 RTC Block Diagram



19.2 Instances

One instance of the RTC peripheral is provided. The RTC counter and alarm register details and description are shown in *Table 19-1*.

Table 19-1: RTC Seconds, Sub-Seconds, Time-of-Day Alarm, and Sub-Second Alarm Register Details

Field	Width (bits)	Counter Increment	Minimum	Maximum	Description
RTC_SEC.sec	32	1 second	1 second	136 years	Seconds counter field
RTC_SSEC.ssec	12	244 µs ($\frac{1}{4096Hz}$)	244 µs	1 second	Sub-second counter field
RTC_TODA.tod_alarm	20	1 second	1 second	12 days	Time-of-day alarm field
RTC_SSECA.ssec_alarm	32	244 µs ($\frac{1}{4096Hz}$)	244 µs	12 days	Sub-second alarm field

19.3 Register Access Control

Access protection mechanisms prevent the software from accessing critical registers and fields while RTC while the hardware is updating them. Monitoring the *RTC_CTRL.busy* and *RTC_CTRL.rdy* fields allows the software to determine when it is safe to write to registers and when registers return valid results.

Table 19-2: RTC Register Access

Register	Field	Read Access	Write Access	***RTC_CTRL.busy*** **= 1 during writes**	Description
RTC_SEC	*.sec*	*RTC_CTRL.busy* = 0 *RTC_CTRL.rdy* = 1[†]	*RTC_CTRL.busy* = 0 *RTC_CTRL.rdy* = 1[†]	Y	Seconds counter
RTC_SSEC	*.ssec*	*RTC_CTRL.busy* = 0 *RTC_CTRL.rdy* = 1[†]	*RTC_CTRL.busy* = 0 *RTC_CTRL.rdy* = 1[†]	Y	Sub-second counter
RTC_TODA	*.tod_alarm*	Always	*RTC_CTRL.busy* = 0 *RTC_CTRL.tod_alarm_ie* = 0	Y	Time-of-day alarm
RTC_SSECA	*.ssec_alarm*	Always	*RTC_CTRL.busy* = 0 *RTC_CTRL.ssec_alarm_ie* = 0	Y	Sub-second alarm
RTC_TRIM	All	Always	*RTC_CTRL.busy* = 0 *RTC_CTRL.wr_en* = 1	Y	Trim
RTC_OSCCTRL	All	Always	*RTC_CTRL.wr_en* = 1	N	Oscillator control
RTC_CTRL	*en*	Always	*RTC_CTRL.busy* = 0 *RTC_CTRL.wr_en* = 1	Y	RTC enable field
	All other fields	Always	*RTC_CTRL.busy* = 0	Y	

[†] See the *RTC_SEC and RTC_SSEC Read Access Control* section for details.

19.3.1 RTC_SEC and RTC_SSEC Read Access Control

The software reads of the *RTC_SEC* and *RTC_SSEC* registers return invalid results if the read operation occurs on the same cycle that the register is being updated by the hardware (*RTC_CTRL.rdy* = 0). The hardware avoids this by setting the *RTC_CTRL.rdy* field to 1 for 120µs when the *RTC_SEC* and *RTC_SSEC* registers are valid and readable by the software.

Alternately, the software can set the *RTC_CTRL.rd_en* field to 1 to allow asynchronous reads of both *RTC_SEC* and *RTC_SSEC*.

Three methods are available to ensure valid results when reading *RTC_SEC* and *RTC_SSEC*:

1. The software clears the *RTC_CTRL.rdy* field to 0.
   a. The software polls the *RTC_CTRL.rdy* field until it reads 1 before reading the registers.
   b. The software must read the *RTC_SEC* and *RTC_SSEC* registers within 120µs to ensure valid register data.
2. The software sets the *RTC_CTRL.rdy_ie* field to 1 to generate an RTC interrupt when the hardware sets the *RTC_CTRL.rdy* field to 1.
   a. The software must service the RTC interrupt and read the *RTC_SEC*, *RTC_SSEC*, or both registers while the *RTC_CTRL.rdy* field is 1 to ensure valid data, avoiding the overhead associated with polling the *RTC_CTRL.rdy* field.
3. The software sets the *RTC_CTRL.rd_en* field to 1 enabling asynchronous reads of both the *RTC_SEC* register and the *RTC_SSEC* register.
   a. The software must read consecutive identical values of each of the *RTC_SEC* register and the *RTC_SSEC* register to ensure valid data.

19.3.2 RTC Write Access Control

The read-only status field *RTC_CTRL.busy* is set to 1 by the hardware following a software instruction that writes to specific registers. The bit remains 1 while the software updates are being synchronized into the RTC. The software should not write to any of the registers until the hardware indicates the synchronization is complete by clearing *RTC_CTRL.busy* to 0.

19.4 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 alarm register's value. The sub-second interval alarm provides an auto-reload timer driven by the trimmed RTC clock source.

19.4.1 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 *RTC_SEC*[19:0] 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. Disable the time-of-day alarm (*RTC_CTRL.tod_alarm_ie* = 0) before changing the *RTC_TODA.tod_alarm* field.

When the alarm occurs, a single event sets the time-of-day alarm interrupt flag (*RTC_CTRL.tod_alarm*) to 1.

Setting the *RTC_CTRL.tod_alarm* bit to 1 in the software results in an interrupt request to the processor if the alarm time-of-day interrupt enable (*RTC_CTRL.tod_alarm_ie*) bit is set to 1, and the RTC's system interrupt enable is set.

19.4.2 Sub-Second Alarm

The *RTC_SSECA.ssec_alarm* and *RTC_CTRL.ssec_alarm_ie* fields control the sub-second alarm. Writing *RTC_SSECA.ssec_alarm* sets the starting value for the sub-second alarm counter. Writing the sub-second alarm enable (*RTC_CTRL.ssec_alarm_ie*) bit to 1 enables the sub-second alarm. Once enabled, an internal alarm counter begins incrementing from the *RTC_SSECA.ssec_alarm* field's value. When the counter rolls over from 0xFFFF FFFF to 0x0000 0000, the hardware sets the *RTC_CTRL.ssec_alarm* bit, triggering the alarm. At the same time, the hardware also reloads the counter with the value previously written to *RTC_SSECA.ssec_alarm*.

Disable the sub-second alarm, *RTC_CTRL.ssec_alarm_ie*, before changing the interval alarm value, *RTC_SSECA.ssec_alarm*.

The delay (uncertainty) associated with enabling the sub-second alarm is up to one sub-second clock period. This uncertainty is propagated to the first interval alarm. After that, 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.

19.4.3 RTC Interrupt and Wakeup Configuration

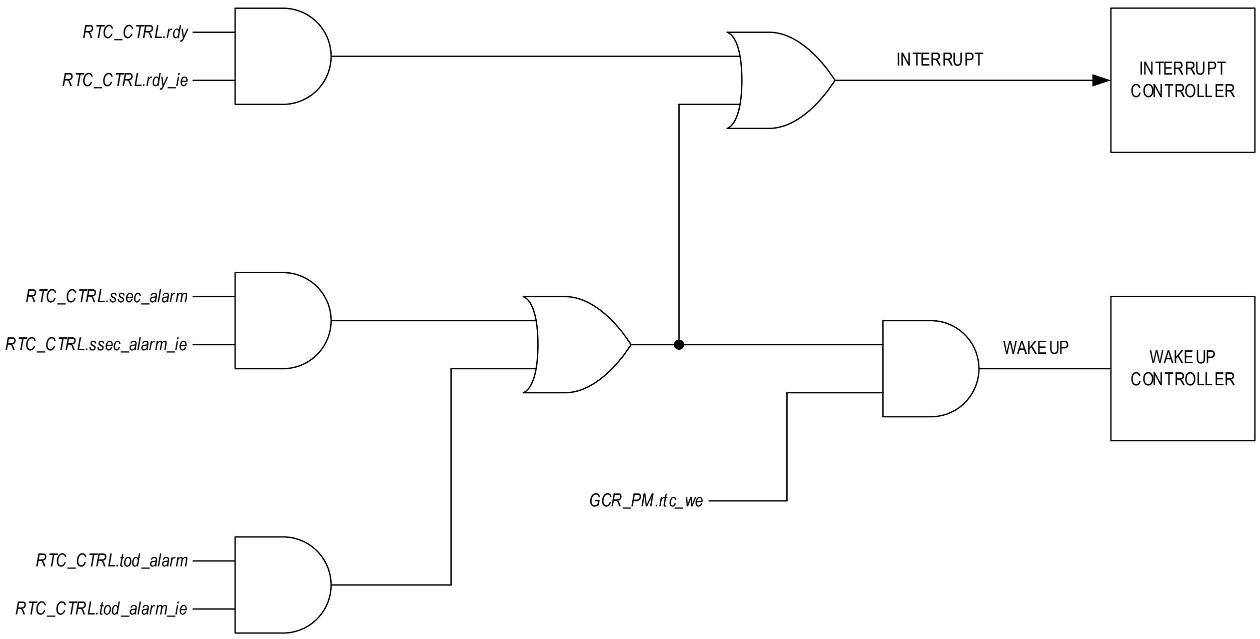
The following is a list of conditions that, when enabled, generate an RTC interrupt:

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 wake-up source for exiting the following low-power modes:

1. *BACKUP*
2. *DEEPSLEEP*
3. *SLEEP*

Figure 19-2: RTC Interrupt/Wakeup Diagram Wake-up Function



Use this procedure to enable the RTC as a wake-up source:

1. Configure the RTC interrupt enable bits, enabling one or more interrupt conditions to generate an RTC interrupt.
2. Create an RTC interrupt handler function and register the address of the RTC_IRQn using the NVIC.
3. Set the *GCR_PM.rtc_we* field to 1 to enable system wake-up by the RTC.
4. Enter the desired low-power mode. See *Operating Modes* for details.

19.5 Square Wave Output

The RTC can output a 50% duty cycle square wave signal derived from the 32kHz oscillator on a selected device pin. See *Table 19-3* for the device pins, frequency options, and control fields specific to this device. Frequencies noted as compensated in *Table 19-3* are used during the RTC frequency calibration procedure because they incorporate the frequency adjustments provided by the digital trim function.

Table 19-3: MAX78002 RTC Square Wave Output Configuration

Function	Option	Control Field
Output Pin	P3.1: SQWOUT	0
Enable Frequency Output	1Hz (Compensated)	*RTC_CTRL.sqw_sel* = 0 *RTC_CTRL.sqw_en* = 1 *RTC_OSCCTRL.sqw_32k* = 0
	512Hz (Compensated)	*RTC_CTRL.sqw_sel* = 1 *RTC_CTRL.sqw_en* = 1 *RTC_OSCCTRL.sqw_32k* = 0
	4kHz	*RTC_CTRL.sqw_sel* = 2 *RTC_CTRL.sqw_en* = 1 *RTC_OSCCTRL.sqw_32k* = 0
	32kHz	*RTC_OSCCTRL.sqw_32k* = 1

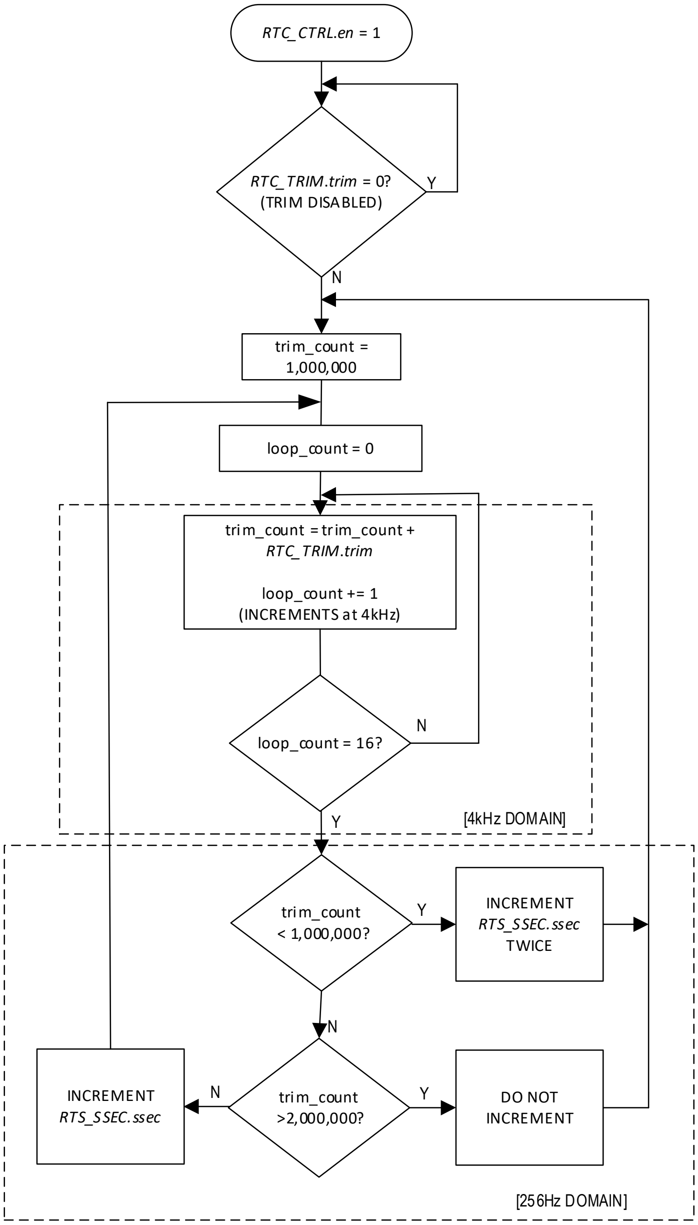
Use the following software procedure to generate and output the square wave:

1. Select the desired output frequency:
   a. Set the field *RTC_CTRL.sqw_sel* to 0 for a 1Hz compensated output frequency, or
   b. set the field *RTC_CTRL.sqw_sel* to 1 for a 512Hz compensated output frequency, or
   c. set the field *RTC_CTRL.sqw_sel* to 2 for a 4kHz output frequency, or
   d. set the field *RTC_OSCCTRL.sqw_32k* to 1 for the 32kHz frequency output.
2. Enable the system level output pin by setting the output pin shown in *Table 19-3*.
3. If the selected frequency is 1Hz, 512Hz, or 4kHz, set the *RTC_CTRL.sqw_en* field to 1 to output the selected output frequency.

19.6 RTC Calibration

A digital trim facility provides the ability to compensate for RTC inaccuracies of up to ± 127ppm when compared against an external reference clock. The trimming function utilizes an independent dedicated timer that increments the sub-second register based on a user-supplied, twos-complement value in the *RTC_TRIM* register as shown in *Figure 19-3*.

Figure 19-3: Internal Implementation of 4kHz Digital Trim



Complete the following steps to perform an RTC calibration:

1. The software must configure and enable 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. Clear the *RTC_CTRL.rdy* field to 0.
4. Wait for the *RTC_CTRL.rdy* to be set to 1 by the hardware:
    a. Set the *RTC_CTRL.rdy_ie* to 1 to generate an interrupt when the *RTC_CTRL.rdy* field is set to 1, or
    b. Poll the *RTC_CTRL.rdy* field until it reads 1.
5. Poll the *RTC_CTRL.busy* field until it reads 0 to allow any active operations to complete.
6. Set the *RTC_CTRL.wr_en* field to 1 to allow access to the *RTC_TRIM.trim* field.
7. Write a trim value to the *RTC_TRIM.trim* field to correct for any measured inaccuracy.
8. Poll the *RTC_CTRL.busy* field until it reads 0
9. Clear the *RTC_CTRL.wr_en* field to 0.
10. Repeat the process as needed until the desired accuracy is achieved.

19.7 Registers

See *Table 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 reset.

Table 19-4: RTC Register Summary

Offset	Register	Description
[0x0000]	*RTC_SEC*	RTC Seconds Counter Register
[0x0004]	*RTC_SSEC*	RTC Sub-Second Counter Register
[0x0008]	*RTC_TODA*	RTC Time-of-Day Alarm Register
[0x000C]	*RTC_SSECA*	RTC Sub-Second Alarm Register
[0x0010]	*RTC_CTRL*	RTC Control Register
[0x0014]	*RTC_TRIM*	RTC 32KHz Oscillator Digital Trim Register
[0x0018]	*RTC_OSCCTRL*	RTC 32KHz Oscillator Control Register

19.7.1 Register Details

Table 19-5: 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 19-6: RTC Sub-Second Counter Register

RTC Sub-Seconds Counter		**RTC_SSEC**		**[0x0004]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:12	-	RO	0	**Reserved**
11:0	ssec	R/W	0	**Sub-Seconds Counter** *RTC_SEC* increments when this field rolls from 0xFFF to 0x000.

Table 19-7: RTC Time-of-Day Alarm Register

RTC Time-of-Day Alarm		**RTC_TODA**		**[0x0008]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:20	-	RO	0	**Reserved**
19:0	tod_alarm	R/W	0	**Time-of-Day Alarm** This field sets the time-of-day alarm from 1 second up to 12-days. When this field matches *RTC_SEC[19:0]*, an RTC system interrupt is generated.

Table 19-8: 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 19-9: RTC Control Register

RTC Control Register			RTC_CTRL	[0x0010]
Bits	**Field**	**Access**	**Reset**	**Description**
31:16	-	RO	0	**Reserved**
15	wr_en	R/W	0*	**Write Enable** This field controls access to the *RTC_TRIM* register and the RTC enable (*RTC_CTRL.en*) field. 1: Writes to the *RTC_TRIM* register and the *RTC_CTRL.en* field are allowed. 0: Writes to the *RTC_TRIM* register and the *RTC_CTRL.en* field are ignored. **Note: Reset on System Reset, Soft Reset, and GCR_RST0.rtc assertion.*
14	rd_en	R/W	0	**Asynchronous Counter Read Enable** Set this field to 1 to allow direct read access of the *RTC_SEC* and *RTC_SSEC* registers without waiting for *RTC_CTRL.rdy.* Multiple consecutive reads of *RTC_SEC* and *RTC_SSEC* must be executed until two consecutive reads are identical to ensure data accuracy. 0: *RTC_SEC* and *RTC_SSEC* registers are synchronized and should only be accessed while *RTC_CTRL.rdy*= 1. 1: *RTC_SEC* and *RTC_SSEC* registers are asynchronous and require software interaction to ensure data accuracy.
13:11	-	RO	0	**Reserved**
10:9	sqw_sel	R/W	0*	**Frequency Output Select** This field selects the RTC-derived frequency to output on the square wave output pin. See *Table 19-3* for configuration details. 0: 1Hz (Compensated) 1: 512Hz (Compensated) 2: 4kHz 3: Reserved **Note: Reset on POR only.*
8	sqw_en	R/W	0*	**Square Wave Output Enable** This field enables the square wave output. See *Table 19-3* for configuration details. 0: Disabled. 1: Enabled. **Note: Reset on POR only.*
7	ssec_alarm	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 device. 0: No sub-second alarm pending. 1: Sub-second interrupt pending. **Note: Reset on POR only.*
6	tod_alarm	R/W	0*	**Time-of-Day Alarm Interrupt Flag** This interrupt flag is set by the hardware when a time-of-day alarm occurs. 0: No time-of-day alarm interrupt pending. 1: Time-of-day interrupt pending. **Note: Reset on POR only.*
5	rdy_ie	R/W	0*	**RTC Ready Interrupt Enable** 0: Disabled. 1: Enabled. **Note: Reset on system reset, soft reset, and GCR_RST0.rtc assertion.*

RTC Control Register			**RTC_CTRL**	**[0x0010]**
Bits	**Field**	**Access**	**Reset**	**Description**
4	rdy	R/W0O	0*	**RTC Ready** This bit is set to 1 for 120µs by the hardware once a hardware update of the *RTC_SEC* and *RTC_SSEC* registers has occurred. The software should read *RTC_SEC* and *RTC_SSEC* while this hardware bit is set to 1. The software can clear this bit at any time. An RTC interrupt is generated if *RTC_CTRL.rdy_ie* = 1. 0: Software reads of *RTC_SEC* and *RTC_SSEC* are invalid. 1: Software reads of *RTC_SEC* and *RTC_SSEC* are valid. **Note: Reset on System Reset, Soft Reset, and* GCR_RST0.rtc *assertion.*
3	busy	RO	0*	**RTC Busy Flag** This field is set to 1 by the hardware while a register update is in progress. Software writes to the following registers result in this field being set to 1: • *RTC_SEC* • *RTC_SSEC* • *RTC_TRIM* The following fields cannot be written when this field is set to 1: • *RTC_CTRL.en* • *RTC_CTRL.tod_alarm_ie* • *RTC_CTRL.ssec_alarm_ie* • *RTC_CTRL.rdy_ie* • *RTC_CTRL.tod_alarm* • *RTC_CTRL.ssec_alarm* • *RTC_CTRL.sqw_en* • *RTC_CTRL.rd_en* This field is automatically cleared by the hardware when the update is complete. The software should poll this field until it reads 0 after changing the *RTC_SEC*, *RTC_SSEC*, or *RTC_TRIM* register before making any other RTC register modifications. 0: RTC not busy 1: RTC busy **Note: Reset on POR only.*
2	ssec_alarm_ie	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. 0: Disable. 1: Enable. **Note: Reset on POR only.*
1	tod_alarm_ie	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. 0: Disable. 1: Enable. **Note: Reset on POR only.*

RTC Control Register			RTC_CTRL	[0x0010]
Bits	**Field**	**Access**	**Reset**	**Description**
0	en	R/W	0*	**Real-Time Clock Enable** The RTC write enable (*RTC_CTRL.wr_en*) 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. 0: Disabled. 1: Enabled. **Note: Reset on POR only.*

Table 19-10: 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*	**RTC Time Counter** The hardware increments this field every 32 seconds while the RTC is enabled. **Note: Reset on POR only.*
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. **Note: Reset on POR only.*

Table 19-11: RTC 32KHz Oscillator Control Register

RTC Oscillator Control			RTC_OSCCTRL	[0x0018]
Bits	**Field**	**Access**	**Reset**	**Description**
31:6	-	R/W	0	**Reserved**
5	sqw_32k	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 19-3* for configuration details. **Note: Reset on POR only.*
4	bypass	R/W	0	**RTC Crystal Bypass** This field disables the RTC oscillator and allows an external clock source to drive the 32KIN pin. 0: Disable bypass. RTC time base is an external 32kHz crystal. 1: Enable bypass. RTC time base is an external square wave driven on 32KIN. **Note: Reset on POR only.*
3:0	-	DNM	9	**Reserved Do Not Modify**

20. Timers (TMR/LPTMR)

Multiple 32-bit and dual 16-bit, reloadable timers are provided.

The features include:

- Operation as a single 32-bit counter or single/dual 16-bit counter(s).
- Programmable clock prescaler with values from 1 to 4096
- Non-overlapping pulse width modulated (PWM) output generation with configurable off-time.
- Capture, compare, and capture/compare capability.
- Timer input and output signals available and mapped as alternate functions.
    - Refer to the device data sheet for alternate function details and availability
- Configurable input pin for event triggering, clock gating, or capture signal
- Timer output pin for event output and PWM signal generation.
- Multiple clock source options.

Instances denoted as LPTMR, shown in *Table 20-1*, are configurable to operate in any of the low-power modes and wake the device from the low-power modes to *ACTIVE*.

Each timer supports multiple operating modes:

- One-shot: the timer counts up to terminal value then halts.
- Continuous: the timer counts up to the terminal value then repeats.
- Counter: the timer counts input edges received on the timer input pin.
- PWM
- Capture: the timer captures a snapshot of the current timer count when the timer's input edge transitions.
- Compare: the timer pin toggles when the timer's count exceeds the terminal count.
- Gated: the timer increments only when the timer's input pin is asserted.
- Capture/Compare: the timer counts when the timer input pin is asserted; the timer captures the timer's count when the input pin is deasserted.

20.1 Instances

Instances of the peripheral are listed in *Table 20-1*. Both the TMR and LPTMR are functionally similar, so for convenience, all timers are referenced as TMR. The LPTMR instances can function while the device is in certain low-power modes.

Refer to the device data sheet for frequency limitations for external clock sources, if available. Refer to the device data sheet for I/O signal configurations and alternate functions for each timer instance.

Table 20-1: MAX78002 TMR/LPTMR Instances

Instance	Register Access Name	Cascade 32-Bit Mode	16-Bit Mode	Operating Modes	CLK0	CLK1	CLK2	CLK3
TMR0	TMR0	Yes	Dual	*ACTIVE* *SLEEP* *LPM*	PCLK	ISO	IBRO	ERTCO
TMR1	TMR1							
TMR2	TMR2							
TMR3	TMR3							
LPTMR0	TMR4	No	Single	*ACTIVE* *SLEEP* *LPM*	IBRO	ERTCO	INRO	LPTMR0_CLK P2.6 (AF1)
				UPM	N/A	N/A	ERTCO	INRO
LPTMR1	TMR5	No	Single	*ACTIVE* *SLEEP* *LPM*	IBRO	$\frac{IBRO}{8}$	INRO	LPTMR1_CLK P2.7 (AF1)
				UPM	N/A	N/A	ERTCO	INRO

Table 20-2: MAX78002 TMR/LPTMR Instances Capture Events

Instance	Capture Event 0	Capture Event 1	Capture Event 2	Capture Event 3
TMR0	Timer Input Pin	TMR0A_IOA	TMR0B_IOA	Software Event
TMR1	Timer Input Pin	TMR1A_IOA	TMR1B_IOA	Software Event
TMR2	-	-	-	-
TMR3	-	-	-	-
LPTMR0	LPTMR0B_IOA	LPCMP0 Interrupt	LPCMP1 Interrupt	-
LPTMR1	LPTMR1B_IOA	LPCMP0 Interrupt	LPCMP1 Interrupt	-

20.2 Basic Timer Operation

The timer modes operate by incrementing the *TMRn_CNT* register, driven by either the timer clock, an external stimulus on the timer pin, or a combination of both. The *TMRn_CNT* 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 the 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 *TMRn_CNT* with a new starting value, or disabling the counter. The end of a timer period always sets the corresponding interrupt bit and can generate an interrupt if enabled.

In most modes, the timer peripheral automatically sets *TMRn_CNT* to 0x0000 0001 at the end of a timer period, but *TMRn_CNT* is set to 0x0000 0000 following a system reset. This means the first timer period following a system reset is one timer clock longer than subsequent timer periods if *TMRn_CNT* is not initialized to 0x0000 0001 during the timer configuration step.

20.3 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 20-1*. 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 20-3*. Most of the other registers have the same fields duplicated in the upper and lower 16-bits and are differentiated with the _a and _b suffixes.

In the 32-bit modes, the fields and controls associated with TimerA control the 32-bit timer functionality. In single 16-bit timer mode, the TimerA fields control the single 16-bit timer, and the TimerB fields are ignored. In dual 16-bit timer modes, both TimerA and TimerB fields control the dual timers; TimerB fields control the upper 16-bit timer, and TimerA fields control the lower 16-bit timer. In dual-16 bit timer modes, TimerB can be used as a single 16-bit timer.

Table 20-3: TimerA/TimerB 32-Bit Field Allocations

Register	Cascade 32-Bit Mode	Dual 16-Bit Mode		Single 16-Bit Mode
Timer Counter	TimerA Count = *TMRn_CNT[31:0]*	TimerA Compare = *TMRn_CNT[15:0]*	TimerB Count = *TMRn_CNT[31:16]*	TimerA Compare = *TMRn_CNT[15:0]*
Timer Compare	TimerA Compare = *TMRn_CMP[31:0]*	TimerA Compare = *TMRn_CMP[15:0]*	TimerB Compare = *TMRn_CMP[31:16]*	TimerA Compare = *TMRn_CMP[15:0]*
Timer PWM	TimerA Count = *TMRn_PWM.pwm[31:0]*	TimerA Count = *TMRn_PWM.pwm[15:0]*	TimerB Count = *TMRn_PWM.pwm[31:16]*	TimerA Count = *TMRn_PWM.pwm[15:0]*

20.4 Timer Clock Sources

Clocking of timer functions is driven by the timer clock frequency, f_{CNT_CLK}, a function of the selected clock source shown in *Table 20-1*. 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_CTRL0.pres* field.

Equation 20-1: Timer Peripheral Clock Equation

$$f_{CNT\_CLK} = \frac{f_{CLK\_SOURCE}}{prescaler}$$

The software configures and controls the timer by reading and writing to the timer registers. External events on timer pins are asynchronous events to the timer's clock frequency. The external events are latched on the next rising edge of the timer's clock. Since it is not possible to externally synchronize to the timer's internal clock input events may require up to 50% of the timer's internal clock before the hardware recognizes the event.

The software must configure the timer's clock source by performing the following steps:

1. Disable the timer peripheral:
   a. Clear *TMRn_CTRL0.en* to 0 to disable the timer.
   b. Read the *TMRn_CTRL1.clken* field until it returns 0, confirming the timer peripheral is disabled.
2. Set *TMRn_CTRL1.clksel* to the new desired clock source.
3. Configure the timer for the desired operating mode. See *Operating Modes* for details on mode configuration.
4. Enable the timer clock source:
   a. Set the *TMRn_CTRL0.clken* field to 1 to enable the timer's clock source.
   b. Read the *TMRn_CTRL1.clkrdy* field until it returns 1, confirming the timer clock source is enabled.
5. Enable the timer:
   a. Set *TMRn_CTRL0.en* to 1 to enable the timer.
   b. Read the *TMRn_CTRL0.clken* field until it returns 1 to confirm the timer is enabled.

The timer peripheral should be disabled while changing any of the registers in the peripheral.

20.5 Timer Pin Functionality

Each timer instance may have an input signal, an output signal, or both depending on the operating mode. Not all instances of the peripheral are available in all packages. The number of input and output signals per peripheral instance may vary as well. Refer to the data sheet for I/O signal configurations and alternate functions for each timer instance.

The physical pin location of the timer input and output signals may vary between packages. However, the timer functionality is always expressed on the same GPIO pin in the same alternate function mode.

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 has the same electrical characteristics as the GPIO mode settings for the 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 TimerA output controls for modes 0, 1, 3, and 5 output signals are shown in *Figure 20-1*. The TimerA input controls for modes 2, 4, 6, 7, 8, and 14 input signals are shown in *Figure 20-2*.

Figure 20-1: MAX78002 TimerA Output Functionality, Modes 0/1/3/5

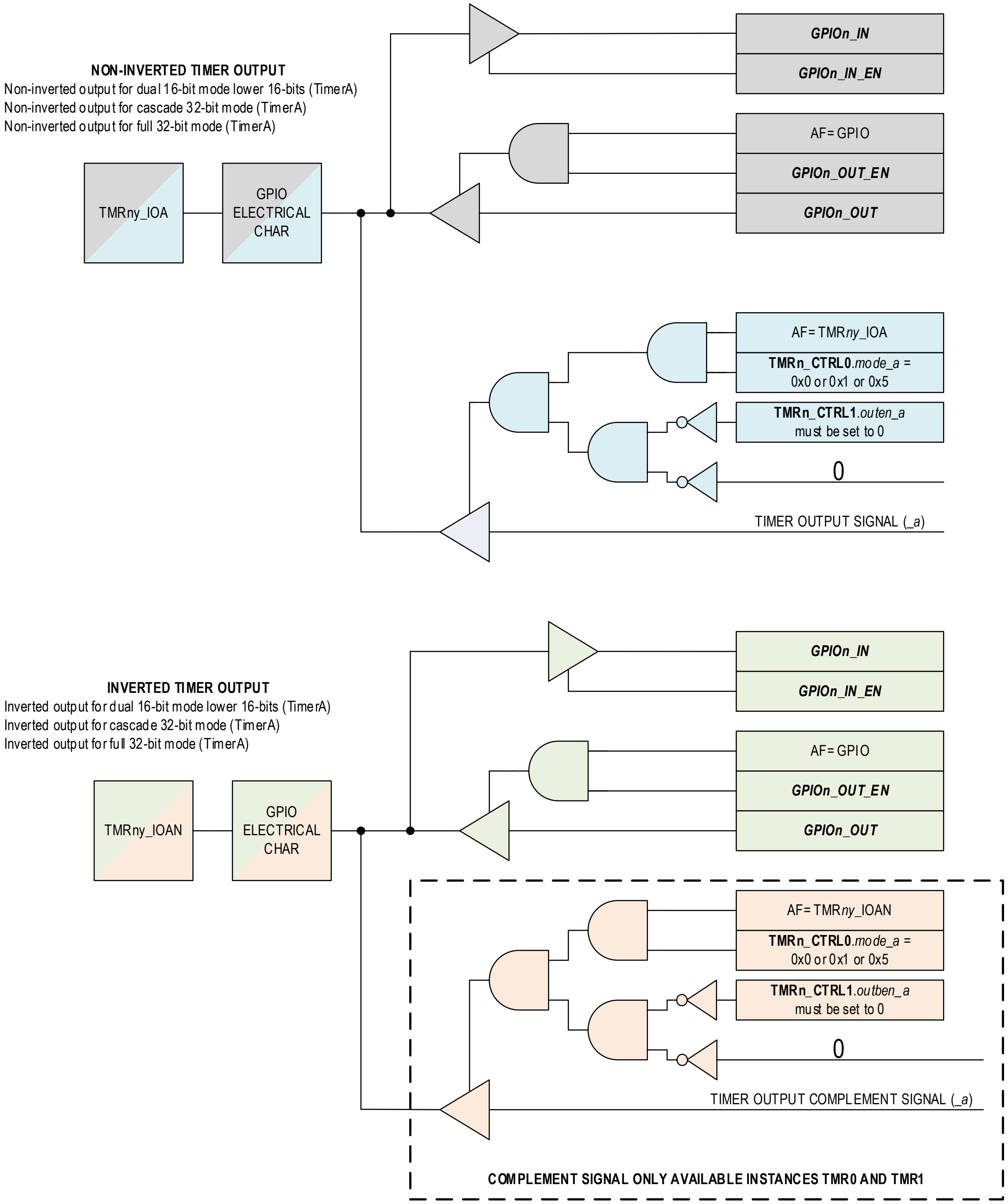
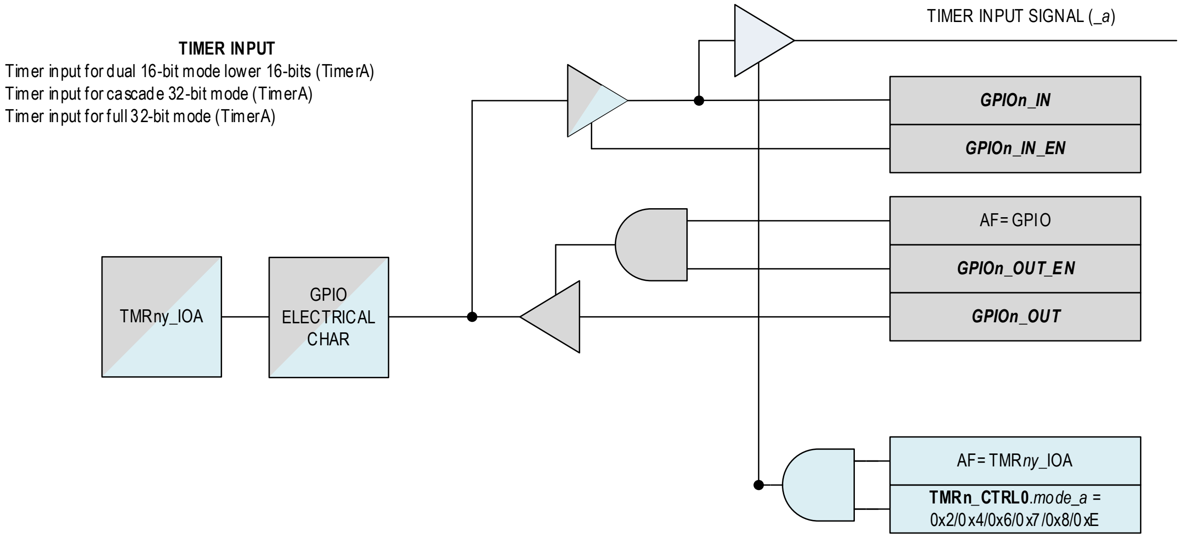

NON-INVERTED TIMER OUTPUT
Non-inverted output for dual 16-bit mode lower 16-bits (TimerA)
Non-inverted output for cascade 32-bit mode (TimerA)
Non-inverted output for full 32-bit mode (TimerA)
TMRny_IOA
GPIO ELECTRICAL CHAR
GPIOn_IN
GPIOn_IN_EN
AF= GPIO
GPIOn_OUT_EN
GPIOn_OUT
AF= TMRny_IOA
TMRn_CTRL0.mode_a = 0x0 or 0x1 or 0x5
TMRn_CTRL1.outen_a must be set to 0
0
TIMER OUTPUT SIGNAL (_a)
INVERTED TIMER OUTPUT
Inverted output for dual 16-bit mode lower 16-bits (TimerA)
Inverted output for cascade 32-bit mode (TimerA)
Inverted output for full 32-bit mode (TimerA)
TMRny_IOAN
GPIO ELECTRICAL CHAR
GPIOn_IN
GPIOn_IN_EN
AF= GPIO
GPIOn_OUT_EN
GPIOn_OUT
AF= TMRny_IOAN
TMRn_CTRL0.mode_a = 0x0 or 0x1 or 0x5
TMRn_CTRL1.outben_a must be set to 0
0
TIMER OUTPUT COMPLEMENT SIGNAL (_a)
COMPLEMENT SIGNAL ONLY AVAILABLE INSTANCES TMR0 AND TMR1

Figure 20-2: MAX78002 TimerA Input Functionality, Modes 2/4/6/7/8/14


TIMER INPUT
Timer input for dual 16-bit mode lower 16-bits (TimerA)
Timer input for cascade 32-bit mode (TimerA)
Timer input for full 32-bit mode (TimerA)
TIMER INPUT SIGNAL (_a)
GPIOn_IN
GPIOn_IN_EN
AF= GPIO
GPIOn_OUT_EN
GPIOn_OUT
TMRny_IOA
GPIO ELECTRICAL CHAR
AF= TMRny_IOA
TMRn_CTRL0.mode_a = 0x2/0x4/0x6/0x7/0x8/0xE

20.6 Wake-Up 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 in *Table 20-1*. In this case, a wake-up event can be configured to wake up the clock control logic and re-enable the system clock.

Programming Sequence Example:

1. Disable the timer peripheral and set the timer clock source as described in *Timer Clock Sources*.
2. Configure the timer operating mode as described in the section *Operating Modes*.
3. Enable the timer by setting *TMRn_CTRL0.en* to 1.
4. Poll *TMRn_CTRL1.clkrdy* until it reads 1.
5. Set the *TMRn_CTRL1.we* field to 1 to enable wake-up events for the timer.
6. If desired, enable the timer interrupt and provide a timer interrupt handler for the timer.
6. Enter a low-power mode as described in the *Operating Modes* section.
8. When the device wakes up from the low-power mode, check the *TMRn_WKFL* register to determine if the timer caused the wake-up event.

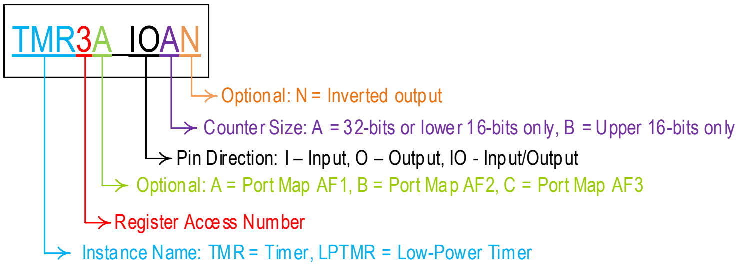
Table 20-4: MAX78002 Wake-Up Events

Condition	Peripheral Wake-Up Flag *TMRn_INTFL*	Peripheral Wake-Up Enable	Low-Power Peripheral Wake-Up Flag	Low-Power Peripheral Wake-Up Enable	Power Management Wake-Up Enable
Any event for LPTMR0	*irq_a*	N/A	*PWRSEQ_LPPWST .lptmr0*	*PWRSEQ_ .lptmr0*	N/A
Any event for LPTMR1	*irq_a*	N/A	*PWRSEQ_LPPWST .lptmr1*	*PWRSEQ_ .lptmr1*	N/A

20.7 Operating Modes

Multiple operating modes are supported. Some operating modes' availability depends on the device and package-specific implementation of the external input and output signals. Refer to the data sheet for I/O signal configurations and alternate functions for each Timer instance.

Figure 20-3: Timer I/O Signal Naming Conventions


TMR3A_IOAN
Optional: N = Inverted output
Counter Size: A = 32-bits or lower 16-bits only, B = Upper 16-bits only
Pin Direction: I – Input, O – Output, IO - Input/Output
Optional: A = Port Map AF1, B = Port Map AF2, C = Port Map AF3
Register Access Number
Instance Name: TMR = Timer, LPTMR = Low-Power Timer

In *Table 20-5*, *Table 20-6*, and *Table 20-7*, the timer's signal name is generically shown where *n* is the timer number (0, 1, 2, 3, etc.) and *y* is the port mapping alternate function. See *Figure 20-3* for details of the timer's naming convention for I/O signals.

Table 20-5: MAX78002 Operating Mode Signals for Timer 0 and Timer 1

Timer Mode	TMR0/TMR1 *TMRn_CTRL1.outen* = 0 *TMRn_CTRL1.outben* = 0	I/O Signal Name†	Pin Required
One-Shot Mode (0)	TimerA Output Signal	TMR*ny*_IOA	Optional
	TimerA Complementary Output Signal	TMR*ny*_IOAN	Optional
	TimerB Output Signal	TMR*ny*_IOB	Optional
	TimerB Complementary Output Signal	TMR*ny*_IOBN	Optional
Continuous Mode (1)	TimerA Output Signal	TMR*ny*_IOA	Optional
	TimerA Complementary Output Signal	TMR*ny*_IOAN	Optional
	TimerB Output Signal	TMR*ny*_IOB	Optional
	TimerB Complementary Output Signal	TMR*ny*_IOBN	Optional
Counter Mode (2)	TimerA Input Signal	TMR*ny*_IOA	Yes
	TimerB Input Signal	TMR*ny*_IOB	Yes
Capture Mode (4)	TimerA Input Signal	TMR*ny*_IOA	Yes
	TimerB Input Signal	TMR*ny*_IOB	Yes
Compare Mode (5)	TimerA Output Signal	TMR*ny*_IOA	Optional
	TimerA Complementary Output Signal	TMR*ny*_IOAN	Optional
	TimerB Output Signal	TMR*ny*_IOB	Optional
	TimerB Complementary Output Signal	TMR*ny*_IOBN	Optional
Gated Mode (6)	TimerA Input Signal	TMR*ny*_IOA	Yes
	TimerB Input Signal	TMR*ny*_IOB	Yes
Capture/Compare Mode (7)	TimerA Input Signal	TMR*ny*_IOA	Yes
	TimerB Input Signal	TMR*ny*_IOB	Yes

Timer Mode	TMR0/TMR1 *TMRn_CTRL1.outen* = 0 *TMRn_CTRL1.outben* = 0	I/O Signal Name†	Pin Required
Dual Edge Capture Mode (8)	TimerA Input Signal	TMR*ny*_IOA	Yes
	TimerB Input Signal	TMR*ny*_IOB	Yes
Reserved (9 - 13)	-	-	-
Inactive Gated Mode (14)	TimerA Input Signal	TMR*ny*_IOA	Yes
	TimerB Input Signal	TMR*ny*_IOB	Yes
Reserved (15)	-	-	-

† *See* Figure 20-3 *for details on the timer I/O signal naming convention and the device data sheet for the alternate functions.*

Table 20-6: MAX78002 Operating Mode Signals for Timer 2 and Timer 3

Timer Mode	TMR2/TMR3 *TMRn_CTRL1.outen_a* = 0 *TMRn_CTRL1.outben_a* = 0	I/O Signal Name†	Required?
One-Shot Mode (0)	TimerA Output Signal	TMR*ny*_IOA	Optional
	TimerB Output Signal	TMR*ny*_IOB	Optional
Continuous Mode (1)	TimerA Output Signal	TMR*ny*_IOA	Optional
	TimerB Output Signal	TMR*ny*_IOB	Optional
Counter Mode (2)	TimerA Input Signal	TMR*ny*_IOA	Yes
	TimerB Input Signal	TMR*ny*_IOB	Yes
Capture Mode (4)	TimerA Input Signal	TMR*ny*_IOA	Yes
	TimerB Input Signal	TMR*ny*_IOB	Yes
Compare Mode (5)	TimerA Output Signal	TMR*ny*_IOA	Optional
	TimerB Output Signal	TMR*ny*_IOB	Optional
Gated Mode (6)	TimerA Input Signal	TMR*ny*_IOA	Yes
	TimerB Input Signal	TMR*ny*_IOB	Yes
Capture/Compare Mode (7)	TimerA Input Signal	TMR*ny*_IOA	Yes
	TimerB Input Signal	TMR*ny*_IOB	Yes
Dual Edge Capture Mode (8)	TimerA Input Signal	TMR*ny*_IOA	Yes
	TimerB Input Signal	TMR*ny*_IOB	Yes
Reserved (0 - 13)	-	-	-
Inactive Gated Mode (14)	TimerA Input Signal	TMR*ny*_IOA	Yes
	TimerB Input Signal	TMR*ny*_IOB	Yes
Reserved (15)	-	-	-

† *See* Figure 20-3 *for details on the timer I/O signal naming convention and the device data sheet for the alternate functions.*

Table 20-7: MAX78002 Operating Mode Signals for Low-Power Timer 0 and Low-Power Timer 1

Timer mode	TMR4/TMR5 *TMRn_CTRL1.outen* = 0 *TMRn_CTRL1.outben* = 0	I/O Signal Name†	Required?
One-Shot Mode (0)	TimerA Output Signal	LPTMR*ny*_IOB	Optional
Continuous Mode (1)	TimerA Output Signal	LPTMR*ny*_IOB	Optional
Counter Mode (2)	TimerA Input Signal	LPTMR*ny*_IOB	Yes
Capture Mode (4)	TimerA Input Signal	LPTMR*ny*_IOB	Yes

Timer mode	**TMR4/TMR5** ***TMRn_CTRL1.outen* = 0** ***TMRn_CTRL1.outben* = 0**	**I/O Signal Name[†]**	**Required?**
Compare Mode (5)	TimerA Output Signal	LPTMR*ny*_IOB	Optional
Gated Mode (6)	TimerA Input Signal	LPTMR*ny*_IOB	Yes
Capture/Compare Mode (7)	TimerA Input Signal	LPTMR*ny*_IOB	Yes
Dual Edge Capture Mode (8)	TimerA Input Signal	LPTMR*ny*_IOB	Yes
Reserved (9 - 13)	-	-	-
Inactive Gated Mode (14)	TimerA Input Signal	LPTMR*ny*_IOB	Yes
Reserved (15)	-	-	-

[†] *See* Figure 20-3 *for details on the timer I/O signal naming convention and the device data sheet for the alternate functions.*

20.7.1 One-Shot Mode (0)

In one-shot mode, the timer peripheral increments the timer's *TMRn_CNT* field until it reaches the timer's *TMRn_CMP* field, and the timer is then disabled. If the timer's output is enabled, the output signal is driven active for one timer clock cycle. One-shot mode provides exactly one timer period and is automatically disabled.

The timer period ends on the timer clock following *TMRn_CNT* = *TMRn_CMP*. The timer peripheral hardware automatically performs the following actions at the end of the timer period:

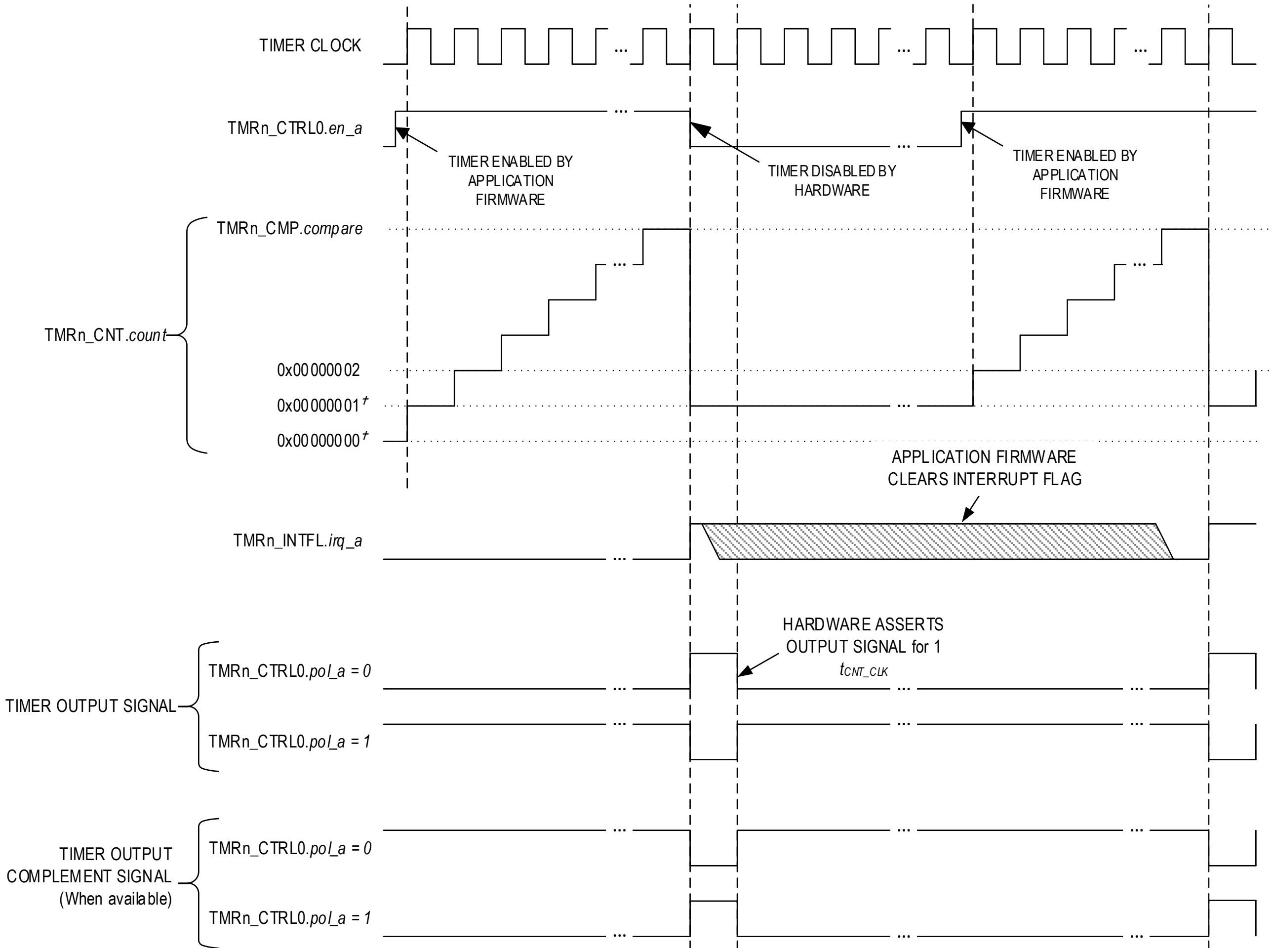
- The *TMRn_CNT* field is set to 0x0000 0001,
- the timer is disabled (*TMRn_CTRL0.en* = 0),
- the timer output, if enabled, is driven to its active state for one timer clock period,
- the *TMRn_INTFL.irq* field is set to 1 to indicate a timer interrupt event occurred.

The timer period is calculated using *Equation 20-2*.

Equation 20-2: One-shot Mode Timer Period

$$One - shot\ mode\ timer\ period\ in\ seconds = \frac{TMRn\_CMP - TMRn\_CNT_{INITIAL\_VALUE} + 1}{f_{CNT\_CLK}(Hz)}$$

Figure 20-4: One-Shot Mode Diagram


TIMER CLOCK
TMRn_CTRL0.*en_a*
TIMER ENABLED BY APPLICATION FIRMWARE
TIMER DISABLED BY HARDWARE
TIMER ENABLED BY APPLICATION FIRMWARE
TMRn_CMP.*compare*
TMRn_CNT.*count*
0x00000002
0x00000001†
0x00000000†
APPLICATION FIRMWARE CLEARS INTERRUPT FLAG
TMRn_INTFL.*irq_a*
HARDWARE ASSERTS OUTPUT SIGNAL for 1 t_{CNT_CLK}
TIMER OUTPUT SIGNAL
TMRn_CTRL0.*pol_a = 0*
TMRn_CTRL0.*pol_a = 1*
TIMER OUTPUT COMPLEMENT SIGNAL (When available)
TMRn_CTRL0.*pol_a = 0*
TMRn_CTRL0.*pol_a = 1*
...

This examples uses the following configuration in addition to the settings shown above:
TMRn_CTRL1.*cascade* = 1 (32-bit Cascade Timer)
TMRn_CTRL0.*mode_a* = 0 (One-shot)

† TMRn_CNT.*count* defaults to 0x00000000 on a timer reset. TMRn_CNT.*count* reloads to 0x00000001 for all following timer periods.

Configure the timer for one-shot mode by performing the following steps:

1. Disable the timer peripheral and set the timer clock source as described in *Timer Clock Sources*.
2. Set the *TMRn_CTRL0*.*mode* field to 0 to select one-shot mode.
3. Set the *TMRn_CTRL0*.*pres* field to set the prescaler for the required timer frequency.
4. 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. Select the correct alternate function mode for the timer output pin.
5. Or, 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. Select the correct alternate function mode for the inverted timer output pin.
6. If using the timer interrupt, enable the corresponding field in the *TMRn_CTRL1* register.
7. Write the compare value to the *TMRn_CMP* field.
8. If desired, write an initial value to the *TMRn_CNT* field.
    a. This affects only the first period; subsequent timer periods always reset the *TMRn_CNT* field to 0x0000 0001.
9. Enable the timer peripheral as described in *Timer Clock Sources*.

20.7.2 Continuous Mode (1)

In continuous mode, the *TMRn_CNT* field increments until it matches the *TMRn_CMP* field; the *TMRn_CNT* field is then set to 0x0000 0001, and the count continues to increment. Optionally, application software can configure continuous mode to toggle the timer output pin at the end of each timer period. A continuous mode timer period ends when the timer count field reaches the timer compare field (*TMRn_CNT* = *TMRn_CMP*).

The timer peripheral hardware automatically performs the following actions on the timer clock cycle after the period ends:

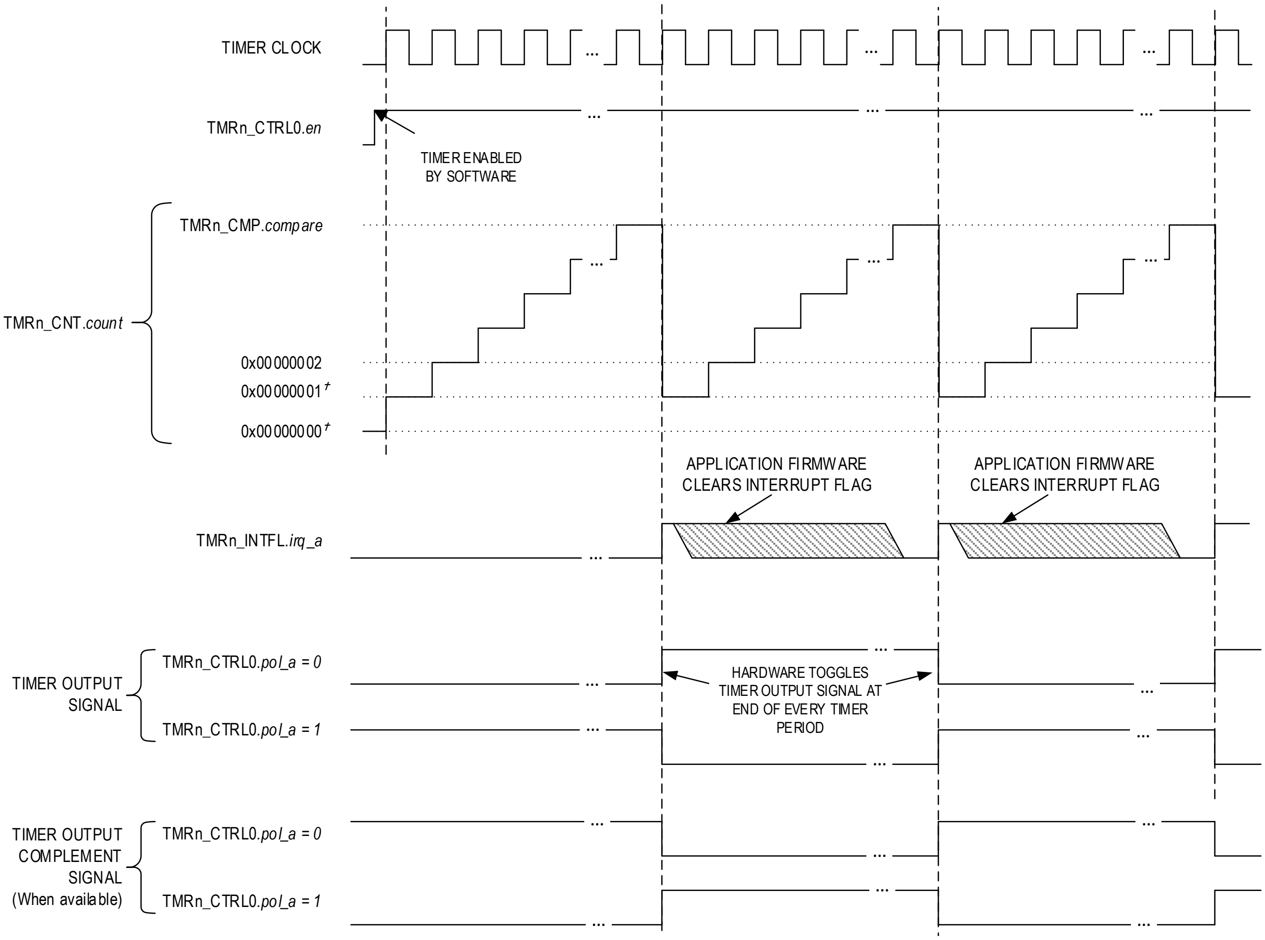
- The *TMRn_CNT* field is set to 0x0000 0001,
- if the timer output signal is toggled,
- the corresponding *TMRn_INTFL*.*irq* field is set to 1 to indicate a timer interrupt event occurred.

The continuous mode timer period is calculated using *Equation 20-3: Continuous Mode Timer Period*.

Equation 20-3: Continuous Mode Timer Period

$$Continuous\ mode\ timer\ period\ (s) = \frac{TMRn\_CMP - TMRn\_CNT_{INITIAL\_VALUE} + 1}{f_{CNT\_CLK}\ (Hz)}$$

Figure 20-5: Continuous Mode Diagram


TIMER CLOCK
TMRn_CTRL0.en
TIMER ENABLED BY SOFTWARE
TMRn_CMP.compare
TMRn_CNT.count
0x00000002
0x00000001†
0x00000000†
APPLICATION FIRMWARE CLEARS INTERRUPT FLAG
APPLICATION FIRMWARE CLEARS INTERRUPT FLAG
TMRn_INTFL.irq_a
TIMER OUTPUT SIGNAL
TMRn_CTRL0.pol_a = 0
TMRn_CTRL0.pol_a = 1
HARDWARE TOGGLES TIMER OUTPUT SIGNAL AT END OF EVERY TIMER PERIOD
TIMER OUTPUT COMPLEMENT SIGNAL (When available)
TMRn_CTRL0.pol_a = 0
TMRn_CTRL0.pol_a = 1

This examples uses the following configuration in addition to the settings shown above:

TMRn_CTRL1.*cascade* = 1 (32-bit Cascade Timer)
TMRn_CTRL0.*mode_a* = 1 (Continuous)

† TMRn_CNT.*count* defaults to 0x00000000 on a timer reset. TMRn_CNT.*count* reloads to 0x00000001 for all following timer periods.

Configure the timer for continuous mode by performing the following steps:

1. Disable the timer peripheral and set the timer clock as described in *Timer Clock Sources*.
2. Set the *TMRn_CTRL0.mode* field to 1 to select continuous mode.
3. Set the *TMRn_CTRL0.pres* field to set the prescaler that determines the timer frequency.
4. 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. Select the correct alternate function mode for the timer output pin.
5. Or, 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. Select the correct alternate function mode for the inverted timer output pin.
6. If using the timer interrupt, enable the corresponding field in the *TMRn_CTRL1* register.
7. Write the compare value to the *TMRn_CMP* field.
8. If desired, write an initial value to the *TMRn_CNT* field.
   a. This affects only the first period; subsequent timer periods always reset the *TMRn_CNT* field to 0x0000 0001.
9. Enable the timer peripheral as described in *Timer Clock Sources*.

20.7.3 Counter Mode (2)

In counter mode, the timer peripheral increments the *TMRn_CNT* each time a transition occurs on the timer input signal. The transition must be greater than $4 \times PCLK$ for a count to occur. When the *TMRn_CNT* reaches the *TMRn_CMP* field, the hardware automatically sets the interrupt bit to 1 (*TMRn_INTFL.irq)*, sets the *TMRn_CNT* field to 0x0000 0001, and continues incrementing. The timer can be configured to increment on either the timer's input signal's rising edge or falling edge, but not both. Use the *TMRn_CTRL0.pol_* field to select which edge is used for the timer's input signal count.

The timer prescaler setting has no effect in this mode. The timer's input signal (f_{CTR_CLK}) frequency must not exceed 25 percent of the PCLK frequency, as shown in *Equation 20-4*.

Note: If the input signal's frequency is equal to f_{PCLK}, it is possible that the timer hardware can miss the transition due to PCLK being an asynchronous internal clock. A minimum of 4 PCLK cycles is required for a count to occur. The timer input signal should be greater than 4 PCLK cycles to guarantee a count occurs.

Equation 20-4: Counter Mode Maximum Clock Frequency

$$f_{CTR\_CLK} \leq \frac{f_{PCLK}\ (Hz)}{4}$$

The timer period ends on the rising edge of PCLK following *TMRn_CNT* = *TMRn_CMP*.

The timer peripheral's hardware automatically performs the following actions at the end of the timer period:

- The *TMRn_CNT* field is set to 0x0000 0001,
- the timer output signal is toggled if the timer output pin is enabled,
- the *TMRn_INTFL.irq* field to 1 indicating a timer interrupt event occurred,
- the timer remains enabled and continues incrementing.

Note: The software must clear the interrupt flag by writing 1 to the TMRn_INTFL.irq field. If the timer period ends and the interrupt flag is already set to 1, a second interrupt does not occur.

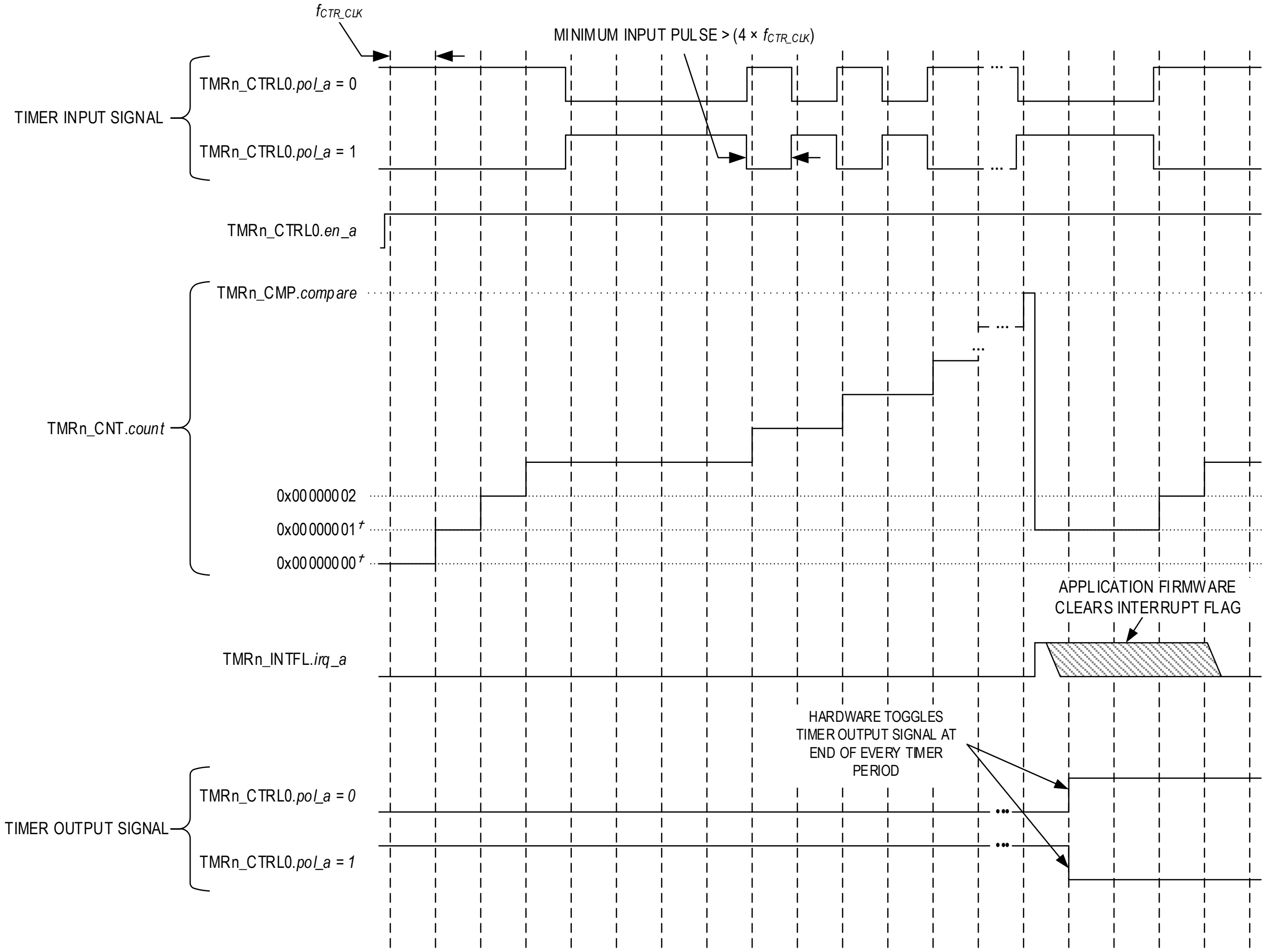
In counter mode, the number of timer input transitions that occurred during a period is equal to the *TMRn_CMP* field's setting. Use *Equation 20-5* to determine the number of transitions that occurred before the end of the timer's period.

Note: Equation 20-5 is only valid during an active timer count before the end of the timer's period.

Equation 20-5: Counter Mode Timer Input Transitions

$$Counter\ mode\ timer\ input\ transitions = TMR\_CNT_{CURRENT\_VALUE}$$

Figure 20-6: Counter Mode Diagram



This examples uses the following configuration in addition to the settings shown above:
TMRn_CTRL1.*cascade* = 1 (32-bit Cascade Timer)
TMRn_CTRL0.*mode_a* = 2 (Counter)

† TMRn_CNT.*count* defaults to 0x00000000 on a timer reset. TMRn_CNT.*count* reloads to 0x00000001 for all following timer periods.

Configure the timer for counter mode by performing 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 2 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_a* 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*.
6. If desired, write an initial value to *TMRn_CNT*. This affects only the first period; subsequent timer periods always reset *TMRn_CNT* = 0x0000 0001.
7. Enable the timer peripheral as described in *Timer Clock Sources*.

20.7.4 PWM Mode (3)

In PWM mode, the timer sends a 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* value matches the *TMRn_PWM*.*pwm*, the timer output signal toggles state. The timer continues counting until it reaches the *TMRn_CMP* value.

The timer period ends on the rising edge of f_{CNT_CLK} following *TMRn_CNT* = *TMRn_CMP*.

The timer peripheral automatically performs the following actions at the end of the timer period:

- The *TMRn_CNT* is reset to 0x0000 0001, and the timer resumes counting,
- the timer output signal is toggled,
- the corresponding *TMRn_INTFL*.*irq* field is 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* value matches the *TMRn_PWM* value. The timer output signal remains high until the *TMRn_CNT* value reaches the *TMRn_CMP*, resulting in the timer output signal transitioning low and the *TMRn_CNT* value resetting to 0x0000 0001.

When *TMRn_CTRL0*.*pol* = 1, the Timer output signal starts high and transitions low when the *TMRn_CNT* value matches the *TMRn_PWM* value. The timer output signal remains low until the *TMRn_CNT* value reaches *TMRn_CMP*, resulting in the timer output signal transitioning high and the *TMRn_CNT* 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 the *TMRn_CTRL0.mode field* to 3 to select PWM mode.
4. Set the *TMRn_CTRL0.pres* field 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* initial value if desired.
    a. The initial *TMRn_CNT* value only affects the initial period in PWM mode, with subsequent periods always setting *TMRn_CNT* to 0x0000 0001.
9. Set the *TMRn_PWM* value to the transition period count.
10. Set the *TMRn_CMP* value for the PWM second transition period. Note: *TMRn_CMP* 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*.

Equation 20-6 shows the formula for calculating the timer PWM period.

Equation 20-6: Timer PWM Period

$$PWM\ period\ (s) = \frac{TMRn\_CNT}{f_{CNT\_CLK}\ (Hz)}$$

If an initial starting value other than 0x0000 0001 is loaded into the *TMRn_CNT* register, use the one-shot mode equation, *Equation 20-2*, 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 20-7*.

Equation 20-7: Timer PWM Output High Time Ratio with Polarity 0

$$PWM\ output\ high\ time\ ratio\ (\%) = \frac{(TMR\_CMP - TMR\_PWM)}{TMR\_CMP} \times 100$$

If *TMRn_CTRL0.pol* is set to 1, the ratio of the PWM output high time to the total period is calculated using *Equation 20-8*.

Equation 20-8: Timer PWM Output High Time Ratio with Polarity 1

$$PWM\ output\ high\ time\ ratio\ (\%) = \frac{TMR\_PWM}{TMR\_CMP} \times 100$$

20.7.5 Capture Mode (4)

Capture mode is used to measure the time between software-determined events. The timer starts incrementing the timer's count field until a transition occurs on the timer's input pin or a rollover event occurs. A capture event is triggered by the hardware when the timer's input pin transitions state. *Equation 20-9* shows the formula for calculating the capture event's elapsed time.

If a capture event does not occur before the timer's count value reaching the timer's compare value (*TMRn_CNT* = *TMRn_CMP*), a rollover event occurs. The capture event and the rollover event set the timer's interrupt flag (*TMRn_INTFL.irq* = 1) resulting in an interrupt if the timer's interrupt is enabled.

A capture event can occur before or after a rollover event. The software must track the number of rollover events that occur before a capture event to determine the elapsed time of the capture event. When a capture event occurs, the software should reset the count of rollover events.

Note: A capture event does not stop the timer's counter from incrementing and does not reset the timer's count value; a rollover event still occurs when the timer's count value reaches the timer's compare value.

20.7.5.1 Capture Event

When a capture event occurs, the timer hardware, on the next timer clock cycle, automatically performs the following actions:

- The *TMRn_CNT* value is copied to the *TMRn_PWM* register,
- the *TMRn_INTFL.irq* field is set to 1,
- the timer remains enabled, and continues counting.

The software must check the value of the TMRn_PWM.pwm field to determine the trigger of the timer interrupt.

Equation 20-9: Capture Mode Elapsed Time Calculation in Seconds

$$Capture\ elapsed\ time\ (s) = \frac{\left(TMR\_PWM - TMR\_CNT_{INITIAL\_VALUE}\right) + \left((Number\ of\ rollover\ events) \times \left(TMR\_CMP - TMR\_CNT_{INITIAL\_VALUE}\right)\right)}{f_{CNT\_CLK}}$$

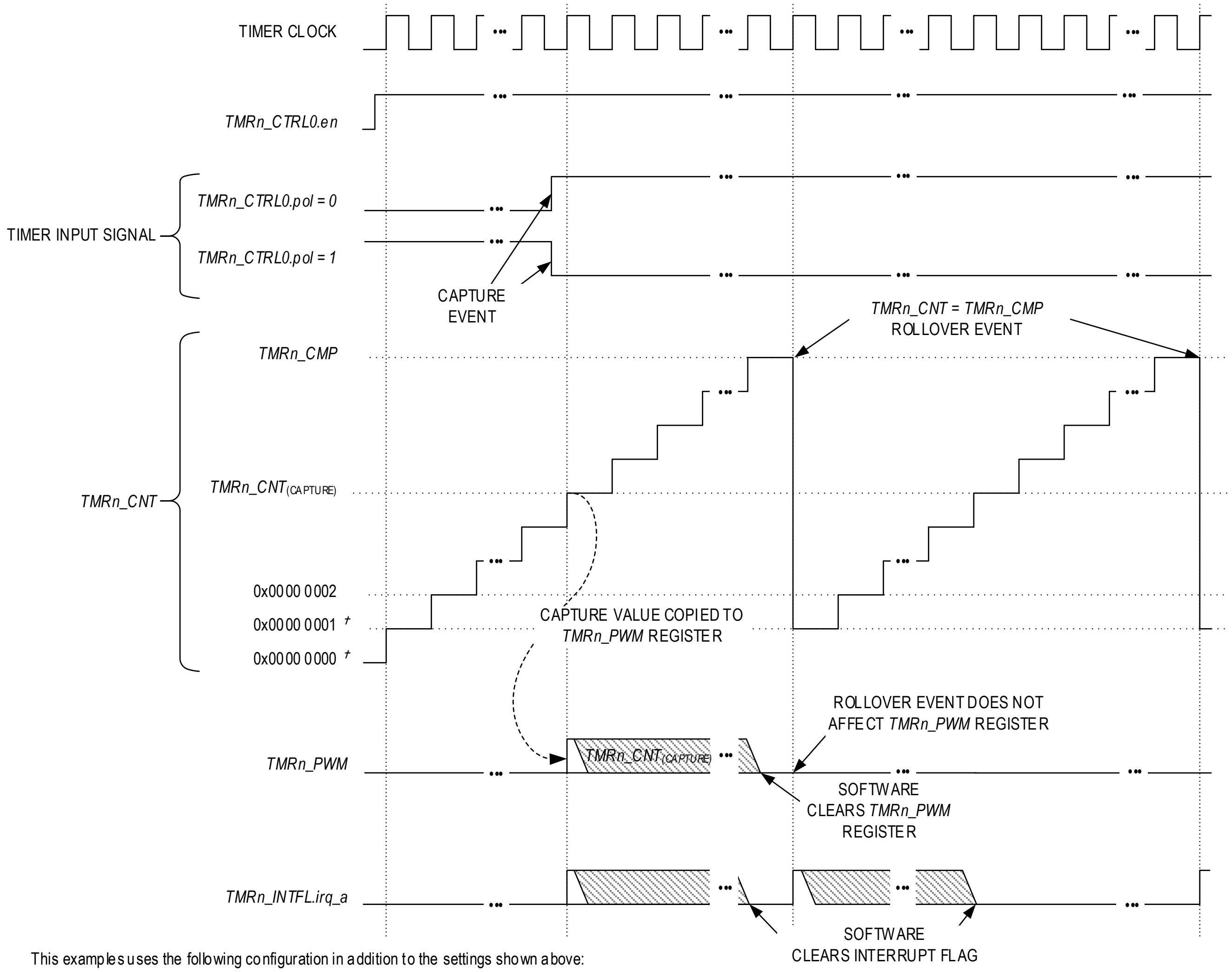
Note: The capture elapsed time calculation is only valid after the capture event occurs, and the timer stores the captured count in the TMRn_PWM register.

20.7.5.2 Rollover Event

A rollover event occurs when the timer's count value reaches the timer's compare value (*TMRn_CNT* = *TMRn_CMP*). A rollover event indicates that a capture event did not occur within the set timer period. When a rollover event occurs, the timer hardware automatically performs the following actions during the next timer clock period:

- The *TMRn_CNT* field is set to 0x0000 0001,
- the *TMRn_INTFL.irq* field is set to 1,
- and the timer remains enabled and continues counting.

Figure 20-7: Capture Mode Diagram



This examples uses the following configuration in addition to the settings shown above:
TMRn_CTRL1.cascade = 1 (32-bit Cascade Timer)
TMRn_CTRL0.mode_a = 4 (Capture)

† *TMRn_CNT* defaults to 0x0000 0000 on a timer reset. *TMRn_CNT* reloads to 0x0000 0001 for all following timer periods.

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 4 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. Select the correct alternate function mode for the timer input pin.
5. Write the initial value to *TMRn_CNT*, if desired.
    a. This affects only the first period; subsequent timer periods always reset *TMRn_CNT* = 0x0000 0001.

6 Write the compare value to the *TMRn_CMP* field.

7. Select the capture event by setting *TMRn_CTRL1.capeventsel*.
8. Enable the timer peripheral as described in *Timer Clock Sources*.

The timer period is calculated using the following equation:

Equation 20-10: Capture Mode Elapsed Time Calculation in Seconds

$$Capture\ elapsed\ time\ in\ seconds = \frac{TMR\_PWM - TMR\_CNT_{INITIAL\_VALUE}}{f_{CNT\_CLK}}$$

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.*

20.7.6 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* = *TMRn_CMP*.

The timer peripheral automatically performs the following actions when a timer period event:

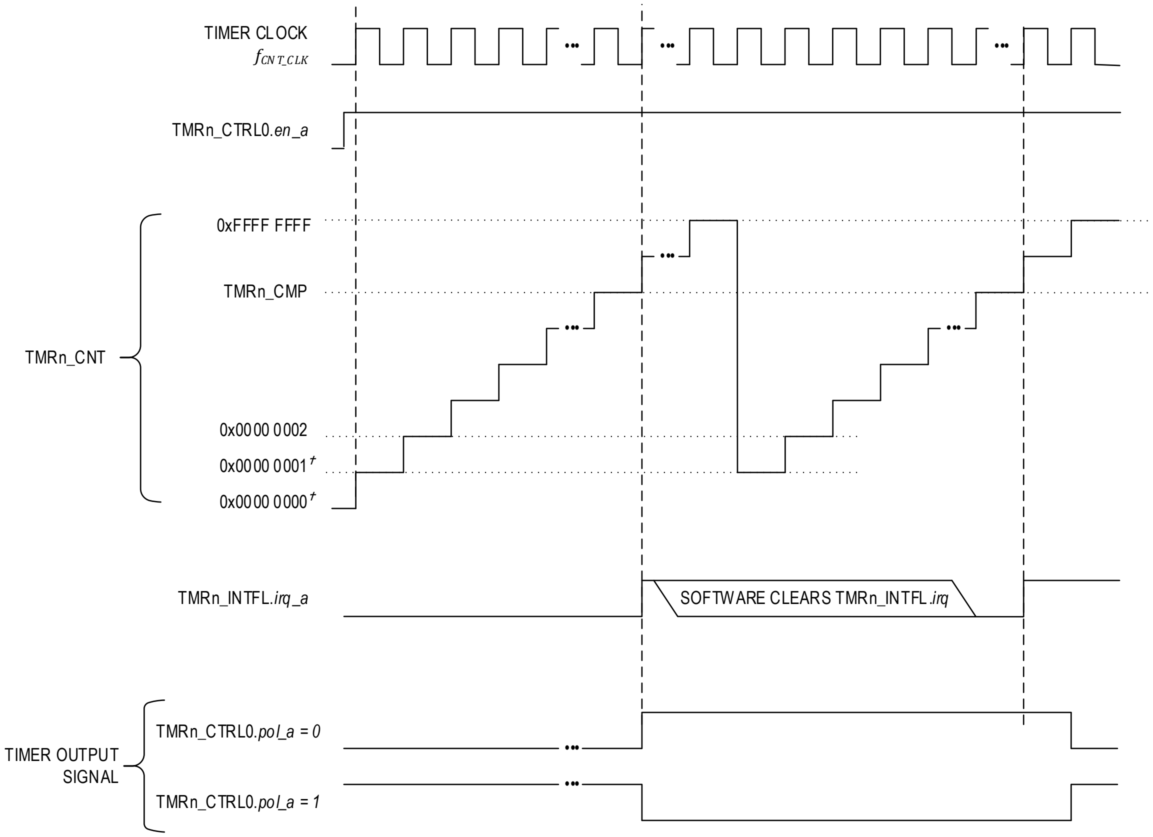
- Unlike other modes, *TMRn_CNT* is reset to 0x0000 00000, not 0x0000 0001 at the end of the timer period.
- The corresponding *TMRn_INTFL.irq* field is set to 1 to indicate a timer interrupt event occurred.
- The hardware toggles the state of the timer output signal. The timer output pin changes state if the timer output is enabled.
- The timer remains enabled and continues incrementing.

The compare Mode timer period is calculated using *Equation 20-12: Capture Mode Elapsed Time*.

Equation 20-11: Compare Mode Timer Period

$$Compare\ mode\ timer\ period\ in\ second = \frac{(TMR\_CMP - TMR\_CNT_{INITIAL\_VALUE} + 1)}{f_{CNT\_CLK}(Hz)}$$

Figure 20-8: Compare Mode Diagram



This examples uses the following configuration in addition to the settings shown above:
TMRn_CTRL1.*cascade* = 1 (32-bit Cascade Timer)
TMRn_CTRL0.*mode_a* = 5 (Compare)

† TMRn_CNT defaults to 0x00000000 on a timer reset. TMRn_CNT reloads to 0x00000001 for all following timer periods.

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 5 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. 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. Select the correct alternate function mode for the inverted timer output pin.
7. If using the timer interrupt, enable the corresponding field in the *TMRn_CTRL1* register.
8. Write the compare value to *TMRn_CMP.*
9. If desired, write an initial value to *TMRn_CNT*.
    a. This affects only the first period; subsequent timer periods always reset *TMRn_CNT* = 0x0000 0001.
10. Enable the timer peripheral as described in *Timer Clock Sources*.

20.7.7 Gated Mode (6)

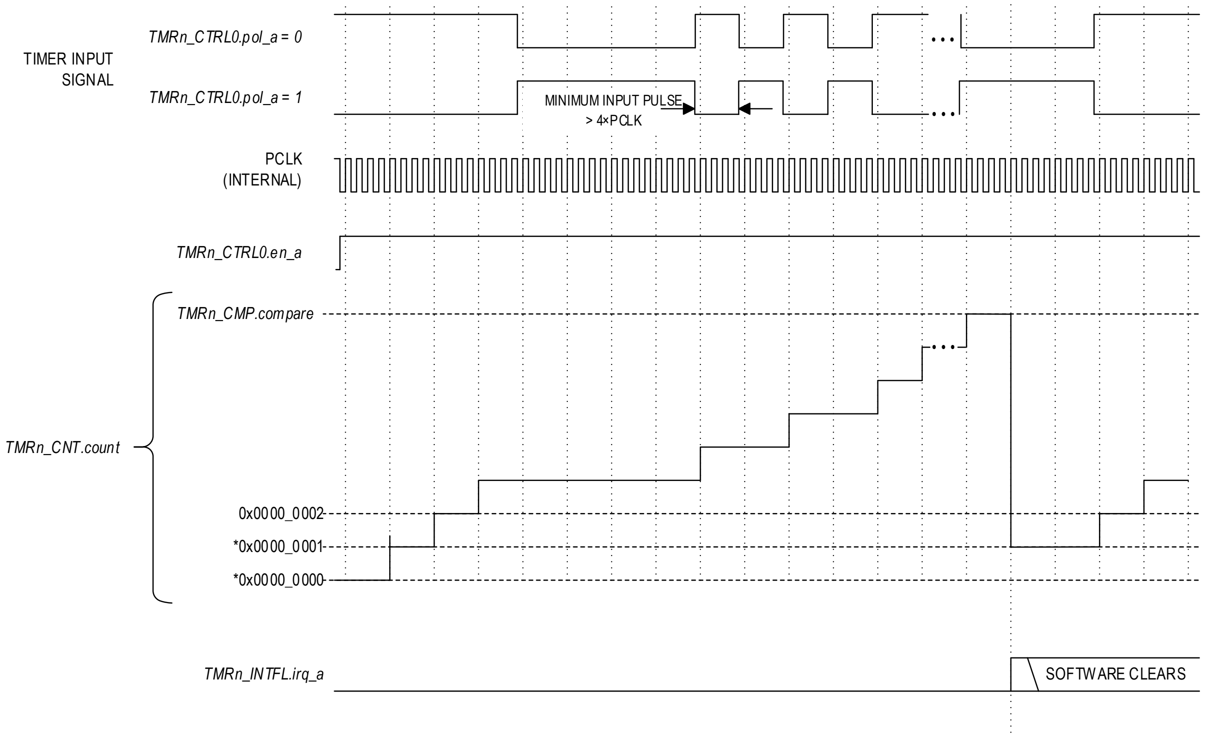
Gated mode is similar to continuous mode, except that *TMRn_CNT* only increments when the timer input signal is in its active state.

The timer period ends on the timer clock following *TMRn_CNT* = *TMRn_CMP*.

The timer peripheral automatically performs the following actions at the end of the timer period:

- The *TMRn_CNT* field is set to 0x0000 0001;
- The timer remains enabled and continues incrementing;
- If the timer output signal toggles state., the timer output pin changes state if the timer output is enabled;
- The corresponding *TMRn_INTFL.irq* field is set to 1 to indicate a timer interrupt event occurred.

Figure 20-9: Gated Mode Diagram



This examples uses the following configuration in addition to the settings shown above:
TMRn_CTRL1.cascade = 1 (32-bit Cascade Timer)
TMRn_CTRL0.mode_a = 6 (Gated)

[†] *TMRn_CNT.count* defaults to 0x00000000 on a timer reset. *TMRn_CNT.count* reloads to 0x00000001 for all following timer periods.

Configure the timer for gated mode by performing the following steps:

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 6 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. Select the correct alternate function mode for the timer input pin.
5. If desired, write an initial value to the *TMRn_CNT* field.
   a. This only effects the first period; subsequent timer periods always reset *TMRn_CNT* = 0x0000 0001.

6 Write the compare value to *TMRn_CMP*.

7. Enable the timer peripheral as described in *Timer Clock Sources*.

20.7.8 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.

After the first transition of the timer input signal, each subsequent transition captures the *TMRn_CNT* value, writing it to the *TMRn_PWM*.*pwm* register (capture event). When a capture event occurs, a timer interrupt is generated, the *TMRn_CNT* value is reset to 0x0000 0001, and the timer resumes counting.

If no capture event occurs, the timer counts up to *TMRn_CMP*. At the end of the cycle, where the *TMRn_CNT* equals the *TMRn_CMP*, a timer interrupt is generated, the *TMRn_CNT* 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 the clock cycle following *TMRn_CNT* = *TMRn_CMP*.

The actions performed at the end of the timer period are dependent on the event that ended the timer period:

If a transition on the timer pin caused the end of the timer period, the hardware automatically performs the following:

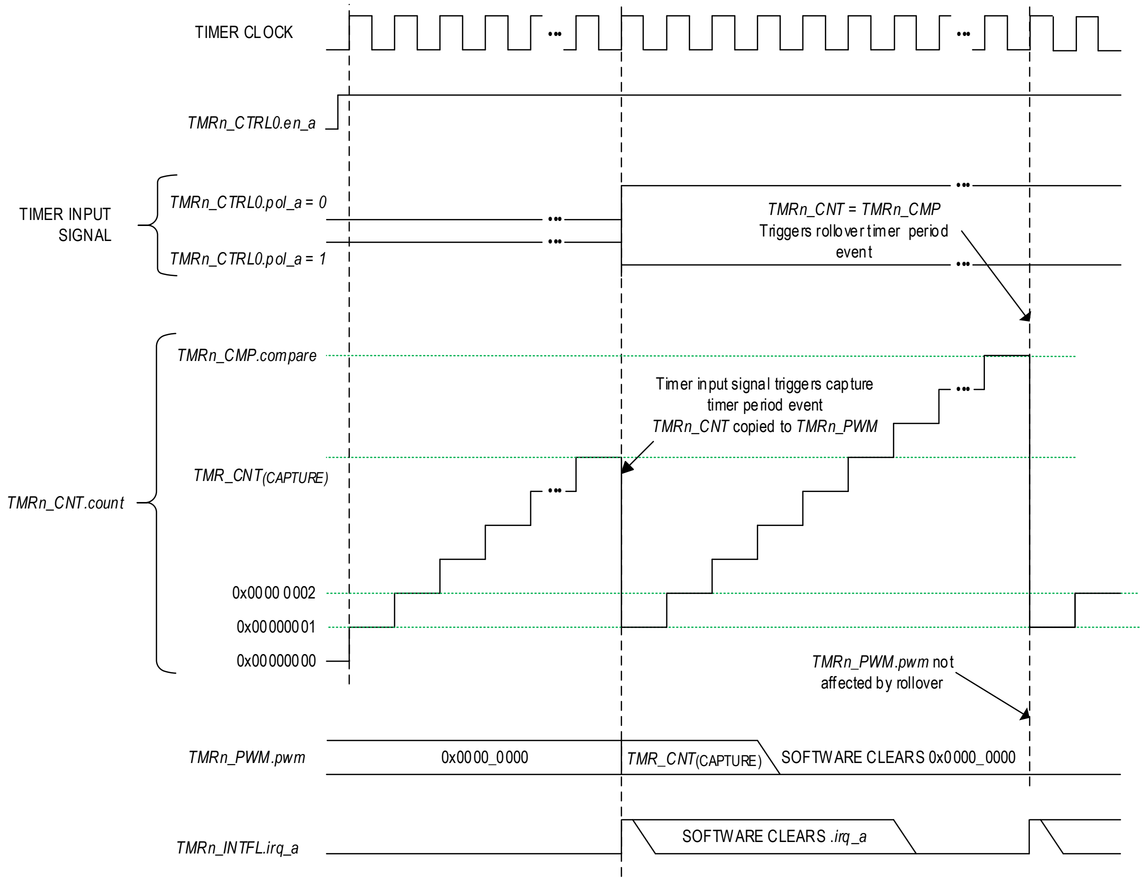
- The value in the *TMRn_CNT* field is copied to the *TMRn_PWM*.*pwm* field,
- the *TMRn_CNT* field is set to 0x0000 0001,
- the timer remains enabled and continues incrementing,
- the corresponding *TMRn_INTFL*.*irq* field is set to 1 to indicate a timer interrupt event occurred.

In capture/compare mode, the elapsed time from the timer start to the capture event is calculated using *Equation 20-12*.

Equation 20-12: Capture Mode Elapsed Time

$$Capture\ elapsed\ time\ (seconds) = \frac{TMRn\_PWM - TMRn\_CNT_{INITIAL\_CNT\_VALUE}}{f_{CNT\_CLK}(Hz)}$$

Figure 20-10: Capture/Compare Mode Diagram



This examples uses the following configuration in addition to the settings shown above:
TMRn_CTRL1.cascade = 1 (32-bit Cascade Timer)
TMRn_CTRL0.mode_a = 7 (Capture/Compare)

† *TMRn_CNT.count* defaults to 0x00000000 on a timer reset. *TMRn_CNT.count* reloads to 0x00000001 for all following timer periods.

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 *TMRn_CTRL0.mode* to 7 to select Capture/Compare mode.
4. Configure the timer input function:
    a. Set *TMRn_CTRL0.pol* to select the positive edge (*TMRn_CTRL0.pol* = 1) or negative edge (*TMRn_CTRL0.pol* = 0) transition to cause the capture event.
    b. Configure the GPIO electrical characteristics as desired.
    c. Select the correct alternate function mode for the timer input pin.
5. If desired, write an initial value to the *TMRn_CNT* field.
    a. This effects only the first period; subsequent timer periods always reset *TMRn_CNT* = 0x0000 0001.

6 Write the compare value to *TMRn_CMP*.

7. Enable the timer peripheral as described in *Timer Clock Sources*.

Note: No interrupt is generated by the first transition of the input signal.

20.7.9 Dual Edge Capture Mode (8)

Dual edge capture mode is similar to capture mode, except the counter can capture on both edges of the timer input pin.

20.7.10 Inactive Gated Mode (14)

Inactive gated mode is similar to gated mode except that the interrupt is triggered when the timer input pin is in its inactive state.

20.8 Registers

See *Table 3-3* 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 20-8*. 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 1-1* 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 20-8: 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_CTRL0*	Timer Control Register
[0x0014]	*TMRn_NOLCMP*	Timer Non-Overlapping Compare Register
[0x0018]	*TMRn_CTRL1*	Timer Configuration Register
[0x001C]	*TMRn_WKFL*	Timer Wake-Up Status Register

20.8.1 Register Details

Table 20-9: 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 is dependent on the 32-bit/16-bit configuration. Reads of this register always return the current value.

Table 20-10: 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 specific mode of the timer determines the compare field meaning. See the timer mode's detailed configuration section for compare usage and meaning.

Table 20-11: Timer PWM Register

Timer PWM			TMRn_PWM	[0x0008]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	pwm	R/W	0	**Timer PWM Match** This field sets the count value for the first transition period of the PWM cycle in PWM mode. At the end of the cycle, when *TMRn_CNT* = *TMRn_CMP*, the PWM output transitions to the second period of the PWM cycle. The second PWM period count is stored in *TMRn_CMP*. *TMRn_PWM.pwm* must be less than *TMRn_CMP* for PWM mode operation. **Timer Capture Value** In capture, compare, and capture/compare modes, this field is used to store the *TMRn_CNT* value when a Capture, Compare, or Capture/Compare event occurs.

Table 20-12: Timer Interrupt Register

Timer Interrupt			TMRn_INTFL	[0x000C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:26	-	RO	0	**Reserved**
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 *TMRn_CNT[31:16]* and *TMRn_PWM.pwm[31:16]*. When this field is set to 0, 32-bit writes to the *TMRn_CNT* and *TMRn_PWM* registers only modify the lower 16-bits associated with TimerA. 0: Enabled 1: Disabled *Note: This field always reads 0 if the timer is configured as a 32-bit cascade timer.*

Timer Interrupt			TMRn_INTFL	[0x000C]
Bits	**Field**	**Access**	**Reset**	**Description**
25	wrdone_b	R	0	**TimerB Write Done** This field is cleared to 0 by the hardware when the software performs a write to *TMRn_CNT[31:16]* or *TMRn_PWM.pwm[31:16]* when in dual timer mode. Wait until the field is set to 1 before proceeding. 0: Operation in progress. 1: Operation complete.
23:17	-	RO	0	**Reserved**
16	irq_b	R/W1C	0	**TimerB Interrupt Event** 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**
9	wr_dis_a	R/W	0	**TimerB Dual Timer Mode Write Protect** This field disables write access to the *TMRn_CNT[31:16]* and *TMRn_PWM.pwm[31:16]* fields so that only the 16 bits associated with updating TimerA are modified during writes to the *TMRn_CNT* and *TMRn_PWM* registers. 0: Enabled 1: Disabled *Note: This field always reads 0 if the timer is configured as a 32-bit cascade timer.*
8	wrdone_a	R	0	**TimerA Write Done** This field is cleared to 0 by the hardware when the application software performs a write to *TMRn_CNT[31:16]* or *TMRn_PWM.pwm[31:16]* when in dual 16-bit timer mode. Wait until the field reads 1 before proceeding. 0: Operation in progress 1: Operation complete
7:1	-	RO	0	**Reserved**
0	irq_a	W1C	0	**TimerA Interrupt Event** This field is set when a TimerA interrupt event occurs. Write 1 to clear. 0: No event 1: Interrupt event occurred

Table 20-13: Timer Control 0 Register

Timer Control 0			TMRn_CTRL0	[0x0010]
Bits	**Field**	**Access**	**Reset**	**Description**
31	en_b	R/W	0	**TimerB Enable** 0: Disabled 1: Enabled
30	clken_b	R/W	0	**TimerB Clock Enable** 0: Disabled 1: Enabled
29	rst_b	R/W1O	0	**TimerB Reset** 0: No action 1: Reset TimerB
28:24	-	RO	0	**Reserved**

Timer Control 0			TMRn_CTRL0	[0x0010]
Bits	**Field**	**Access**	**Reset**	**Description**
23:20	clkdiv_b	R/W	0	**TimerB Prescaler Select** The *clkdiv_b* field selects a prescaler that divides the timer's source clock to set the timer's count clock as follows: $f_{CNT\_CLK} = {f_{CLK\_SOURCE}}/{prescaler}$ See *Operating Modes* for details on which timer 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	**TimerB Mode Select** Set this field to the desired 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 12: Internally Gated 13-15: Reserved
15	en_a	R/W	0	**TimerA Enable** 0: Disabled 1: Enabled
14	clken_a	R/W	0	**TimerA Clock Enable** 0: Disabled 1: Enabled
13	rst_a	R/W1O	0	**TimerA Reset** 0: No action 1: Reset TimerA

Timer Control 0			TMRn_CTRL0	[0x0010]
Bits	**Field**	**Access**	**Reset**	**Description**
12	pwmckbd_a	R/W	1	**TimerA PWM Output $\phi A'$ Disable** Set this field to 0 to enable the $\phi A'$ output signal. The $\phi A'$ output signal is disabled by default. 0: Enable the PWM $\phi A'$ output signal. 1: Disable PWM $\phi A'$ output signal.
11	nollpol_a	R/W	0	**TimerA PWM Output $\phi A'$ Polarity Bit** Set this field to 1 to invert the PWM $\phi A'$ signal. 0: Do not invert the PWM $\phi A'$ output signal. 1: Invert the PWM $\phi A'$ output signal.
10	nolhpol_a	R/W	0	**TimerA PWM Output ϕA Polarity Bit** Set this field to 1 to invert the PWM ϕA signal. 0: Do not invert the ϕA PWM output signal. 1: Invert the ϕA output signal.
9	pwmsync_a	R/W	0	**TimerA/TimerB PWM Synchronization Mode** 0: Disabled 1: Enabled
8	pol_a	R/W	0	**TimerA Polarity** This field selects the polarity of the timer's input and output signal. This setting is not used if the GPIO is not configured for the timer's alternate function. This field's usage and settings are operating mode specific. See the *Operating Modes* section for details on the mode selected.
7:4	clkdiv_a	R/W	0	**TimerA Prescaler Select** The *clkdiv_a* field selects a prescaler that divides the timer's clock source to set the timer's count clock as follows: $f_{CNT\_CLK} = {f_{CLK\_SOURCE}}/{prescaler}$ See the *Operating Modes* section 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

Timer Control 0		TMRn_CTRL0		[0x0010]
Bits	**Field**	**Access**	**Reset**	**Description**
3:0	mode_a	R/W	0	**TimerA Mode Select** Set this field to the desired operating 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. 12: Internally Gated 13-15: Reserved.

Table 20-14: Timer Non-Overlapping Compare Register

Timer Non-Overlapping Compare		TMRn_NOLCMP		[0x0014]
Bits	**Field**	**Access**	**Reset**	**Description**
31:24	hi_b	R/W	0	**TimerA Non-Overlapping High Compare 1** The 8-bit timer count value of non-overlapping time between the falling edge of the PWM output $\phi A'$ (phase A prime) and the next rising edge of the PWM output $\phi A\ (phase\ A)$.
23:16	lo_b	R/W	0	**TimerA Non-Overlapping Low Compare 1** The 8-bit timer count value of non-overlapping time between the falling edge of the PWM output ϕA and the next rising edge of the PWM output $\phi A'$.
15:8	hi_a	R/W	0	**TimerA Non-Overlapping High Compare 0** The 8-bit timer count value of non-overlapping time between the falling edge of the PWM output $\phi A'$ and the next rising edge of the PWM output ϕA.
7:0	lo_a	R/W	0	**TimerA Non-Overlapping Low Compare 0** The 8-bit timer count value of non-overlapping time between the falling edge of the PWM output ϕA and the next rising edge of the PWM output $\phi A'$.

Table 20-15: Timer Control 1 Register

Timer Control 1		TMRn_CTRL1		[0x0018]
Bits	**Field**	**Access**	**Reset**	**Description**
31	cascade	R/W	0	**32-bit Cascade Timer Enable** This field is only supported by timer instances with support for 32-bit cascade mode. 0: Dual 16-bit timers 1: 32-bit cascade timer
30:29	-	RO	0	**Reserved**

Timer Control 1			TMRn_CTRL1	[0x0018]
Bits	**Field**	**Access**	**Reset**	**Description**
28	we_b	R/W	0	**TimerB Wake-Up Function** 0: Disabled 1: Enabled
27	sw_capevent_b	R/W	0	**TimerB Software Event Capture** Write this field to 1 to initiate a software event capture when operating the timer in capture mode to perform a software event capture. 0: No event 1: Reserved
26:25	capevent_sel_b	R/W	0	**TimerB Event Capture Selection** Set this field to the desired capture event source. See *Table 20-2* for available capture event 0 and capture event 1 options. 0-3: Reserved
24	ie_b	R/W	0	**TimerB Interrupt Enable** 0: Disabled 1: Enabled
23	negtrig_b	R/W	0	**TimerB Negative Edge Trigger for Event** 0: Rising-edge trigger 1: Falling-edge trigger
22:20	event_sel_b	R/W	0	**TimerB Event Selection** 0: Event disabled 1-7: Reserved
19	clkrdy_b	RO	0	**TimerB Clock Ready Status** This field indicates if the timer clock is ready. 0: Timer clock not ready or synchronization in progress 1: Timer clock is ready
18	clken_b	RO	0	**TimerB Clock Enable Status** Set this field to 1 to enable the TimerB clock. 0: Timer not enabled or synchronization in progress 1: Timer is enabled
17:16	clksel_b	R/W	0	**TimerB Clock Source** See *Table 20-1* for the clock sources supported by each instance. 0: Clock option 0. 1: Clock option 1. 2: Clock option 2. 3: Clock option 3.
15	-	RO	0	**Reserved**
14	outben_a	RO	0	**Output B Enable** Reserved.
13	outen_a	RO	0	**Output Enable** Reserved.
12	we_a	R/W	0	**TimerA Wake-Up Function** 0: Disabled 1: Enabled.

Timer Control 1				TMRn_CTRL1	[0x0018]
Bits	**Field**	**Access**	**Reset**	**Description**	
11	sw_capevent_a	R/W	0	**TimerA Software Event capture** 0: No software capture event triggered 1: Trigger software capture event	
10:9	capevent_sel_a	R/W	0	**TimerA Event capture Selection** Set this field to the desired capture event source. See *Table 20-2* for available capture event 0 and capture event 1 options. 0: Capture event 0 1: Capture event 1 2: Capture event 2 3: Capture event 3	
8	ie_a	R/W	0	**TimerA Interrupt Enable** 0: Disabled 1: Enabled	
7	negtrig_a	R/W	0	**TimerA Edge Trigger Selection for Event** 0: Positive-edge triggered 1: Negative-edge triggered	
6:4	event_sel_a	R/W	0	**TimerA Event Selection** 0: Event disabled 1-7: Reserved	
3	clkrdy_a	RO	0	**TimerA Clock Ready** This field is set to 1 after software enables the TimerA clock by writing 1 to the 0: Timer not enabled or synchronization in progress 1: TimerA clock is ready	
2	clken_a	R/W	0	**TimerA Clock Enable** Write this field to 1 to enable the TimerA clock. 0: Timer not enabled or synchronization in progress 1: Timer is enabled	
1:0	clksel_a	R/W	0	**Clock Source TimerA** See *Table 20-1* for the available clock options for each timer instance. 0: Clock option 0 1: Clock option 1 2: Clock option 2 3: Clock option 3	

Table 20-16: Timer Wake-Up Status Register

Timer Wake-Up Status				TMRn_WKFL	[0x001C]
Bits	**Field**	**Access**	**Reset**	**Description**	
31:17	-	RO	0	**Reserved**	
16	b	R/W1C	1	**TimerB Wake-Up Event** This flag is set when a wake-up event occurs for TimerB. Write 1 to clear. 0: No event 1: Wake-up event occurred	
15:1	-	RO	0	**Reserved**	

Timer Wake-Up Status		TMRn_WKFL		[0x001C]
Bits	**Field**	**Access**	**Reset**	**Description**
0	a	R/W1C	1	**TimerA Wake-Up Event** This flag is set when a wake-up event occurs for TimerA. Write 1 to clear. 0: No event 1: Wake-up event occurred

21. Wake-Up Timer (WUT)

The WUT is a unique instance of a 32-bit timer.

- The wake-up timer uses the ERTCO for its clock source.
- Programmable prescaler with values from 1 to 4096.
- Supports three timer modes, all of which can wake the device from low-power modes:
    - One-Shot: The timer counts up to the terminal value, generates a wake-up timer event then halts.
    - Continuous: The timer counts up to the terminal value, generates a wake-up timer event then continues counting.
    - Compare: The timer counts up to the terminal value, generates a wake-up timer event, resets the count and continues counting.
- Independent interrupt handler (WUT_IRQn).

21.1 Instances

There is one instance of the WUT peripheral in the MAX78002.

21.2 Basic Operation

The timer modes operate by incrementing the *WUT_CNT* register. The *WUT_CNT* 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 always sets the corresponding interrupt flag and generates a wake-up timer interrupt (WUT_IRQn), if enabled.

The timer clock frequency, f_{CNT_CLK}. is a divided version of the 32.768kHz RTC clock as shown in *Equation 21-1*.

Equation 21-1: Wake-Up Timer Clock Frequency

$$f_{CNT\_CLK} = \frac{f_{RTC\_CLK}}{prescaler}$$

The divisor (prescaler) can be set from 1 to 4096 using the concatenated fields *WUT_CTRL.pres3*:*WUT_CTRL.pres* as shown in *Table 21-1*.

Table 21-1: MAX78002 WUT Clock Period

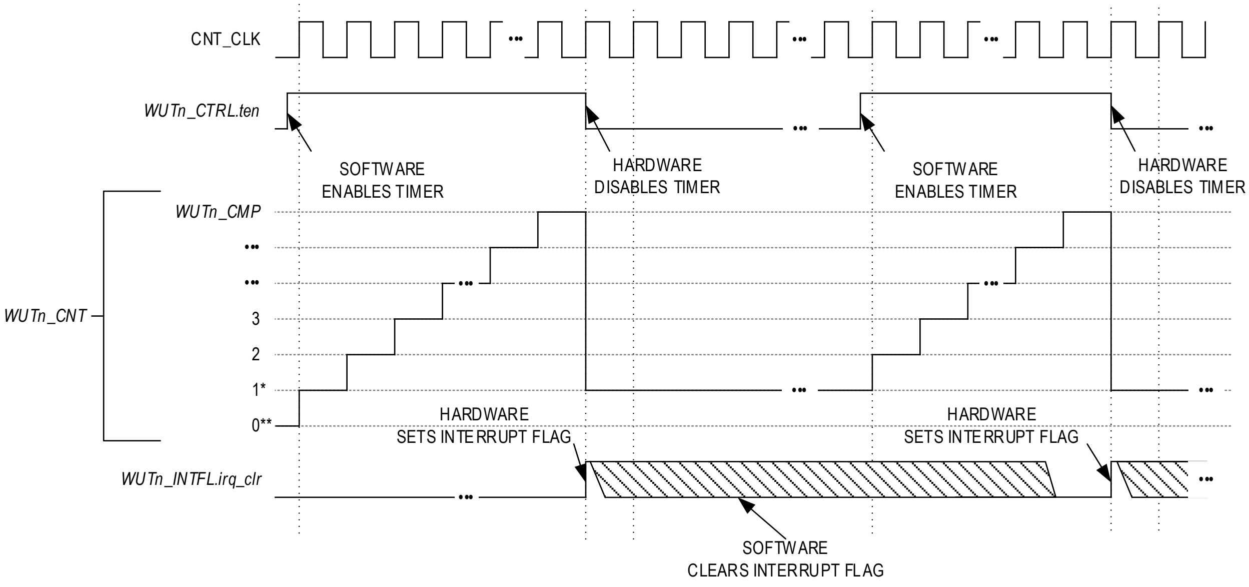
WUT_CTRL.pres3	***WUT_CTRL.pres***	**Prescaler**	**f_{CNT_CLK} (Hz)**
0	0b000	1	32,768
0	0b001	2	16,384
0	0b010	4	8,192
0	0b011	8	4,096
0	0b100	16	2,048
0	0b101	32	1,024
0	0b110	64	512
0	0b111	128	256
1	0b000	256	128
1	0b010	512	64
1	0b011	1024	32
1	0b100	2048	16

WUT_CTRL.pres3	*WUT_CTRL.pres*	Prescaler	f_{CNT_CLK} (Hz)
1	0b101	4096	8
1	0b110	Reserved	Reserved
1	0b111	Reserved	Reserved

21.3 One-Shot Mode (0)

In one-shot mode, the timer peripheral increments the *WUT_CNT* register until it matches the *WUT_CMP* register, generates a wake-up event, stops incrementing, and disables the timer. In this mode, the timer must be re-enabled to start another one-shot mode event.

Figure 21-1: One-Shot Mode Diagram



* *WUTn_CNT* automatically reloads with 1 at the end of the WUT period, but software can write any initial value to *WUTn_CNT* prior to enabling the timer.

** The default value of *WUTn_CNT* for the first period after a system reset is 0 unless changed by software.

21.3.1 One-Shot Mode Timer Period

The timer period ends on the timer clock when *WUT_CNT* = *WUT_CMP*.

The timer peripheral automatically performs the following actions at the end of the timer period:

1. *WUT_CNT* is reset to 1.
2. The timer is disabled by setting *WUT_CTRL.ten* = 0.
3. The timer interrupt bit *WUT_INTFL.irq_clr* is set and wakes up the device if the wake-up timer is enabled as a wake-up event, generating an interrupt.

21.3.2 One-Shot Mode Configuration

Configure the timer for one-shot mode by performing the following steps:

1. Set *WUT_CTRL.ten* = 0 to disable the timer.
2. Set *WUT_CTRL.tmode* to 0 to select one-shot mode.
3. Set *WUT_CTRL.pres3*:*WUT_CTRL.pres* to determine the timer period.
4. If desired, register a wake-up interrupt handler (WUT_IRQn).
5. Write an initial value to the *WUT_CNT* register, if desired. This effects only the first period; subsequent timer periods always reset the *WUT_CNT* register to 1.
6. Write the compare value to the *WUT_CMP* register.
7. Clear the wake-up timer interrupt flag by writing 0 to *WUT_INTFL.irq_clr*.
8. Set *WUT_CTRL.ten* to 1 to enable the timer.
9. Enter a low-power sleep mode. See *Operating Modes* for details.

The timer period is calculated using the following equation:

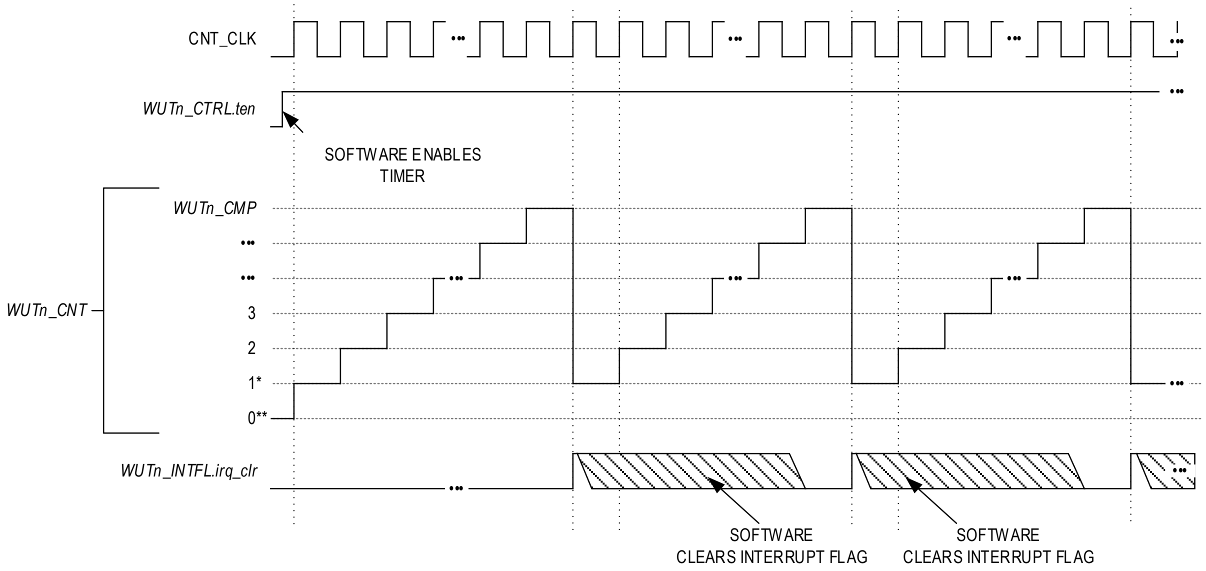
Equation 21-2: One-Shot Mode Timer Period

$$One\text{-}Shot\ mode\ timer\ period\ in\ seconds = \frac{WUTn\_CMP - WUTn\_CNT_{INITIAL\_VALUE} + 1}{f_{CNT\_CLK}\ (Hz)}$$

21.4 Continuous Mode (1)

In continuous mode, the timer peripheral increments the *WUT_CNT* register until it matches the *WUT_CMP* register, generates a wake-up event, the hardware resets the *WUT_CNT* register to 1, and continues incrementing.

Figure 21-2: Continuous Mode Diagram



* *WUTn_CNT* automatically reloads with 1 at the end of the wakeup timer period, but software can write any initial value to *WUTn_CNT* prior to enabling the wakeup timer.

** The value of *WUTn_CNT* for the first period after a system reset is 0 unless changed by software.

21.4.1 Continuous Mode Timer Period

The wake-up timer period ends on the timer clock following *WUT_CNT* = *WUT_CMP*.

The wake-up timer peripheral automatically performs the following actions at the end of the timer period:

1. *WUT_CNT* is reset to 1. The wake-up timer remains enabled and continues incrementing.
2. The timer interrupt bit *WUT_INTFL.irq_clr* is set. An interrupt is generated if enabled.

21.4.2 Continuous Mode Configuration

Configure the timer for continuous mode by performing the steps following:

1. Set *WUT_CTRL.ten* = 0 to disable the timer.
2. Set *WUT_CTRL.tmode* to 1 to select continuous mode.
3. Set *WUT_CTRL.pres3*:*WUT_CTRL.pres* to determine the timer period.
4. If desired, register a wake-up interrupt handler (WUT_IRQn).
5. Write an initial value to the *WUT_CNT* register, if desired. The initial value is only used for the first period; subsequent timer periods always reset the *WUT_CNT* register to 1.
6. Write the compare value to the *WUT_CMP* register.
7. Clear the wake-up timer interrupt flag by writing 0 to *WUT_INTFL.irq_clr*.
8. Set *WUT_CTRL.ten* to 1 to enable the timer.
9. Enter a low-power sleep mode. See *Operating Modes* for details.

The continuous mode timer period is calculated using *Equation 21-3*.

Equation 21-3: Continuous Mode Timer Period

$$Continuous\ Mode\ Timer\ Period\ in\ seconds = \frac{WUTn\_CMP - WUTn\_CNT_{INITIAL\_VALUE} + 1}{f_{CNT\_CLK}\ (Hz)}$$

21.4.3 Compare Mode (5)

In compare mode, the timer peripheral increments continually from 0x0000 0000 (after the first timer period) to the maximum value, 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 *WUT_CNT* = *WUT_CMP*.

The timer peripheral automatically performs the following actions when a timer period event ends:

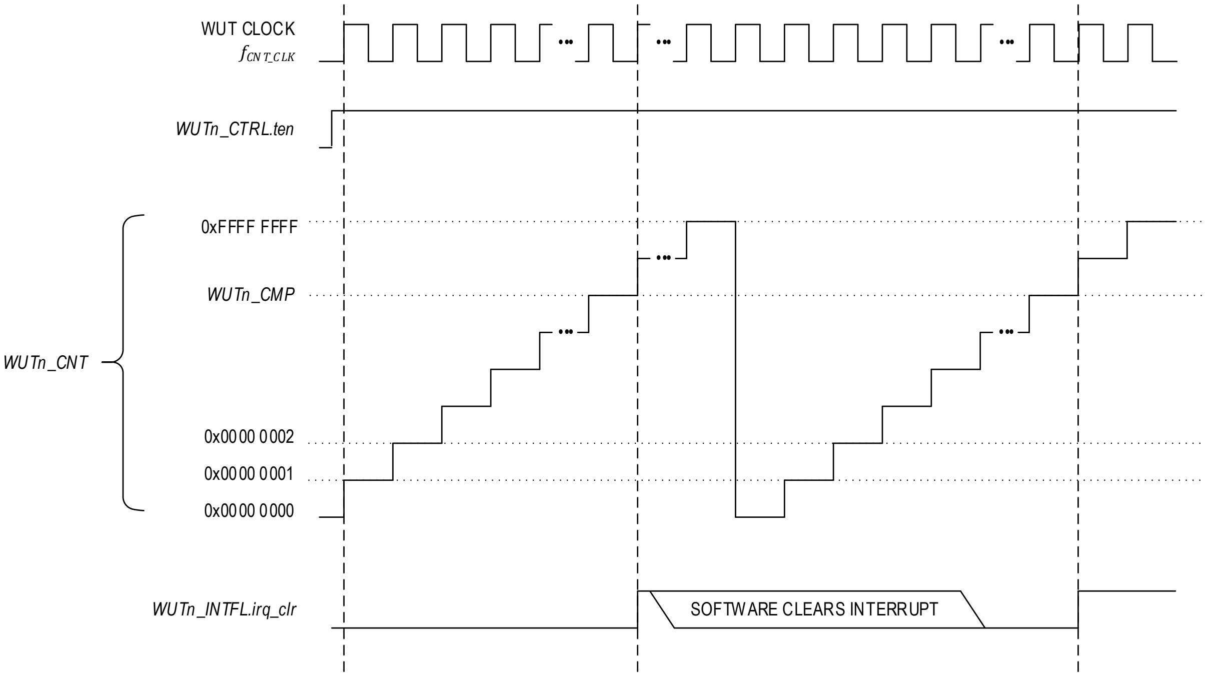
- *WUT_CNT* is reset to 0x0000 00000.
- The *WUT_INTFL.irq_clr* field is set to 1 to indicate a timer interrupt event occurred.
- The timer remains enabled and continues incrementing.

The initial compare mode timer period is calculated using *Equation 21-4*. Subsequent compare mode timer periods are always 0xFFFF FFFF.

Equation 21-4: Compare Mode Timer Initial Period

$$Compare\ mode\ timer\ period\ in\ seconds = \frac{(WUT\_CMP - WUT\_CNT_{INITIAL\_VALUE} + 1)}{f_{CNT\_CLK}(Hz)}$$

Figure 21-3: Compare Mode Diagram



This examples uses the following configuration in addition to the settings shown above:
WUTn_CTRL.tmode = 5 (Compare)

Configure the timer for compare mode by doing the following:

1. Set *WUT_CTRL.ten* = 0 to disable the timer.
2. Set *WUT_CTRL.tmode* to 1 to select continuous mode.
3. Set *WUT_CTRL.pres3*:*WUT_CTRL.pres* to determine the timer period.
4. If desired, register a wake-up interrupt handler (WUT_IRQn).
5. Write the compare value to the *WUT_CMP* register.
6. If desired, write an initial value to *WUT_CNT* register.
7. Clear the wake-up timer interrupt flag by writing 0 to *WUT_INTFL.irq_clr*.
8. Set *WUT_CTRL.ten* to 1 to enable the timer.
9. Enter a low-power sleep mode. See *Operating Modes* for details.

21.5 Registers

See *Table 3-3* 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 21-2*. 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 1-1* 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 21-2: Wake-Up Timer Register Summary

Offset	Register Name	Description
[0x0000]	*WUT_CNT*	Wake-up Timer Counter Register
[0x0004]	*WUT_CMP*	Wake-up Timer Compare Register
[0x0008]	*WUT_PWM*	Wake-up Timer PWM Register
[0x000C]	*WUT_INTFL*	Wake-up Timer Interrupt Register
[0x0010]	*WUT_CTRL*	Wake-up Timer Control Register
[0x0014]	*WUT_NOLCMP*	Wake-up Timer Non-Overlapping Compare Register

21.5.1 Register Details

Table 21-3: Wake-Up Timer Count Register

Wake-Up Timer Count			**WUT_CNT**	**[0x0000]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	-	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. Before writing this field, disable the timer by clearing the bit *WUT_CTRL.ten*.

Table 21-4: Wake-Up Timer Compare Register

Wake-Up Timer Compare			**WUT_CMP**	**[0x0004]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	-	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 21-5: Wake-Up Timer PWM Register

Wake-Up Timer PWM			**WUT_PWM**	**[0x0008]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	-	RO	0	**Reserved**

Table 21-6: Wake-Up Timer Interrupt Register

Wake-Up Timer Interrupt			**WUT_INTFL**	**[0x000C]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**
0	irq_clr	R/W	0	**Timer Interrupt Flag** If set, this field indicates a wake-up timer interrupt condition occurred. Writing any value to this bit clears the wake-up timer's interrupt. 0: Normal operation. 1: Wake-up timer interrupt occurred.

Table 21-7: Wake-Up Timer Control Register

Wake-Up Timer Control			**WUT_CTRL**	**[0x0010]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:13	-	DNM	0	**Reserved, Do Not Modify**
12	pwmckbd	DNM	0	**Reserved, Do Not Modify**
11	nollpol	DNM	0	**Reserved, Do Not Modify**
10	nolhpol	DNM	0	**Reserved, Do Not Modify**
9	pwmsync	DNM	0	**Reserved, Do Not Modify**

Wake-Up Timer Control			WUT_CTRL	[0x0010]
Bits	**Name**	**Access**	**Reset**	**Description**
8	pres3	R/W	0	**Timer Prescaler Select MSB** See *WUT_CTRL.pres* for details on this field's usage.
7	ten	R/W	0	**Timer Enable** 0: Timer disable 1: Timer enabled
6	tpol	DNM	0	**Reserved, Do Not Modify**
5:3	pres	R/W	0	**Timer Prescaler Select** Sets the timer's prescaler value. The prescaler divides the RTC 's 32.768KHz input clock sets the timer's count clock as shown in *Equation 21-1*. The wake-up timer's prescaler setting is a 4-bit value with *pres3* as the most significant bit and *pres* as the three least significant bits. See *Table 21-1* for details.
2:0	tmode	R/W	0	**Timer Mode Select** Sets the timer's operating mode. 0: One-shot 1: Continuous 2 – 4: Reserved 5: Compare 6 – 7: Reserved

Table 21-8: Wake-Up Timer Non-Overlapping Compare Register

Wake-Up Timer Non-Overlapping Compare			WUT_NOLCMP	[0x0014]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	-	DNM	0	**Reserved, Do Not Modify**

22. Watchdog Timer (WDT)

The watchdog timer (WDT) protects against corrupt or unreliable software, power faults, and other system-level problems which can place the IC into an improper operating state. The software must periodically write a unique 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. If the application cannot regain normal operation, the watchdog timer can generate a system reset as a last resort.

Some instances provide a windowed timer function. These instances support an additional feature that can detect watchdog timer resets that occur too early, 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 caused by a WDT late reset event or a WDT early reset event if the window function is enabled. If so, the software should take the desired action as part of its restart sequence.

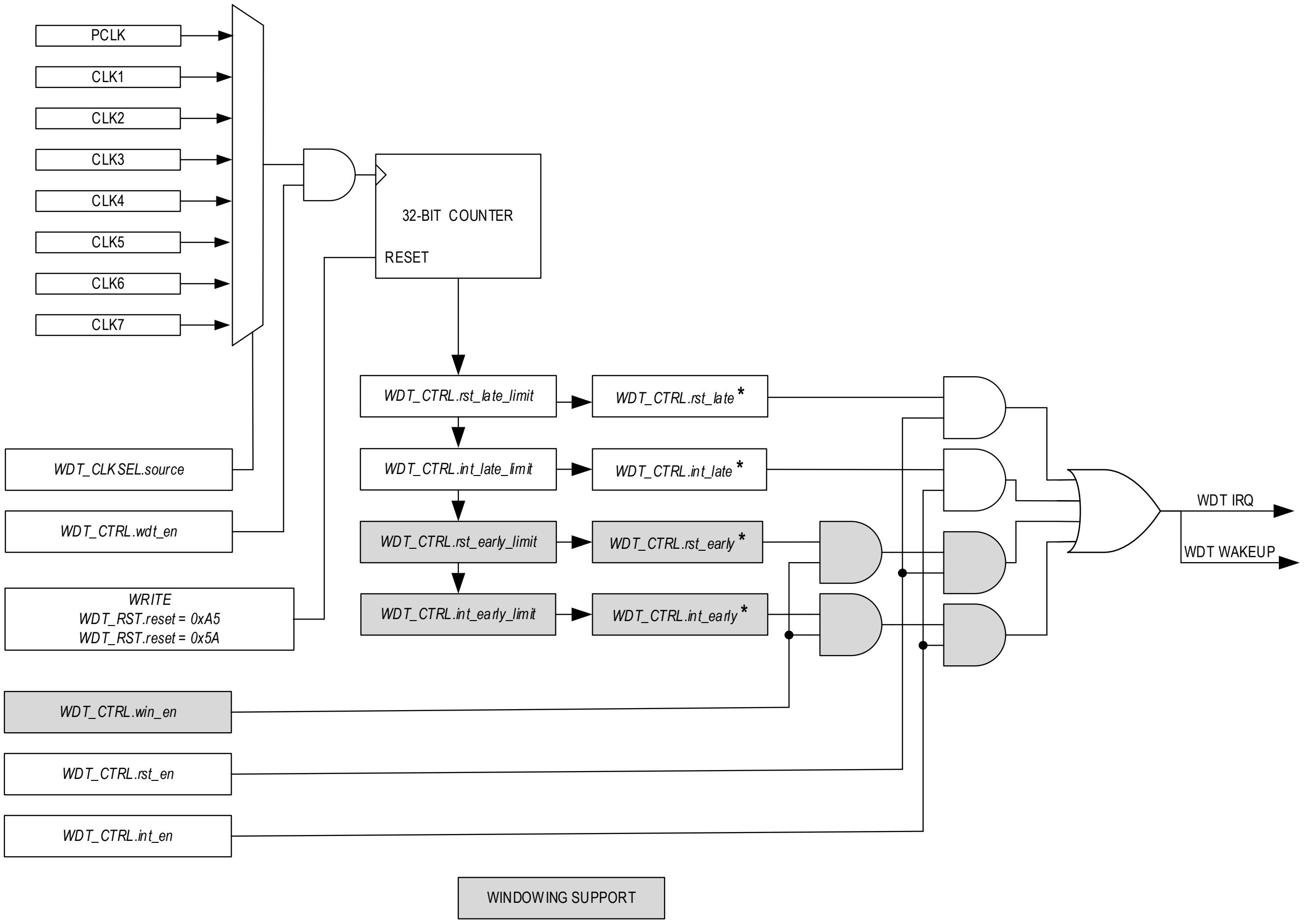
The WDT is a critical safety feature, and most fields are reset on POR or system reset events only.

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^{32} time-base ticks.

Figure 22-1 shows a high-level block diagram of the WDT.

Figure 22-1: Windowed Watchdog Timer Block Diagram



* INTERRUPT FLAGS ARE SET REGARDLESS OF THE ENABLED STATE OF WDTn_CTRL.win_en, WDTn_CTRL.wdt_int_en and WDTn_CTRL.wdt_rst_en.

22.1 Instances

Table 22-1 shows the peripheral instances, available clock sources, and windowed watchdog support.

Table 22-1: MAX78002 WDT Instances Summary

Instance	Register Access Name	Window Support	CLK0	CLK1	CLK2	CLK3	CLK4	CLK5	CLK6
WDT0	WDT0	Yes	PCLK	IPO	IBRO	INRO	ERTCO	EXT_CLK1 (P0.28 AF2)	ERFO
WDT1	WDT1								

22.2 Usage

When enabled, *WDTn_CNT.count* is incremented once every t_{WDTCLK} period. The software periodically executes the feed sequence during correct operation, resetting the *WDTn_CNT.count* field 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 interrupts 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 can 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 forces 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 attempts to regain program control by vectoring to the dedicated WDT interrupt service routine (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 generates a late reset event which sets the WDT late reset flag and generates a system interrupt.

Instances that 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. Analogously, the device attempts 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 to the device to correct the problem. The WDT generates an early reset event that sets the WDT late reset flag and generates a system interrupt.

The event flags are set regardless of the corresponding interrupt or reset enable and include the early interrupt and early event flags, even if the WDT is disabled (*WDTn_CTRL.win_en* = 0).

22.2.1 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.

22.2.2 Using the WDT as a Long-Interval Wakeup Timer

The WDT can be used as a very long internal wakeup source. Another application of the WDT is as a very long interval wakeup source from *SLEEP*.

22.3 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 re-enabled after writing 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 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*: 0xA5.
   b. Write *WDTn_RST.reset*: 0x5A.
3. If desired, enable or disable the timer.
4. Re-enable interrupts.

22.4 WDT Events

Multiple events are supported, as shown in *Table 22-2*. The corresponding event flag is set when the event occurs.

Typically, the system is configured such that the late interrupt events occur before the late reset events, and early interrupts occur when the feed sequence has the least error from the target time before the early reset events.

The event flags are set even if the corresponding interrupt enable or reset enable are not enabled and include the early interrupt flag and early event flag even if the window feature is disabled (*WDTn_CTRL.win_en* = 0).

The software must clear the event flags before enabling the WDT.

Table 22-2: WDT Event Summary

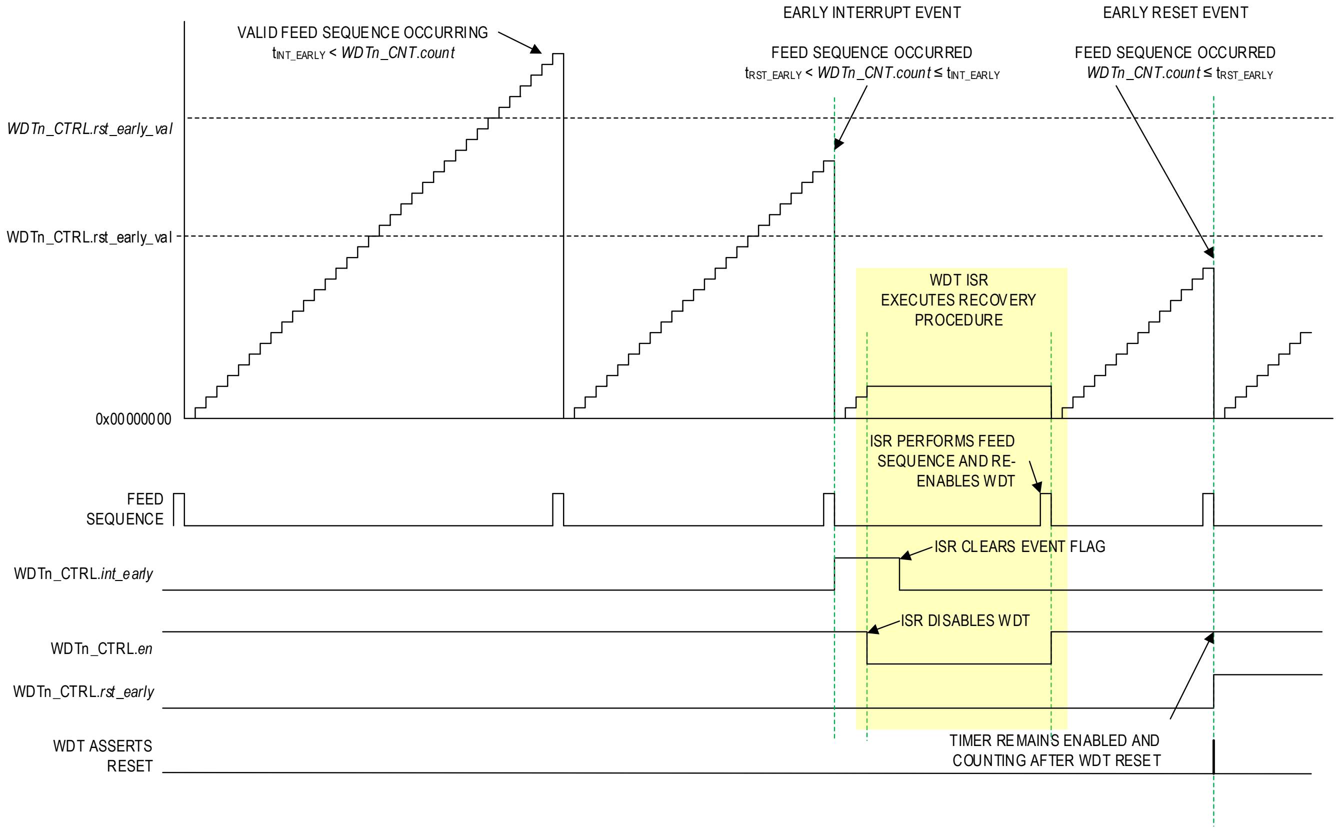
Event	Condition	Peripheral Interrupt Event Flag	Peripheral Interrupt Event Enable
Early Interrupt	Feed sequence occurs while *WDTn_CTRL.rst_early_val* ≤ *WDTn_CNT.count* < *WDTn_CTRL.int_early_val* *WDTn_CTRL.win_en = 1*	*WDTn_CTRL.int_early*	*WDTn_CTRL.wdt_int_en*
Early Reset	Feed sequence occurs while *WDTn_CNT.count* < *WDTn_CTRL.rst_early_val* *WDTn_CTRL.win_en = 1*	*WDTn_CTRL.rst_early*	*WDTn_CTRL.wdt_rst_en*
Interrupt Late	*WDTn_CNT.count* = *WDTn_CTRL.int_late_val*	*WDTn_CTRL.int_late*	*WDTn_CTRL.wdt_int_en*
Reset Late	*WDTn_CNT.count* = *WDTn_CTRL.rst_late_val*	*WDTn_CTRL.rst_late*	*WDTn_CTRL.wdt_rst_en*
Timer Enabled	*WDTn_CTRL.clkrdy* 0 → 1	No event flags are set by a timer enabled event	

22.4.1 WDT Early Reset

The early reset event occurs if the software performs the WDT feed sequence while the WDT count is less than the reset late value (*WDTn_CNT.count* < *WDTn_CTRL.rst_late_val*).

Figure 22-2 shows the sequencing details associated with an early reset event.

Figure 22-2: WDT Early Interrupt and Reset Event Sequencing Details


EARLY INTERRUPT EVENT
EARLY RESET EVENT
VALID FEED SEQUENCE OCCURRING
t_{INT_EARLY} < *WDTn_CNT.count*
FEED SEQUENCE OCCURRED
t_{RST_EARLY} < *WDTn_CNT.count* ≤ t_{INT_EARLY}
FEED SEQUENCE OCCURRED
WDTn_CNT.count ≤ t_{RST_EARLY}
WDTn_CTRL.rst_early_val
WDTn_CTRL.rst_early_val
WDT ISR EXECUTES RECOVERY PROCEDURE
0x00000000
ISR PERFORMS FEED SEQUENCE AND RE-ENABLES WDT
FEED SEQUENCE
ISR CLEARS EVENT FLAG
WDTn_CTRL.*int_early*
ISR DISABLES WDT
WDTn_CTRL.*en*
WDTn_CTRL.*rst_early*
TIMER REMAINS ENABLED AND COUNTING AFTER WDT RESET
WDT ASSERTS RESET

The following occurs when a WDT early reset event occurs:

1. The hardware sets *WDTn_CTRL.rst_early* to 1.
2. The hardware initiates a system reset.
   a. The hardware resets *WDTn_CNT.count* to 0x0000 0000 during the system reset event.
   b. The *WDTn_CTRL.en* and the *WDTn_CTRL.rst_early* fields are unaffected by a system reset.
3. After the system reset is complete, the WDT continues incrementing.

22.4.2 WDT Early Interrupt

The early interrupt event occurs if the software performs the WDT feed sequence while *WDTn_CTRL.rst_early_val* ≤ *WDTn_CNT.count* < *WDTn_CTRL.int_early_val* as shown in *Table 22-2*. *Figure 22-2* shows the sequencing details associated with an early reset event, including:

- The sequencing details associated with an early interrupt event.
- The required functions performed by the WDT interrupt handler.

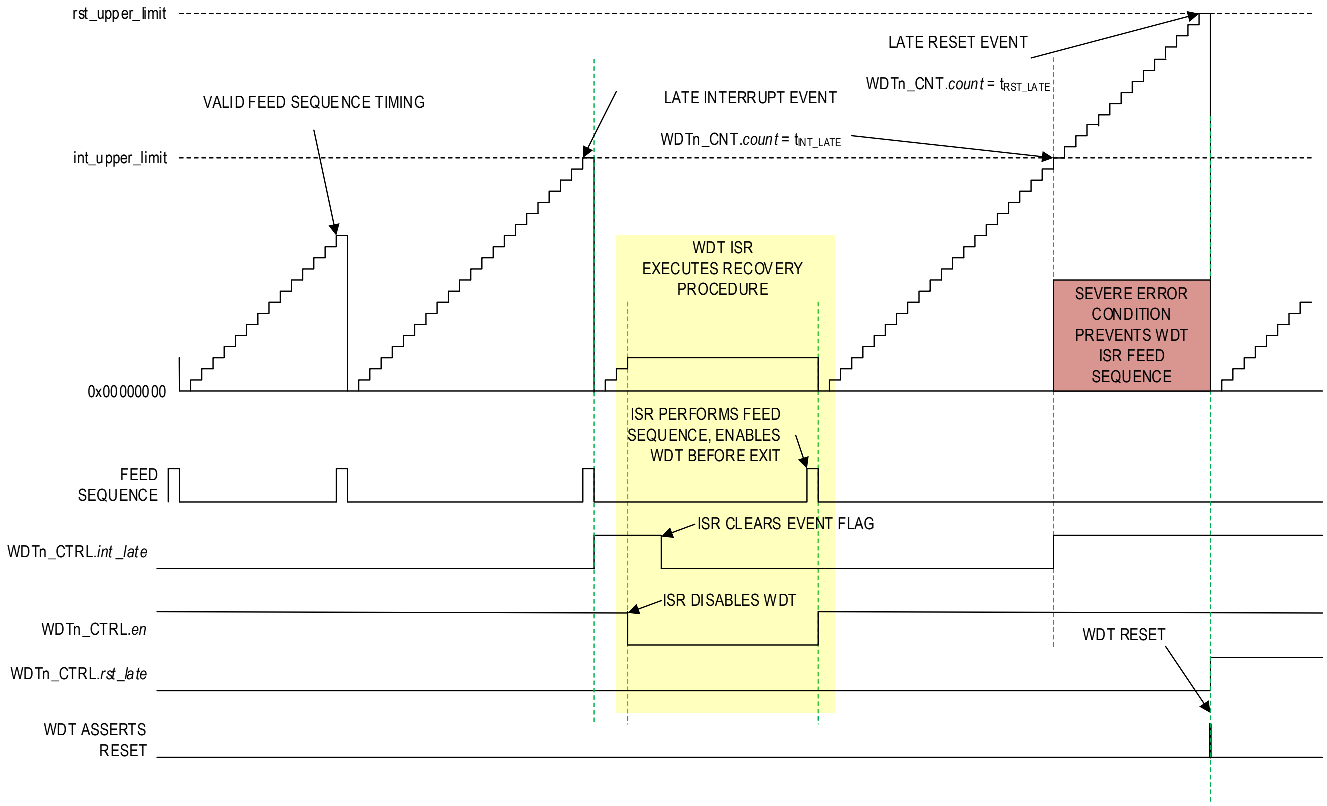
The following occurs when a WDT late interrupt event occurs:

1. The hardware sets *WDTn_CTRL.int_late* to 1.
2. The hardware initiates the WDT interrupt if enabled.

22.4.3 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 22-2*. *Figure 22-3* shows the sequencing details associated with a late reset event.

Figure 22-3: WDT Late Interrupt and Reset Event Sequencing Details



The following occurs when a WDT late reset event occurs:

1. The hardware sets *WDTn_CTRL.rst_late* to 1.
2. The hardware initiates a system reset:
   a. The hardware resets *WDTn_CNT.count* to 0x0000 0000 during the reset event.
   b. The *WDTn_CTRL.en* and *WDTn_CTRL.rst_late* fields are unaffected by a system reset.
3. After the hardware exits the system reset, the WDT continues incrementing after the system reset completes.

22.4.4 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 22-2*. *Figure 22-3* shows the sequencing details associated with a late interrupt event, including the required functions performed by the WDT interrupt handler.

The following occurs when WDT late interrupt event occurs:

1. The hardware sets *WDTn_CTRL.int_late* to 1.
2. The hardware initiates the WDT interrupt if enabled.

22.5 Initializing the WDT

The complete procedure for configuring the WDT is as follows:

1. Execute the WDT feed sequence and disable the WDT:
   a. Disable global interrupts.
   b. Write *WDTn_RST.reset* to 0xA5.
   c. Write *WDTn_RST.reset* to 0x5A.
   d. The hardware resets the WDT count (*WDTn_CNT.count* = 0x0000 0000).
   e. Set *WDTn_CTRL.en* to 0 to disable the WDT.
2. Verify the peripheral is disabled before proceeding:
   a. Poll *WDTn_CTRL.clkrdy* until it reads 1.
3. Set *WDTn_CTRL.clkrdy_ie =* 1 to generate a WDT enabled interrupt event.
4. Re-enable global interrupts.
5. Configure *WDTn_CLKSEL.source* to select the clock source.
6. Configure the standard thresholds:
   a. Configure *WDTn_CTRL.int_late* 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.
7. 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.
8. 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 is generated by both a WDT late interrupt event, and a WDT early interrupt event.
9. 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 is generated by a WDT late reset event and a WDT early reset event.
10. Execute the WDT feed sequence and enable the WDT:
    a. Disable global interrupts.
    b. Write *WDTn_RST.reset* to 0xA5.
    c. Write *WDTn_RST.reset* to 0x5A. The hardware resets *WDTn_CNT.count* = 0x0000 0000.
    d. Set *WDTn_CTRL.en* to 1 to enable the WDT.
11. Verify the peripheral is enabled before proceeding:
    a. Poll *WDTn_CTRL.clkrdy* until it reads 1, or
12. Set *WDTn_CTRL.clkrdy_ie =* 1 to generate a WDT enabled event interrupt.
13. Re-enable global interrupts.

22.6 Resets

The WDT is a critical safety feature. Most of the fields are reset by a POR or system reset events only; however, the enable field (*WDTn_CTRL.en*) and the interrupt flag fields are not reset by a system reset event.

22.7 Registers

See *Table 3-3* 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 22-3*. Register names for a specific instance are defined by replacing "n" with the instance number. For example, a register PERIPHERALn_CTRL resolves to PERIPHERAL0_CTRL and PERIPHERAL1_CTRL for instances 0 and 1, respectively.

See *Table 1-1* 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 22-3: 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

22.7.1 Register Details

Table 22-4: 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 is set even if *WDTn_CTRL.win_en* = 0 or *WDTn_CTRL.wdt_rst_en* = 0. The software must clear this field to 0. 0: Watchdog did not cause a 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 the *WDTn_CTRL.rst_early_val* field. This flag is set even if *WDTn_CTRL.win_en* = 0 or *WDTn_CTRL.wdt_rst_en* = 0. The software must clear this field to 0. 0: Watchdog did not cause a reset event. 1: Watchdog reset occurred before the time specified in the *WDTn_CTRL.rst_early_val* field.
29	win_en	R/W	0	**Window Function Enable** 0: Disabled. The WDT recognizes interrupt late and reset late events, supporting legacy implementations. 1: Enabled
28	clkrdy	R	0	**Clock Status** This field is cleared to 0 by the hardware when the software changes the state of the *WDTn_CTRL.en* field. The hardware sets this field to 1 when the change to the requested enable or disable is complete. 0: WDT clock is off 1: WDT clock is on
27	clkrdy_ie	R/W	0	**Clock Switch Ready Interrupt Enable** This interrupt prevents the software from needing to poll the *WDTn_CTRL.clkrdy* field to determine when the WDT clock is ready. When the *WDTn_CTRL.clkrdy* field transitions from 1 to 0, this interrupt signals the transition is complete. 0: Disabled 1: Enabled
26:24	-	RO	0	**Reserved**

WDT Control		WDTn_CTRL		[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
23:20	rst_early_val	R/W	0	**Reset Early 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: $2^{16} \times t_{WDTCLK}$ *Note: The watchdog timer must be disabled (WDTn_CTRL.en = 0) before changing this field.*
19:16	int_early_val	R/W	0	**Interrupt Early 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: $2^{16} \times t_{WDTCLK}$ *Note: The watchdog timer must be disabled (WDTn_CTRL.en = 0) before changing this field.*
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* field. This flag is set even if *WDTn_CTRL.win_en* = 0. 0: No interrupt event. 1: Interrupt event occurred. *Note: A WDT interrupt is generated if the WDT interrupt is enabled (WDTn_CTRL.wdt_int_en = 1).*
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

WDT Control		**WDTn_CTRL**		**[0x0000]**
Bits	**Name**	**Access**	**Reset**	**Description**
9	int_late	R/W	0	**Interrupt Late Flag** A watchdog feed sequence did not occur before the time determined by the *WDTn_CTRL.int_late_val* field. 0: No interrupt event 1: Interrupt event occurred *Note: A WDT interrupt is generated if the WDT interrupt is enabled (WDTn_CTRL.wdt_int_en = 1).*
8	en	R/W	0	**WDT Enable** This field enables/disables the WDT clock into the peripheral. *WDTn_CNT.count* 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: $2^{16} \times t_{WDTCLK}$ *Note: The watchdog timer must be disabled (WDTn_CTRL.en = 0) before changing this field.*
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: $2^{16} \times t_{WDTCLK}$ *Note: The watchdog timer must be disabled (WDTn_CTRL.en = 0) before changing this field.*

Table 22-5: 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 *WDTn_RST.reset*: 0xA5 2. Write *WDTn_RST.reset*: 0x5A Writes to the *WDTn_CTRL.en* field, which enables or disables 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 22-6: 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** See *Table 22-1* for the available clock options. 0: CLK0 1: CLK1 2: CLK2 3: CLK3 4: CLK4 5: CLK5 6: CLK6 7: CLK7 *†Note: This field is only reset on a POR and unaffected by other resets.* *Note: The watchdog timer must be disabled (WDTn_CTRL.en = 0) before changing this field.*

Table 22-7: WDT Count Register

WDT Count			**WDTn_CNT**	**[0x000C]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	count	R	0	**WDT Counter** The counter value for debugging. This register is reset by system reset, as well as the watchdog feeding sequence. *Note: The watchdog timer must be disabled (WDTn_CTRL.en = 0) before reading this field.*

23. 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-bits 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.

23.1 Instances

The device provides four instances of the pulse train engine peripheral.

- PT0 to PT3

All peripheral registers share a common register set.

23.2 Features

The pulse train outputs with individually programmable modes, patterns, and output enables. The pulse train engine uses the PCLK, ensuring all pulse train outputs use the same clock source. The pulse trains support the following features:

- 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.

23.3 Engine

The pulse train engine uses the PCLK 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.

23.3.1 Pulse Train Output Modes

Each pulse train output supports the following modes:

- Pulse train mode
- Bit pattern length
- Square wave mode

23.3.1.1 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 is configured in pulse train mode. The value of the *mode* field is the pattern bit length.

PTn_RATE_LENGTH.mode = 0:

PTn configured for pulse train mode (32-bit pattern).

23.3.1.2 In Pulse Train Mode, Set the Bit Pattern

If an output is set to pulse train mode, configure a custom bit pattern from 2- to 32-bits in length in the 32-bit register *PTn_TRAIN*. The pattern is shifted out LSB first. If the output is configured in square wave mode, then the *PTn_TRAIN* register is ignored.

Equation 23-1: Pulse Train Mode Output Function

$$PTn\_TRAIN = [Bit\ pattern\ for\ PTn]$$

23.3.1.3 Synchronize Two or More Outputs, if Needed

The write-only register *PTG_RESYNC* "PT Global Resync" allows two or more outputs to be reset and synchronized. Write to any bit in *PTG_RESYNC* 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.

23.3.1.4 Pulse Train Loop Mode

By default, a pulse train engine runs indefinitely until the software disables it.

A pulse train engine can be configured to repeat its pattern a specified number of times, referred to as 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.

23.3.1.5 Pulse Train Loop Delay

If the pulse train is configured in loop mode, a delay can be inserted after each repeated output pattern. Write the 12-bit field *PTn_LOOP.delay* with the number of PCLK cycles to delay between the MSB of the last pattern to the LSB of the next pattern to enable a delay. 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.

23.3.1.6 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 square wave mode.

If another pulse train's stop event triggers a running pulse train engine, automatic restart restarts the running pulse train engine from the beginning of its pattern. If another pulse train's stop event triggers a pulse train engine, 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: The configuration for automatic restart is set using the pulse train engine(s) triggered by the automatic restart, not the pulse train engine(s) that trigger the automatic restart. For example, the PT2_RESTART register configures which pulse train engine triggers PT2 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 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*

23.4 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: Before changing a pulse train output's configuration, the corresponding pulse train output should be halted to prevent unexpected behavior.

23.5 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.

Two additional registers are used to enable and disable the outputs to ensure safe and predictable operation.

23.5.1 Pulse Train Atomic Enable

PTG_SAFE_EN "Global Safe Enable" is a write-only register. To safely enable outputs without a read/modify/write, 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, enabling the corresponding pulse train engine. Writing a 0 to any bit position in the *PTG_SAFE_EN* register does not affect 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.

23.5.2 Pulse Train Atomic Disable

PTG_SAFE_DIS "Global Safe Disable" is a write-only register for disabling a pulse train engine without performing a read/modify/write. 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 does not affect 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.

23.6 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 does not affect 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.

23.7 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.

23.8 Registers

See *Table 3-3* 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 23-1*. 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 1-1* 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 23-1: Pulse Train Engine Register Summary

Offset	Register	Description
[0x0000]	*PTG_ENABLE*	PT Global Enable/Disable Control
[0x0004]	*PTG_RESYNC*	PT Global Resync
[0x0008]	*PTG_INTFL*	PT Stopped Global Status Flags
[0x000C]	*PTG_INTEN*	PT Global Interrupt Enable
[0x0010]	*PTG_SAFE_EN*	PT Global Safe Enable
[0x0014]	*PTG_SAFE_DIS*	PT Global Safe Disable
[0x0020]	*PTn_RATE_LENGTH*	PTn Configuration
[0x0024]	*PTn_TRAIN*	PTn Pulse Train Mode Bit Pattern
[0x0028]	*PTn_LOOP*	PTn Loop Control
[0x002C]	*PTn_RESTART*	PTn Automatic Restart
[0x0030]	*PTn_RATE_LENGTH*	PTn Configuration
[0x0034]	*PTn_TRAIN*	PT1 Pulse Train Mode Bit Pattern
[0x0038]	*PTn_LOOP*	PT1 Loop Control
[0x003C]	*PTn_RESTART*	PT1 Automatic Restart
[0x0040]	*PTn_RATE_LENGTH*	PT2 Configuration

Offset	Register	Description
[0x0044]	*PTn_TRAIN*	PT2 Pulse Train Mode Bit Pattern
[0x0048]	*PTn_LOOP*	PT2 Loop Control
[0x004C]	*PTn_RESTART*	PT2 Automatic Restart
[0x0050]	*PTn_RATE_LENGTH*	PT3 Configuration
[0x0054]	*PTn_TRAIN*	PT3 Pulse Train Mode Bit Pattern
[0x0058]	*PTn_LOOP*	PT3 Loop Control
[0x005C]	*PTn_RESTART*	PT3 Automatic Restart

23.8.1 Register Details

23.8.1.1 Pulse Train Engine Global Enable/Disable Register

This register enables each of the individual pulse trains. Write a 1 to the individual pulse train enable bits to enable the corresponding pulse train. When, for a given pulse train, the *PTn_LOOP.count* loop counter is set to a non-zero number, when the loop counter reaches zero, then the given pulse train engine stops, and the corresponding enable bit in this register is cleared by hardware.

Table 23-2: Pulse Train Engine Global Enable/Disable Register

PT Global Enable/Disable Control			**PTG_ENABLE**	**[0x0000]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved**
3	pt3	R/W	0	**Enable PT3** 0: Disabled 1: Enabled *Note: Disabling an active pulse train halts the output and does not generate a stop event.*
2	pt2	R/W	0	**Enable PT2** 0: Disabled 1: Enabled *Note: Disabling an active pulse train halts the output and does not generate a stop event.*
1	pt1	R/W	0	**Enable PT1** 0: Disabled 1: Enabled *Note: Disabling an active pulse train halts the output and does not generate a stop event.*
0	pt0	R/W	0	**Enable PT0** 0: Disabled 1: Enabled *Note: Disabling an active pulse train halts the output and does not generate a stop event.*

Table 23-3: Pulse Train Engine Resync Register

PT Resync Register			**PTG_RESYNC**	**[0x0004]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved**

PT Resync Register			**PTG_RESYNC**	**[0x0004]**
Bits	**Field**	**Access**	**Reset**	**Description**
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 *Note: Writing 1 has no effect if the corresponding pulse train is disabled.*
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 *Note: Writing 1 has no effect if the corresponding pulse train is disabled.*
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 *Note: Writing 1 has no effect if the corresponding pulse train is disabled.*
0	pt0	WO	-	**Resync Control for PT0** 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 *Note: Writing 1 has no effect if the corresponding pulse train is disabled.*

Table 23-4: Pulse Train Engine Stopped Interrupt Flag Register

PT Stopped Interrupt Flag Register			**PTG_INTFL**	**[0x0008]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved**
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.

PT Stopped Interrupt Flag Register			PTG_INTFL	[0x0008]
Bits	**Field**	**Access**	**Reset**	**Description**
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 23-5: Pulse Train Engine Interrupt Enable Register

PT Interrupt Enable Register			PTG_INTEN	[0x000C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved**
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 *PTG_INTFL* register. 0: Disabled. 1: Enabled.
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.

23.8.1.2 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 23-6: Pulse Train Engine Safe Enable Register

Pulse Train Engine Safe Enable Register			**PTG_SAFE_EN**	**[0x0010]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved**
3	pt3	WO	-	**Safe Enable Control for PT3** Writing a 1 sets *PTG_ENABLE.pt3*. 1: Enable corresponding pulse train. 0: No effect.
2	pt2	WO	-	**Safe Enable Control for PT2** Writing a 1 sets *PTG_ENABLE.pt2*. 1: Enable corresponding pulse train. 0: No effect.
1	pt1	WO	-	**Safe Enable Control for PT1** Writing a 1 sets *PTG_ENABLE.pt1*. 1: Enable corresponding pulse train 0: No effect
0	pt0	WO	-	**Safe Enable Control for PT0** Writing a 1 sets *PTG_ENABLE.pt0*. 1: Enable corresponding pulse train. 0: No effect.

23.8.1.3 Pulse Train Engine Safe Disable 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 23-7: Pulse Train Engine Safe Disable Register

Pulse Train Engine Safe Disable Register			**PTG_SAFE_DIS**	**[0x0014]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved**
3	pt3	WO	-	**Safe Disable Control for PT3** Writing a 1 clears *PTG_ENABLE.pt3*. 1: Disable corresponding pulse train. 0: No effect.
2	pt2	WO	-	**Safe Disable Control for PT2** Writing a 1 clears *PTG_ENABLE. pt2*. 1: Disable corresponding pulse train. 0: No effect.
1	pt1	WO	-	**Safe Disable Control for PT1** Writing a 1 clears *PTG_ENABLE.pt1*. 1: Disable corresponding pulse train. 0: No effect.

Pulse Train Engine Safe Disable Register		PTG_SAFE_DIS		[0x0014]
Bits	**Field**	**Access**	**Reset**	**Description**
0	pt0	WO	-	**Safe Disable Control for PT0** Writing a 1 clears *PTG_ENABLE.pt0*. 1: Disable corresponding pulse train. 0: No effect.

23.8.1.4 Pulse Train Registers

Table 23-8: Pulse Train Engine Configuration Register

Pulse Train *n* Configuration Register		PTn_RATE_LENGTH		[0x0020]
Bits	**Field**	**Access**	**Reset**	**Description**
31:27	mode	R/W	1	**Square Wave or Pulse Train Output Mode** This field selects 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. …: … 31: Pulse train mode, 31-bits long. *Note: If this field is set to 1 square wave mode, the PTn_TRAIN 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 PT*n* changes state by setting the divisor of the PT clock. Setting this field to 0 disables the PT*n*. For all other values, the following equation is used to calculate the rate.: $f_{PTn} = \frac{f_{PTE\_CLK}}{rate\_control}$ 0: Output halted. 1: $f_{PTn} = f_{PTE\_CLK}$ 2: $f_{PTn} = {f_{PTE\_CLK}}/{2}$ 3: $f_{PTn} = {f_{PTE\_CLK}}/{3}$ …

Table 23-9: Pulse Train Mode Bit Pattern Register

Pulse Train Mode Bit Pattern			PTn_TRAIN	[0x0024]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	-	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 23-10: Pulse Train n Loop Configuration Register

Pulse Train Loop Configuration			PTn_LOOP	[0x0028]
Bits	**Field**	**Access**	**Reset**	**Description**
31:28	-	RO	0	**Reserved**
27:16	delay	R/W	0	**Pulse Train Delay Between Loops** Sets the delay in the number of PCLK cycles, that the output pauses between loops. The *PTn_LOOP.count* is decremented after the delay. *Note: This field is ignored if software writes 0 to the PTn_LOOP.count field.*
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 *PTG_INTFL* flag is set. Writing this field to 0 to repeat the pulse train pattern indefinitely. *Note: This field is ignored in square wave mode.*

Table 23-11: 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**
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**
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. 0: PT0. 1: PT1. 2: PT2. 3: PT3. 4 - 31: 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 the pulse train from the beginning of its pattern.

Pulse Train Automatic Restart Configuration		PTn_RESTART		[0x002C]
Bits	**Field**	**Access**	**Reset**	**Description**
6:5	-	RO	0	**Reserved**
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. 0: PT0. 1: PT1. 2: PT2. 3: PT3. 4 - 31: Reserved.

24. Cyclic Redundancy Check (CRC)

The CRC engine performs CRC calculations on data written to the CRC data input register.

The features include:

- User-definable polynomials up to x^{32} (33 terms).
- DMA support.
- Supports automatic byte swap of data input and calculated output.
- Supports big-endian or little-endian data input and calculated output.
- Supports input reflection.

An *n*-bit CRC can detect the following types of errors:

- Single-bit errors.
- Two bit errors for block lengths less than 2^k 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.

In general, all but 1 out of 2^n errors are detected:

- 99.998% for a 16-bit CRC.
- 99.99999998% for a 32-bit CRC.

24.1 Instances

Instances of the peripheral are listed in *Table 24-1*.

Table 24-1: MAX78002 CRC Instances

Instance	Maximum Terms	DMA Support	Big- and Little-Endian
CRC	33 (2^{32})	Yes	Yes

24.2 Usage

A CRC value is often 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.

The software must configure the CRC polynomial, the starting CRC value, and the endianness of the data. Once configured, the software or the standard DMA engine transfers the data in either 8-bit, 16-bit, or 32-bit words to the CRC engine by writing to the corresponding data input register. Use the *CRC_DATAIN8* register for 8-bit data, the *CRC_DATAIN16* register for 16-bit data, and the *CRC_DATAIN32* register for 32-bit data. The hardware automatically sets the *CRC_CTRL*.*busy* field to 1 while the CRC engine is calculating a CRC over the input data. When the *CRC_CTRL*.*busy* field reads 0, the CRC result is available in the *CRC_VAL* register. The software or the standard DMA engine must track the data transferred to the CRC engine to determine when the CRC is finalized.

Because the receiving end calculates a new CRC on both the data and received CRC, 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, the CRC is reversed bitwise, with the highest-order term of the remainder in the LSB position. Software CRC algorithms typically manage this by calculating everything backward. The software reverses the polynomial and does right shifts on the data. The resulting CRC is bit swapped and in the correct format.

By default, the CRC is calculated using the LSB first (*CRC_CTRL.msb* = 0.) When calculating the CRC using MSB first data (reflected), the software must set *CRC_CTRL.msb* to 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 ($x^0 \ldots x^{32}$).

Table 24-2: Organization of Calculated Result in the CRC_VAL.value Field

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 reflected but not byte swapped
0	1	The CRC value returned is byte swapped but not reflected
1	1	The CRC value returned is reflected and then byte swapped

The CRC can be calculated on the MSB of the data first by setting the *CRC_CTRL.msb* field to 1, this is referred to as reflection. The CRC polynomial register, *CRC_POLY*, must be left-justified. The hardware implies the MSB of the polynomial just as it does when calculating the CRC LSB first. The LSB position of the polynomial must be set; this defines the length of the CRC. The initial value of the CRC, *CRC_VAL.value*, must also be left justified. When the CRC calculation is complete using MSB first, the software must right shift the calculated CRC value, *CRC_VAL.value*, by right shifting the output value if the CRC polynomial is less than 32-bits.

24.3 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 24-3* shows common CRC polynomials.

The reset value of the *CRC_POLY.poly* field is the *CRC-32 Ethernet* polynomial. This polynomial is used by Ethernet and file compression utilities such as zip or gzip.

Note: Only write to the CRC polynomial register, CRC_POLY.poly, when the CRC_CTRL.busy field is 0.

Table 24-3: Common CRC Polynomials

Algorithm	**Polynomial Expression**	**Order**	**Polynomial**
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
CRC-CCITT	$x^{16}+x^{12}+x^{5}+x^{0}$	LSB	0x0000 8408
CRC-16	$x^{16}+x^{15}+x^{2}+x^{0}$	LSB	0x0000 A001
USB Data	$x^{16}+x^{15}+x^{2}+x^{0}$	MSB	0x8005 0000
Parity	$x^{1}+x^{0}$	LSB	0x0000 0001

24.4 Software CRC Calculations

The software can perform CRC calculations by writing directly to the CRC data input register. Each write to the CRC data input register triggers the hardware to compute the CRC value. The software is responsible for loading all data for the CRC into the CRC data input register. When complete, the result is read from the *CRC_VAL* register.

Use the following procedure to calculate a CRC:

1. Disable the CRC peripheral by setting the field *CRC_CTRL.en* to 0.
2. Configure input and output data format fields:
   a. *CRC_CTRL.msb*
   b. *CRC_CTRL.byte_swap_in*
   c. *CRC_CTRL.byte_swap_out*
3. Set the polynomial by writing to the *CRC_POLY.poly* field.
4. Set the initial value by writing to the *CRC_VAL.value* field.
   a. For a 32-bit CRC, write the initial value to the *CRC_VAL* register.
   b. For a 16-bit or 8-bit CRC, the unused bits in the *CRC_VAL* register must be set to 0.
5. Set the *CRC_CTRL.en* field to 1 to enable the peripheral.
6. Write a value to be processed to data input register.
   a. Calculate an 8-bit CRC by writing an 8-bit value to the *CRC_DATAIN8* register.
   b. Calculate a 16-bit CRC by writing a 16-bit value to the *CRC_DATAIN16* register.
   c. Calculate a 32-bit CRC by writing a 32-bit value to the *CRC_DATAIN32* register.
7. Poll the *CRC_CTRL.busy* field until it reads 0.
8. Repeat steps 6 and 7 until all input data is complete.
9. Disable the CRC peripheral by clearing the *CRC_CTRL.en* field to 0.
10. Read the CRC value from the *CRC_VAL.value* field.

24.5 DMA CRC Calculations

The CRC engine requests new data from the DMA controller when the fields *CRC_CTRL.en* and *CRC_CTRL.dma_en* are both set to 1. Enable the corresponding DMA channel's interrupt to receive an interrupt event when the CRC is complete. It is also possible for software to poll the DMA channel's interrupt flag directly by reading the *DMA_INTFL.ch<n>* flag until it reads 1.

Use the following procedure to calculate a CRC value using DMA:

1. Set *CRC_CTRL.en* = 0 to disable the peripheral.
2. Configure the DMA:
   a. Set *CRC_CTRL.dma_en* = 1 to enable DMA mode.
   b. See the DMA *Usage* section for details on configuring the DMA for a memory to peripheral transfer.
   c. Set the *DMA_CHn_CTRL.dstwd* field to match the size of the CRC calculation (0 for 8-bit, 1 for half-word, or 2 for word)
3. Configure the input and output data formats:
   a. *CRC_CTRL.msb*
   b. *CRC_CTRL.byte_swap_in*
   c. *CRC_CTRL.byte_swap_out*
4. Set the polynomial by writing to the *CRC_POLY.poly* field.
5. Set the initial value by writing to the *CRC_VAL* register.
   a. For a 32-bit CRC, write the initial value to the *CRC_VAL* register.
   b. For a 16-bit or an 8-bit CRC, the unused bits in the *CRC_VAL* register must be set to 0.
6. Set the *CRC_CTRL.en* field to 1 to enable the peripheral.
7. When the DMA operation completes, the hardware:
   a. Clears the *CRC_CTRL.busy* field to 0.
   b. Loads the new CRC value into the *CRC_VAL.value* field.
   c. Sets the *DMA_INTFL.ch<n>* field to 1 and generates a DMA interrupt if the *DMA_INTEN.ch<n>* field was set to 1.
8. Disable the CRC peripheral by clearing the *CRC_CTRL.en* field to 0.
9. Read the CRC value from the *CRC_VAL.value* field.

24.6 Registers

See *Table 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 24-4: CRC Register Summary

Offset	Name	Description
[0x0000]	*CRC_CTRL*	CRC Control Register
[0x0004]	*CRC_DATAIN8*	CRC 8-Bit Data Input Register
[0x0004]	*CRC_DATAIN16*	CRC 16-Bit Data Input Register
[0x0004]	*CRC_DATAIN32*	CRC 32-Bit Data Input Register
[0x0008]	*CRC_POLY*	CRC Polynomial Register
[0x000C]	*CRC_VAL*	CRC Value Register

24.6.1 Register Details

Table 24-5: CRC Control Register

CRC Control			CRC_CTRL	[0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**
31:17	-	RO	0	**Reserved**
16	busy	R	0	**CRC Busy** 0: Not busy. 1: Busy.
15:5	-	RO	0	**Reserved**
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: The data input is processed least significant byte first. 1: The data input is processed most significant byte first.
2	msb	R/W	0	**Most Significant Bit Order** 0: LSB data first. 1: MSB data first (reflected).
1	dma_en	R/W	0	**DMA Enable** Set this field to 1 to enable a DMA request when the CRC calculation is complete (*CRC_CTRL.busy* = 0.) 0: Disabled. 1: Enabled.
0	en	R/W	0	**CRC Enable** 0: Disabled. 1: Enabled.

Table 24-6: CRC 8-Bit Data Input Register

CRC 8-Bit Data Input			CRC_DATAIN8	[0x0004]
Bits	**Field**	**Access**	**Reset**	**Description**
7:0	data	W	0	**CRC Data Input** Write 8-bit values to this register to calculate 8-bit CRCs. See *Table 24-2* for details on the byte and bit ordering of the data in this register. *Note: Do not write to this register if CRC_CTRL.busy = 1 or CRC_CTRL.en = 0.*

Table 24-7: CRC 16-Bit Data Input Register

CRC Data 16-Bit Input			CRC_DATAIN16	[0x0004]
Bits	**Field**	**Access**	**Reset**	**Description**
15:0	data	W	0	**CRC Data Input** Write 16-bit values to this register to calculate 16-bit CRCs. See *Table 24-2* for details on the byte and bit ordering of the data in this register. *Note: Do not write to this register if CRC_CTRL.busy = 1 or CRC_CTRL.en = 0.*

Table 24-8: CRC 32-Bit Data Input Register

CRC 32-Bit Data Input			CRC_DATAIN32	[0x0004]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	data	W	0	**CRC Data Input** Write 32-bit values to this register to calculate 32-bit CRCs. See *Table 24-2* for details on the byte and bit ordering of the data in this register. *Note: Do not write to this register if CRC_CTRL.busy = 1 or CRC_CTRL.en = 0.*

Table 24-9: CRC Polynomial Register

CRC Polynomial			CRC_POLY	[0x0008]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	poly	R/W	0xEDB8 8320	**CRC Polynomial** See *Table 24-2* for details on the byte and bit ordering of the data in this register.

Table 24-10: CRC Value Register

CRC Value			CRC_VAL	[0x000C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:0	value	R/W	0	**Current CRC Value** The software can write to this register to set the initial state of the accelerator. This register should only be read or written when *CRC_CTRL*.*busy* = 0. See *Table 24-2* for details on the byte and bit ordering of the data in this register.

25. Advanced Encryption Standard (AES)

The device provides a hardware AES accelerator to perform calculations on blocks up to 128 bits.

The features include:

- Supports multiple key sizes:
  - 128-bits.
  - 192-bits.
  - 256-bits.
- DMA support for both receive and transmit channels.
- Supports multiple key sources:
  - Encryption using the external AES key.
  - Decryption using the external AES key.
  - Decryption using the locally generated decryption key.

25.1 Instances

Instances of the peripheral are listed in *Table 25-1*. Disable the peripheral by clearing *AES_CTRL.en* = 0 before writing to any register field.

Table 25-1: MAX78002 AES Instances

Instance	128-Bit Key	192-Bit Key	256-Bit Key	DMA Support
AES	Yes	Yes	Yes	Yes

25.2 Encryption of 128-Bit Blocks of Data Using FIFO

AES operations are typically performed on 128-bits of data at a time. The simplest use case is to have software encrypt 128-bit blocks of data. The *AES_CTRL.start* field is unused in this case.

1. Generate a key.
2. Wait for the hardware to clear *AES_STATUS.busy* = 0.
3. Clear *AES_CTRL.en* = 0 to disable the peripheral.
4. If *AES_STATUS.input_em* = 0, set *AES_CTRL.input_flush* = 1 to flush the input FIFO.
5. If *AES_STATUS.output_em* = 0, set *AES_CTRL.output_flush* = 1 to flush the output FIFO.
6. Set *AES_CTRL.key_size* to desired setting.
7. Configure *AES_CTRL.type* = 00 (encryption with external key).
8. If interrupts are desired, set *AES_INTEN.done* = 1 so that an interrupt is triggered at the end of the AES calculation.
9. Set *AES_CTRL.en* = 1 to enable the peripheral.
10. Write four 32-bit words of data to *AES_FIFO.data*.
    a. The hardware starts the AES calculation.
11. If *AES_INTEN.done* = 1, an interrupt is triggered after the AES calculation is complete.
12. If *AES_INTEN.done* = 0, the software should poll *AES_STATUS.busy* until it reads 0.
13. Read four 32-bit words from *AES_FIFO.data* (least significant word first).
14. Repeat steps 10 to 13 until all 128-bit blocks are processed.

25.3 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 FIFO. This is not a requirement. The AES could use DMA on one side and software on the other for the application. It is required that for each DMA transmit request the DMA writes four 32-bit words of data into the AES FIFO. It is required that for each DMA receive request, the DMA reads four 32-bit words of data out of the AES FIFO.

The *AES_CTRL.start* field is unused in this case. The state of *AES_STATUS.busy* and *AES_INTFL.done* is indeterminate during DMA operations. The software must clear *AES_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 *AES_FIFO.data*.

Assuming the DMA is continuously filling the data input FIFO, the calculations are completed in the following number of SYS_CLK cycles:

- 128-bit key: 181
- 192-bit key: 213
- 256-bit key: 245

The procedure to use DMA encryption/decryption is:

1. Generate a key.
2. Initialize the AES receive and transmit channels for the DMA controller.
3. Wait for the hardware to clear *AES_STATUS.busy* = 0.
4. Clear *AES_CTRL.en* = 0 to disable the peripheral.
5. If *AES_STATUS.input_em* = 0, set *AES_CTRL.input_flush* = 1 to flush the input FIFO.
6. If *AES_STATUS.output_em* = 0, set *AES_CTRL.output_flush* = 1 to flush the output FIFO.
7. Set *AES_CTRL.key_size* to the desired setting.
8. Configure *AES_CTRL.type* = 0 (encryption with external key).
9. Ensure *AES_INTEN.done* = 0 during DMA operations.
10. Set *AES_CTRL.en* = 1 to enable the peripheral.
11. The DMA fills the FIFO, and the hardware begins the AES calculation.
12. When an AES calculation is completed, the AES hardware signals to the DMA that the data output FIFO is full and that it should be emptied. If the DMA does not empty the FIFO before the next calculation is complete, the hardware sets *AES_STATUS.output_full* = 1.

Step 11 and step 12 are repeated if the DMA has new data to write to the data input FIFO.

Note: 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 must be encrypted manually. See Encryption of Blocks Less Than 128-Bits *for details*

25.4 Encryption of Blocks Less Than 128-Bits

The AES engine automatically starts a calculation when a write of 128-bits or four writes of 32-bits occurs. Operations of less than 128-bits use the start field to initiate the AES calculation.

1. Generate a key.
2. Wait for the hardware to clear *AES_STATUS.busy* = 0.
3. Clear *AES_CTRL.en* = 0 to disable the peripheral.
4. If *AES_STATUS.input_em* = 0, set *AES_CTRL.input_flush* = 1 to flush the input FIFO.
5. If *AES_STATUS.output_em* = 0, set *AES_CTRL.output_flush* = 1 to flush the output FIFO.
6. Set *AES_CTRL.key_size* to desired setting.
7. Configure *AES_CTRL.type* = 0 (encryption with external key).
8. If interrupts are desired, set *AES_INTEN.done* = 1, so that an interrupt is triggered at the end of the AES calculation.
9. Set *AES_CTRL.en* = 1 to enable the peripheral.
10. Write from one to three 32-bit words of data to *AES_FIFO.data* (least significant word first).
11. Start the calculation manually by setting *AES_CTRL.start* = 1.
12. If *AES_INTEN.done* = 1, an interrupt is triggered after the AES calculation is complete.
13. If *AES_INTEN.done* = 0, the software should poll *AES_STATUS.busy* until it reads 0.
14. Read four 32-bit words from *AES_FIFO.data* (least significant word first).

25.5 Decryption

The decryption of data is very similar to encryption. The only difference is in the setting of the *AES_CTRL.type* field. There are two 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 before doing the first decryption to ensure that it has been generated.

25.6 Interrupt Events

The peripheral generates interrupts for the events shown in *Table 25-2*. 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 generate an interrupt if the corresponding interrupt enable field is set. The flags must be cleared by the software, typically in the interrupt handler.

Table 25-2: Interrupt Events

Event	Local Interrupt Flag	Local Interrupt Enable
Data Output FIFO Overrun	*AES_INTFL.ov*	*AES_INTEN.ov*
Key Zero	*AES_INTFL.key_zero*	*AES_INTEN.key_zero*
Key Change	*AES_INTFL.key_change*	*AES_INTEN.key_change*
Calculation Done	*AES_INTFL.done*	*AES_INTEN.done*

25.6.1 Data Output FIFO Overrun

When an AES calculation is completed, the AES hardware signals to the DMA that the data output FIFO is full and that it should be emptied. If the DMA does not empty the FIFO before the next calculation is complete, a data output FIFO overrun event occurs, and the corresponding local interrupt flag is set.

25.6.2 Key Zero

Attempting a calculation with a key of all zeros generates a key zero event.

25.6.3 Key Change

Writing to any key register while *AES_STATUS.busy* = 1 generates a key change event.

25.6.4 Calculation Done

The transition of *AES_STATUS.busy* = 1 to *AES_STATUS.busy* = 0 generates a calculation done event. The calculation done event interrupt must be disabled when using the DMA.

25.7 Registers

See *Table 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 reset.

Table 25-3: AES Register Summary

Offset	Name	Description
[0x0000]	*AES_CTRL*	AES Control Register
[0x0004]	*AES_STATUS*	AES Status Register
[0x0008]	*AES_INTFL*	AES Interrupt Flag Register
[0x000C]	*AES_INTEN*	AES Interrupt Enable Register
[0x0010]	*AES_FIFO*	AES Data FIFO

25.7.1 Register Details

Table 25-4: AES Control Register

AES Control			**AES_CTRL**	**[0x0000]**
Bits	**Field**	**Access**	**Reset**	**Description**
31:10	-	RO	0	**Reserved**
9:8	type	R/W	0	**Encryption Type** 0b00: Encryption using the external AES key. 0b01: Decryption using the external AES key. 0b10: Decryption using the locally generated decryption key. 0b11: Reserved.
7:6	key_size	R/W	0	**Encryption Key Size** 0b00: 128-bits. 0b01: 192-bits. 0b10: 256-bits. 0b11: Reserved.
5	output_flush	R/W1O	0	**Flush Data Output FIFO** This field always read 0. 0: No action. 1: Flush.

AES Control			AES_CTRL	[0x0000]
Bits	**Field**	**Access**	**Reset**	**Description**
4	input_flush	R/W1O	0	**Flush Data Input FIFO** This field always read 0. 0: No action. 1: Flush.
3	start	R/W1O	0	**Start AES Calculation** This field forces the start of an AES calculation regardless of the state of the data input FIFO. This allows an AES calculation on less than 128-bits of data since an AES calculation normally starts when the data input FIFO is full. This field always read 0. 0: No action. 1: Start calculation.
2	dma_tx_en	R/W	0	**DMA Request To Write Data Input FIFO** When enabled, a DMA request is generated when the data input FIFO is empty. 0: Disabled. 1: Enabled.
1	dma_rx_en	R/W	0	**DMA Request To Read Data Output FIFO** When enabled, a DMA request is generated when the data output FIFO is full. 0: Disabled. 1: Enabled.
0	en	R/W	0	**AES Enable** 0: Disabled. 1: Enabled.

Table 25-5: AES Status Register

AES Status			AES_STATUS	[0x0004]
Bits	**Field**	**Access**	**Reset**	**Description**
31:5	-	RO	0	**Reserved**
4	output_full	R	0	**Output FIFO Full** 0: Normal operation. 1: Full.
3	output_em	R	0	**Output FIFO Empty** 0: Normal operation. 1: Empty.
2	input_full	R	0	**Input FIFO Full** 0: Normal operation. 1: Full.
1	input_em	R	0	**Input FIFO Empty** 0: Normal operation. 1: Empty.
0	busy	R	0	**AES Busy** 0: Normal operation. 1: Busy.

Table 25-6: AES Interrupt Flag Register

AES Interrupt Flag			AES_INTFL	[0x0008]
Bits	**Field**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved**
3	ov	R/W1C	0	**Data Output FIFO Overrun Event Interrupt** 0: Normal operation. 1: Event occurred.

AES Interrupt Flag			AES_INTFL	[0x0008]
Bits	**Field**	**Access**	**Reset**	**Description**
2	key_zero	R/W1C	0	**Key Zero Event Interrupt** 0: Normal operation. 1: Event occurred.
1	key_change	R/W1C	0	**Key Change Event Interrupt** 0: Normal operation. 1: Event occurred.
0	done	R/W1C	0	**Calculation Done Event Interrupt** 0: Normal operation. 1: Event occurred.

Table 25-7: AES Interrupt Enable Register

AES Interrupt Enable			AES_INTEN	[0x000C]
Bits	**Field**	**Access**	**Reset**	**Description**
31:4	-	RO	0	**Reserved**
3	ov	R/W1C	0	**Data Output FIFO Overrun Event Interrupt Enable** 0: Enabled. 1: Disabled.
2	key_zero	R/W1C	0	**Key Zero Event Interrupt Enable** 0: Enabled. 1: Disabled.
1	key_change	R/W1C	0	**Key Change Event Interrupt Enable** 0: Enabled. 1: Disabled.
0	done	R/W1C	0	**Calculation Done Event Interrupt Enable** This interrupt must be disabled when using the DMA. 0: Enabled. 1: Disabled.

Table 25-8: AES FIFO Register

AES Data			AES_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. The 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.

26. TRNG Engine

The Analog Devices-supplied Universal Cryptographic Library (UCL) provides a function to generate random numbers intended to meet the requirements of common 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. Analog Devices works directly with the customer's accredited testing laboratory to provide any information regarding the TRNG 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 Analog Devices UCL to generate random numbers.

Software can use the TRNG engine to generate AES keys using a hardware key derivation function (HKDF) and using the TRNG as input to the HKDF.

26.1 Registers

See *Table 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 26-1: TRNG Register Summary

Offset	Register	Name
[0x0000]	*TRNG_CTRL*	TRNG Control Register
[0x0004]	*TRNG_STATUS*	TRNG Status Register
[0x0008]	*TRNG_DATA*	TRNG Data Register

26.1.1 Register Details

Table 26-2: TRNG Control Register

Control		**TRNG_CTRL**		**[0x0000]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:16	-	RO	0	**Reserved**
15	keywipe	R/W	0	**Wipe Key** Write this field to 1 to wipe the TRNG key.
14:4	-	RO	0	**Reserved**
3	keygen	R/W	0	**Generate Key** Write this field to 1 to generate a key using the TRNG.
2	-	RO	0	**Reserved**
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**

Table 26-3: TRNG Status Register

Status		**TRNG_STATUS**		**[0x0004]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:1	-	RO	0	**Reserved**

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 is generated if *TRNG_CTRL.rnd_ie* = 1.

Table 26-4: TRNG Data Register

Data			TRNG_DATA	[0x0008]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	data	RO	0	**TRNG Data** The 32-bit random number generated is available in this field when *TRNG_STATUS.rdy* = 1.

27. Secure Digital Host Controller (SDHC)

The SDHC provides an interface between the AHB and Embedded Multimedia Cards (e.MMC), Secure Digital I/O (SDIO) cards, Standard Capacity SD Memory Cards, and High-Capacity SD Memory Cards. The SDHC handles the SDIO/SD protocol at the transmission level, packing data, adding cyclic redundancy check (CRC), Start/End bit, and checking for transaction format correctness. Details of the SD communication and protocol are not part of the scope of this document. The SDHC only supports a single SD card.

SD memory card and SDIO card specifications are available at *https://www.sdcard.org*.

The e.MMC specifications are available from JEDEC at *http://www.jedec.org*.

Compliance

- SD Host Controller Standard Specification Version 3.00.
- SDIO Card Specification Version 3.0.
- SD Memory Card Specification Version 3.01.
- SD Memory Card Security Specification version 1.01.
- e.MMC Specification version 4.51.

SD/SDIO Card Interface

- Supports SDR50 with SDHC clock of up to 60MHz (30MB/sec).
- Supports DDR50 with SDHC clock of up to 30MHz (30MB/sec).
- Designed to work with I/O cards, Read-Only cards, and Read/Write cards.
- 1-bit and 4-bit data transfers in SD modes and SPI mode.
- Double buffer for transfers configurable from 512B to 1KB.
- Auto Command (AutoCMD12 or AutoCMD23) support.
- Multi-block transfers.
- Variable-length data transfers.
- Default and high-speed mode transfers.
- Card insertion/removal events.
- Read Wait Control, Suspend/Resume operation.
- CRC7 for command and CRC16 for data integrity.
- Single Operation DMA (SDMA) for data transfer.
- Advanced DMA (ADMA) support.

27.1 Instances

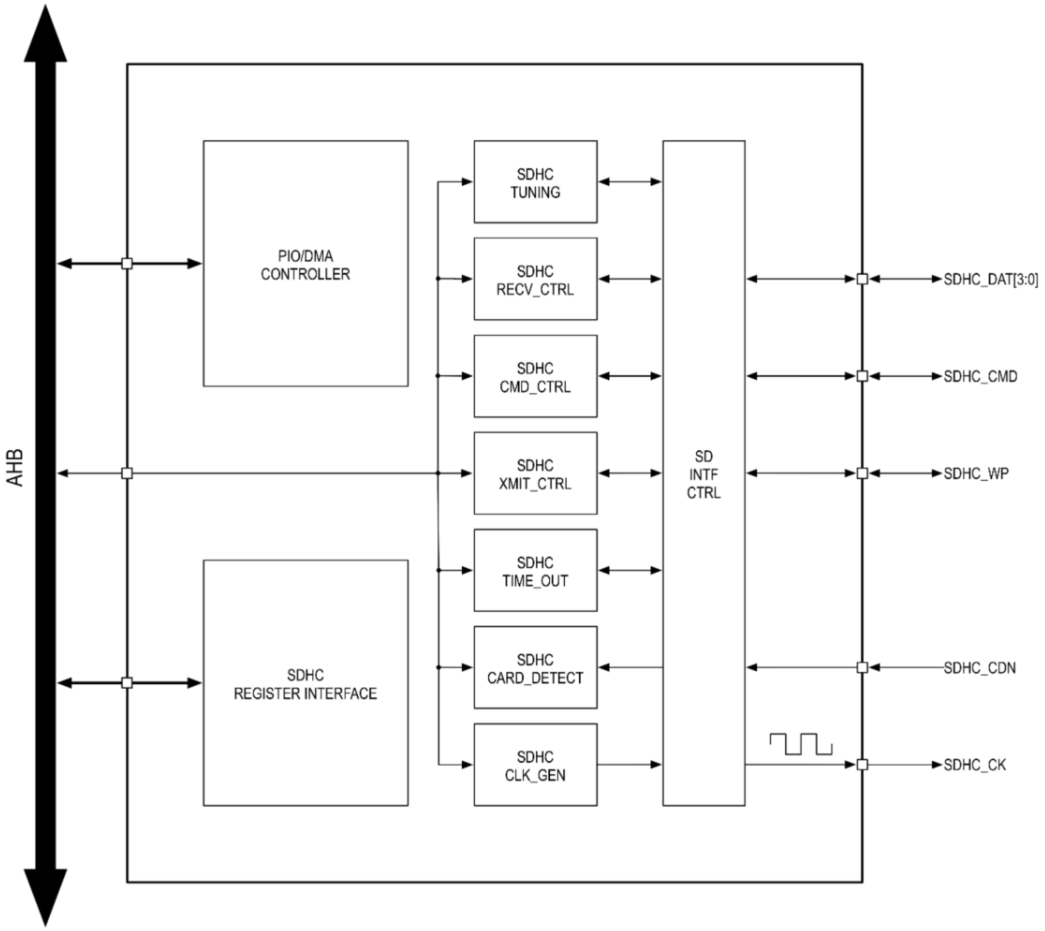
The SDHC pin mapping for the SD Host Controller Standard Specification Version 3.0 are shown in *Table 27-1*.

Table 27-1: MAX78002 SDHC Alternate Function Names to SDHC Specification Pin Names

Alternate Function	SDHC Pin Name	Direction	Signal Description
SDHC_CDN	SDCD#	I	Card present, active low.
SDHC_CLK	SDCLK	O	SD clock signal.
SDHC_WP	SDWP	I	Write protect signal, active high.
SDHC_CMD	CMD	I/O	SD bus command signal.
SDHC_DAT0	DAT[0]	I/O	SD data bus bit 0.
SDHC_DAT1	DAT[1]	I/O	SD data bus bit 1.
SDHC_DAT2	DAT[2]	I/O	SD data bus bit 2.
SDHC_DAT3	DAT[3]	I/O	SD data bus bit 3.
Note: Refer to the device data sheet's Pin Description table for Alternate Function mapping to pin numbers.			

For configuration of the GPIO for SDHC peripheral usage see *Alternate Function Configuration*.

Figure 27-1: SDHC Block Diagram



27.2 SDHC Peripheral Clock Selection

The input clock to the SDHC peripheral is driven by the high speed system oscillator always, 150MHz. This 150MHz input clock is either divided by 2 (default) or by 4 to drive the SDHC peripheral. Set the SDHC peripheral clock divisor using the *GCR_PCLKDIS1.sdhc* bit as shown:

Equation 27-1: SDHC Peripheral Clock

$$f_{SDHC\_CLK} = \frac{150MHz}{2^{GCR\_PCLKDIS1.\mathrm{sdhc}}}$$

27.3 Usage

Communication over the SD bus is based on command and data bit streams/blocks that are initiated by a start bit and terminated by a stop bit:

> **Command**: A command is a token that starts an operation and is sent by the SDHC to the card in the embedded card slot. A command is transferred serially using the *SDHC_CMD* pin.
>
> **Response**: A response is a token sent from the card to the SDHC in response to a previously received command and is transferred serially using the *SDHC_CMD* pin.
>
> **Data**: You can transfer data from the card to the SDHC or vice versa using the SDHC_DAT[3:0] pins.

Figure 27-2, *Figure 27-3*, and *Figure 27-4* show the basic types of SD operations as described in the Physical Layer Simplified Specification Version 6.00 from the SD Card Association.

Figure 27-2: SD Bus Protocol - No Response and No Data Operations

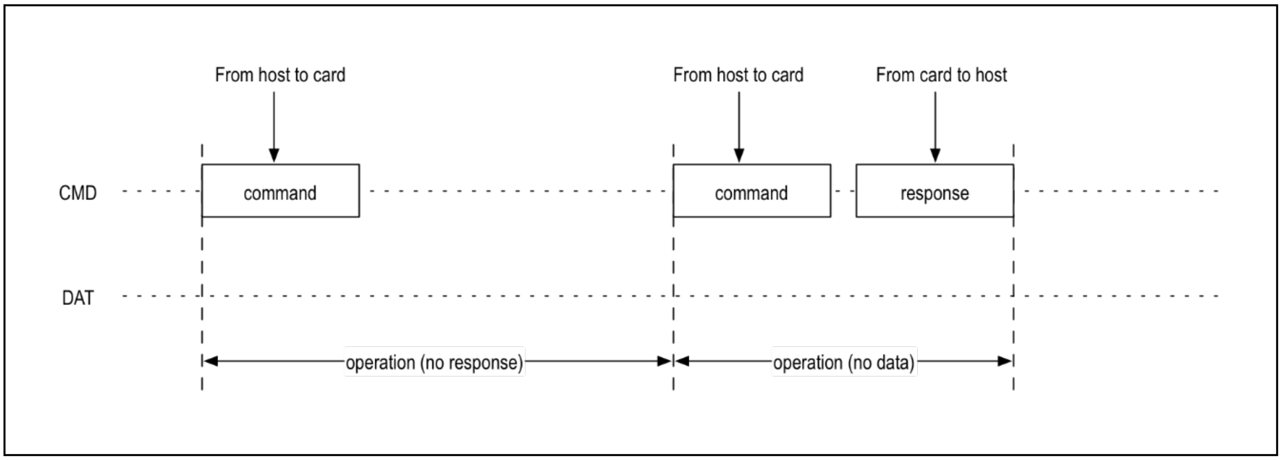


Figure 27-3: SD Bus Protocol - Multi-Block Read Operation

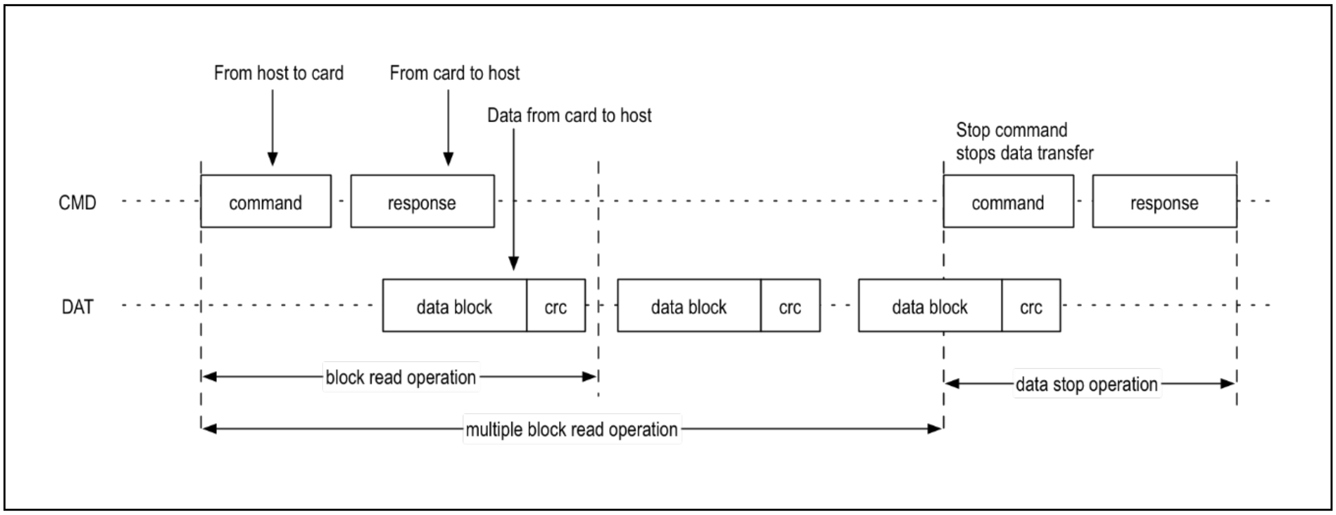
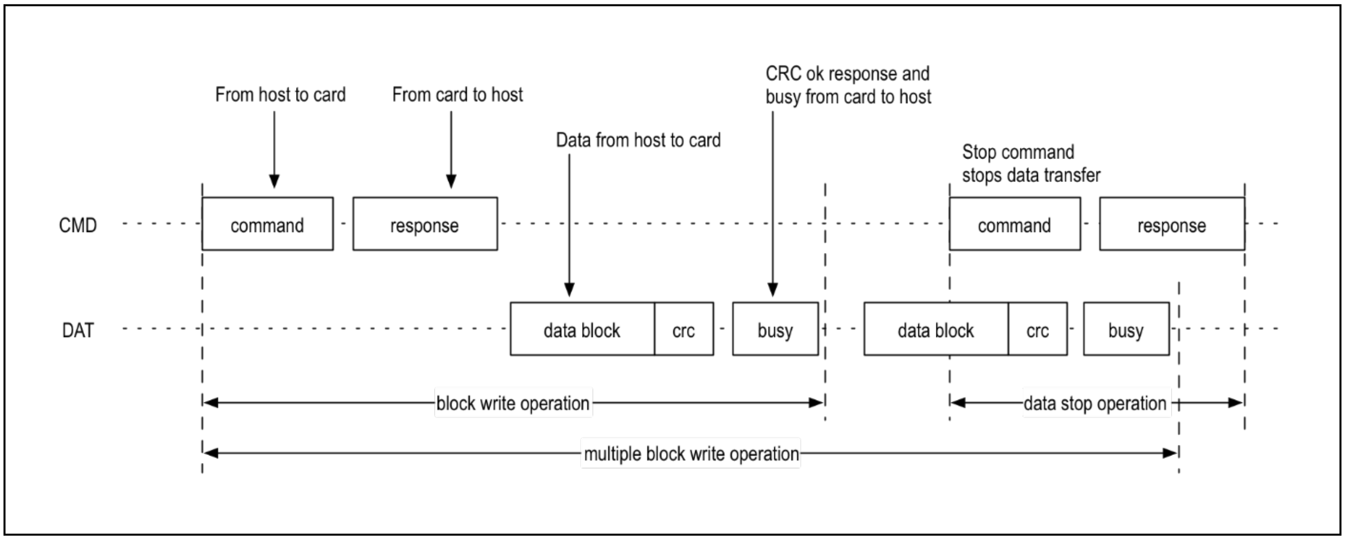


Figure 27-4: SD Bus Protocol - Multi Block Write Operation



27.4 SD Command Generation

Table 27-2 shows the registers required for three transaction types: SDMA generated transactions, ADMA generated transactions, and CPU transactions (includes data transfers and Non-DAT transfers). When initiating a transaction, you should program the registers sequentially starting with the *SDHC_SDMA* register and finishing with the *SDHC_CMD* register. When the upper byte of the *SDHC_CMD* register is written, it triggers the SDHC to issue the SD command.

Table 27-2: Registers Used to Generate SD Commands

Register	SDMA Command	ADMA Command	CPU Data Transfer	Non-DAT (No Data) Transfer
SDMA System Address / Argument 2 *SDHC_SDMA*	Yes/No	No/Auto CMD23	No/Auto CMD23	No/No
Block Size *SDHC_BLK_SIZE*	Yes	Yes	Yes	No (Protected)
Block Count *SDHC_BLK_CNT*	Yes	Yes	Yes	No (Protected)
Argument 2 *SDHC_SDMA*	Yes	Yes	Yes	No (Protected)
Command *SDHC_CMD*	Yes	Yes	Yes	Yes

27.5 Registers

See *Table 3-3* for the base address of this peripheral/module. See *Table 1-1* 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 27-3: SDHC Register Offsets, Names and Descriptions

Offset	Register Name	Description
[0x0000]	*SDHC_SDMA*	SDMA System Address / Argument 2
[0x0004]	*SDHC_BLK_SIZE*	Block Size register
[0x0006]	*SDHC_BLK_CNT*	Block Count register
[0x0008]	*SDHC_ARG_1*	Argument 1 register

Offset	Register Name	Description
[0x000C]	SDHC_TRANS	Transfer Mode register
[0x000E]	SDHC_CMD	Command register
[0x0010]	SDHC_RESP[0]	Response register 0
[0x0012]	SDHC_RESP[1]	Response register 1
[0x0014]	SDHC_RESP[2]	Response register 2
[0x0016]	SDHC_RESP[3]	Response register 3
[0x0018]	SDHC_RESP[4]	Response register 4
[0x001A]	SDHC_RESP[5]	Response register 5
[0x001C]	SDHC_RESP[6]	Response register 6
[0x001E]	SDHC_RESP[7]	Response register 7
[0x0020]	SDHC_BUFFER	Buffer Data Port register
[0x0024]	SDHC_PRESENT	Present State register
[0x0028]	SDHC_HOST_CN_1	Host Control 1 register
[0x0029]	SDHC_PWR	Power Control register
[0x002A]	SDHC_BLK_GAP	Block Gap Control register
[0x002B]	SDHC_WAKEUP	Wakeup Control register
[0x002C]	SDHC_CLK_CN	Clock Control register
[0x002E]	SDHC_TO	Timeout Control register
[0x002F]	SDHC_SW_RESET	Software Reset register
[0x0030]	SDHC_INT_STAT	Normal Interrupt Status register
[0x0032]	SDHC_ER_INT_STAT	Error Interrupt Status register
[0x0034]	SDHC_INT_EN	Normal Interrupt Status Enable register
[0x0036]	SDHC_ER_INT_EN	Error Interrupt Status Enable register
[0x0038]	SDHC_INT_SIGNAL	Normal Interrupt Signal Enable register
[0x003A]	SDHC_ER_INT_SIGNAL	Error Interrupt Signal Enable register
[0x003C]	SDHC_AUTO_CMD_ER	Auto CMD Error Status register
[0x003E]	SDHC_HOST_CN_2	Host Control 2 register
[0x0040]	SDHC_CFG_0	Capabilities register 0
[0x0044]	SDHC_CFG_1	Capabilities register 1
[0x0048]	SDHC_MAX_CURR_CFG	Maximum Current Capabilities register
[0x0050]	SDHC_FORCE_CMD	Force Event Register for Auto CMD Error Status
[0x0052]	SDHC_FORCE_EVENT_INT_STAT	Force Event Register for Error Interrupt Status
[0x0054]	SDHC_ADMA_ER	ADMA Error Status register
[0x0058]	SDHC_ADMA_ADDR_0	ADMA System Address register 0
[0x005C]	SDHC_ADMA_ADDR_1	ADMA System Address register 1
[0x0060]	SDHC_PRESET_0	Preset Value for Initialization
[0x0062]	SDHC_PRESET_1	Preset Value for Default Speed
[0x0064]	SDHC_PRESET_2	Preset Value for High Speed
[0x0066]	SDHC_PRESET_3	Preset Value for SDR12
[0x0068]	SDHC_PRESET_4	Preset Value for SDR25
[0x006A]	SDHC_PRESET_5	Preset Value for SDR50
[0x006C]	SDHC_PRESET_6	Preset Value for SDR104
[0x006E]	SDHC_PRESET_7	Preset Value for DDR50
[0x00FC]	SDHC_SLOT_INT	Slot Interrupt Status register

Offset	Register Name	Description
[0x00FE]	*SDHC_HOST_CN_VER*	Host Controller Version register

27.6 Register Details

Table 27-4: SDHC SDMA System Address / Argument Register

SDMA System Address / Argument 2 Register			SDHC_SDMA	[0x0000]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	addr	R/W	0	**SDMA System Address** This register is the address of the buffer used for a SDMA transfer. You must set this register to a valid data buffer address prior to starting an SDMA transfer. A SDHC DMA interrupt (*SDHC_INT_STAT*.dma = 1) is generated if the total size of the SDMA transfer exceeds the Host SDMA Buffer Size (*SDHC_BLK_SIZE.host_buff*). The card driver must update the SDMA System Address (*SDHC_SDMA*) with the address of the next data to transfer and clear the SDHC DMA interrupt flag prior to the transfer resuming. When the SDMA transfer is complete, this register contains the address of the next contiguous data address. When resuming a SDMA transfer, using the Resume command or by setting the *SDHC_BLK_GAP.cont* bit to 1, the SDHC resumes using the address in this register for the data to transfer. Reading this register during a SDMA transfer might return an invalid value unless the transfer is paused as the result of a SDHC DMA interrupt. This field is not used for ADMA transfers. **Argument 2** This register is used with Auto CMD23 to set a 32-bit block count value to the argument of CMD23 while executing Auto CMD23. If Auto CMD23 is used with ADMA, then the full 32-bit block count value is used. If Auto CMD23 is used without ADMA, the available block count value is limited by the *SDHC_BLK_GAP* register to 65,535 blocks.

Table 27-5: SDHC SDMA Block Size Register

SDMA Block Size Register			SDHC_BLK_SIZE	[0x0004]
Bits	**Name**	**Access**	**Reset**	**Description**
31:15	-	RO	0	**Reserved**

<table>
<tr><th colspan="3">SDMA Block Size Register</th><th>SDHC_BLK_SIZE</th><th>[0x0004]</th></tr>
<tr><th>Bits</th><th>Name</th><th>Access</th><th>Reset</th><th>Description</th></tr>
<tr><td>14:12</td><td>host_buff</td><td>R/W</td><td>0</td><td><b>Host SDMA Buffer Size</b>
This field specifies the size of the contiguous buffer in the system memory for SDMA transfers. SDMA transfers larger than this buffer generates a SDHC DMA interrupt (<i>SDHC_INT_STAT.dma</i>) when the transfer reaches the <i>host_buff</i> size boundary. The SDMA transfer pauses until the card driver updates the SDMA System Address (<i>SDHC_SDMA</i>) register with the next buffer address to transfer and clears the SDHC DMA interrupt flag. When the SDMA transfer is complete, a SDHC transfer complete interrupt (<i>SDHC_INT_STAT.trans_comp</i> = 1) is generated. The SDHC DMA interrupt flag is not set when the SDMA transfer completes.
<table>
<tr><th><i>host_buff</i> Value</th><th>Host SDMA Buffer Size (KB)</th></tr>
<tr><td>0b000</td><td>4</td></tr>
<tr><td>0b001</td><td>8</td></tr>
<tr><td>0b010</td><td>16</td></tr>
<tr><td>0b011</td><td>32</td></tr>
<tr><td>0b100</td><td>64</td></tr>
<tr><td>0b101</td><td>128</td></tr>
<tr><td>0b110</td><td>256</td></tr>
<tr><td>0b111</td><td>512</td></tr>
</table>
<i>Note: This field is used for SDMA transfers only.</i></td></tr>
<tr><td>11:0</td><td>trans</td><td>R/W</td><td>0x0200</td><td><b>Data Transfer Block Size</b>
Sets the block size of data transfers for CMD17, CMD18, CMD24, CMD25, and CMD53. You can set values ranging from 1 up to the maximum buffer size. Setting this field to 0 indicates there is no data to transfer.
During a transfer, reading this field might return an invalid value, and writes to this field are ignored.
<table>
<tr><th><i>trans</i> Value</th><th>Block Size in Bytes</th></tr>
<tr><td>0x0800</td><td>2,048</td></tr>
<tr><td>0x07FF</td><td>2,047</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>0x0200</td><td>512</td></tr>
<tr><td>0x01FF</td><td>511</td></tr>
<tr><td>...</td><td>...</td></tr>
<tr><td>0x0004</td><td>4</td></tr>
<tr><td>0x0003</td><td>3</td></tr>
<tr><td>0x0002</td><td>2</td></tr>
<tr><td>0x0001</td><td>1</td></tr>
<tr><td>0x0000</td><td>No data transfer</td></tr>
</table></td></tr>
</table>

Table 27-6: SDHC SDMA Block Count Register

<table>
<tr><th colspan="3">SDMA Block Count Register</th><th colspan="2">SDHC_BLK_CNT</th><th>[0x0006]</th></tr>
<tr><th>Bits</th><th>Name</th><th>Access</th><th>Reset</th><th colspan="2">Description</th></tr>
<tr><td>31:16</td><td>-</td><td>RO</td><td>0</td><td colspan="2"><b>Reserved</b></td></tr>
<tr><td>15:0</td><td>count</td><td>R/W</td><td>0x0200</td><td colspan="2"><b>Current Transfer Block Count</b>
Set to the total number of blocks to transfer prior to a block transfer operation. Set the Block Count Enable (<i>SDHC_TRANS.blk_cnt_en</i>) bit to 1 for a block transfer. If Block Count Enable is clear, then this field is unused.
When set to 1, the value in this register is the total number of blocks to transfer. After each block transfer, this register is decremented by 1, and stops when the count reaches 0.
Reads from this register are only valid when no transactions are active. A setting of 0 results in no blocks transferred.
When a Suspend command is complete, the number of remaining blocks to transfer is contained in this field.
Before issuing a Resume command, the card driver must restore the previously-saved block count to this field.
<table>
<tr><th><i>trans</i> Value</th><th>Block Count</th></tr>
<tr><td>0xFFFF</td><td>65,535</td></tr>
<tr><td>0xFFFE</td><td>65,534</td></tr>
<tr><td>….</td><td>….</td></tr>
<tr><td>0x0002</td><td>2</td></tr>
<tr><td>0x0001</td><td>1</td></tr>
<tr><td>0x0000</td><td>Stop count or no block transfer</td></tr>
</table>
</td></tr>
</table>

Table 27-7: SDHC SDMA Argument 1 Register

SDMA Argument 1 Register			SDHC_ARG_1	[0x0008]
Bits	**Name**	**Access**	**Reset**	**Description**
31:0	cmd	R/W	0	**SD Command Argument 1** The SD Command Argument 1 is specified as bit [39:8] of the Command-Format in the Physical Layer Specification.

Table 27-8: SDHC SDMA Transfer Mode Register

SDMA Transfer Mode Register			SDHC_TRANS	[0x000C]
Bits	**Name**	**Access**	**Reset**	**Description**
31:6	-	RO	0	**Reserved**
5	multi	R/W	0	**Multi/Single Block Select** Used for DAT line transfers and multiple-block commands. For all other commands, set this bit to 0. 1: Multiple-block or DAT line transfer. 0: Single Block. *Note: The SDHC_BLK_CNT register is ignored if this field is set to 0.*

SDMA Transfer Mode Register			SDHC_TRANS	[0x000C]
Bits	**Name**	**Access**	**Reset**	**Description**
4	read_write	R/W	0	**Data Transfer Direction Select** Sets the direction for DAT line data transfers. Set to 1 to transfer data from the SD card to the SDHC (Read). For all other commands, set this bit to 0 (Write). 1: Read (from card to host). 0: Write (from host to card).
3:2	auto_cmd_en	R/W	0	**Auto CMD Enable / Function Selection** 0b00: Auto Command Disabled. 0b01: Auto CMD12 Enable. 0b10: Auto CMD23 Enable. 0b11: Reserved. **Auto CMD12 Enable** When auto_cmd_en is set to 1, the SDHC issues CMD12 automatically after completion of the last block transfer. If an error occurs from Auto CMD12, then the error is saved to the *SDHC_AUTO_CMD_ER* register. *Note: Do not set to 1 if an Auto CMD12 is not required.* **Auto CMD23 Enable** When this bit field is set to 0b10, the Host Controller issues a CMD23 automatically before issuing the command specified in the *SDHC_CMD* (Command) register. The following conditions are required to use Auto CMD23: • Auto CMD23 support (Host Controller Version is 3.00 or later). • A memory card that supports CMD23 (SCR[33] = 1). • If using DMA, ADMA mode only. • Only when CMD18 or CMD25 is issued. You can use Auto CMD23 with or without ADMA. By writing to the Command register, the SDHC issues a CMD23 first, and then issues the command specified by the Command Index (*SDHC_CMD.idx*) in the Command register. If response errors are detected from CMD23, then the second command is not issued. A CMD23 error is indicated in the Auto CMD Error Status register (*SDHC_AUTO_CMD_ER*). The 32-bit block count value for CMD23 is set to the SDMA System Address / Argument 2 register (*SDHC_SDMA*). *Note: The SDHC does not check the command index.*
1	blk_cnt_en	R/W	0	**Block Count Enable** Set to enable the Block Count register (*SDHC_BLK_CNT*) for multiple block transfers. When this bit is 0, the Block Count register (*SDHC_BLK_CNT*) is disabled, which is useful if executing an infinite transfer. 1: Enable *SDHC_BLK_CNT* register. 0: Disable *SDHC_BLK_CNT* register.
0	dma_en	R/W	0	**DMA Enable** Enables DMA functionality per the Capabilities register. If this bit is set to 1, a DMA operation begins when the card driver writes to the upper byte of the Command register (*SDHC_CMD*). 1: DMA mode is enabled as specified in the *SDHC_HOST_CN_1.dma_select* field. 0: DMA mode disabled.

Table 27-9: Summary of how register settings determine type of data transfer

Multi/Single Block Select ***SDHC_TRANS.multi***	**Block Count Enable** ***SDHC_TRANS.blk_cnt_en***	**Block Count** ***SDHC_BLK_CNT.count***	**Function**
0	N.A.	N.A.	Single transfer
1	0	N.A.	Infinite transfer
1	1	≠0	Multiple transfer
1	1	0	Stop Multiple transfer

Table 27-10: SDHC Command Register

<table>
<tr><td colspan="4"><b>Command Register</b></td><td><b>SDHC_CMD</b>   <b>[0x000E]</b></td></tr>
<tr><th>Bits</th><th>Name</th><th>Access</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:14</td><td>-</td><td>RO</td><td>0</td><td><b>Reserved</b></td></tr>
<tr><td>13:8</td><td>idx</td><td>R/W</td><td>0</td><td><b>Command Index</b>
Valid command number (CMD0-63, ACMD0-63) per the SD Physical Specification and SDIO Card Specification.</td></tr>
<tr><td>7:6</td><td>type</td><td>R/W</td><td>0</td><td><b>Command Type</b>
The following table lists the values for this field, the type of command, and provides notes about what the command type is typically used for:
<table>
<tr><th><i>type</i> Value</th><th>Command Type</th><th>Notes</th></tr>
<tr><td>0b11</td><td>Abort</td><td>CMD12, CMD52 for writing I/O Abort in CCCR.</td></tr>
<tr><td>0b10</td><td>Resume</td><td>CMD52 for writing Function Select in CCCR.</td></tr>
<tr><td>0b01</td><td>Suspend</td><td>CMD52 for writing Bus Suspend in CCCR.</td></tr>
<tr><td>0b00</td><td>Normal</td><td>Other commands.</td></tr>
</table></td></tr>
<tr><td>5</td><td>data_pres_sel</td><td>R/W</td><td>0</td><td><b>Data Present Select</b>
1: Set to indicate data is present and transferable using the DAT line.
0: Commands that only use the CMD line (for example, CMD52), commands with no data transfer but are using the busy signal on SDHC_DAT[0], or a Resume command.</td></tr>
<tr><td>4</td><td>idx_chk_en</td><td>R/W</td><td>0</td><td><b>Command Index Check Enable</b>
1: SDHC checks the index field in the response and sets a Command Index Error if it does not match the value in the <i>SDHC_CMD.idx</i> field.
0: Index of response is not checked.</td></tr>
<tr><td>3</td><td>crc_chk_en</td><td>R/W</td><td>0</td><td><b>Command CRC Check Enable</b>
1: SDHC verifies the CRC field in the response, and if an error is detected, it is reported as a Command CRC Error.
0: CRC not checked by hardware.</td></tr>
<tr><td>2</td><td>-</td><td>RO</td><td>0</td><td><b>Reserved</b></td></tr>
<tr><td>1:0</td><td>resp_type</td><td>R/W</td><td>0</td><td><b>Response Type Select</b>
0b00: No Response.
0b01: Response Length 136.
0b10: Response Length 48.
0b11: Response Length 48, and check if busy after response.</td></tr>
</table>

Table 27-11: Relationship between Parameters and the Name of Response Type

Response Type ***SDHC_CMD.resp_type***	**Index Check Enable** ***SDHC_CMD.idx_chk_en***	**CRC Check Enable** ***SDHC_CMD.crc_chk_en***	**Name of Response Type**
0b00	0	0	No Response

Response Type *SDHC_CMD.resp_type*	Index Check Enable *SDHC_CMD.idx_chk_en*	CRC Check Enable *SDHC_CMD.crc_chk_en*	Name of Response Type
0b01	0	1	R2
0b10	0	0	R3, R4
0b10	1	1	R1, R5, R6, R7
0b11	1	1	R1b, R5b

Table 27-12: SDHC Response 0 Register

Response 0 Register			SDHC_RESP[0]	[0x0010]
Bits	**Name**	**Access**	**Reset**	**Description**
15:0	cmd_resp	RO	0	**Response Register 0** Response 7 to Response 0 registers are referenced as a contiguous, single register in the SD Host Controller Spec V3.0. *Table 27-20* shows the mapping from the Response Registers to the SD Host Controller Standard Specification REP[127:0] notation. *Table 27-21* shows the SD types of response mapped to the response registers.

Table 27-13: SDHC Response 1 Register

Response 1 Register			SDHC_RESP[1]	[0x0012]
Bits	**Name**	**Access**	**Reset**	**Description**
15:0	cmd_resp	RO	0	**Response Register 1** Response 7 to Response 0 registers are referenced as a contiguous, single register in the SD Host Controller Spec V3.0. *Table 27-20* shows the mapping from the Response Registers to the SD Host Controller Standard Specification REP[127:0] notation. *Table 27-21* shows the SD types of response mapped to the response registers.

Table 27-14: SDHC Response 2 Register

Response 2 Register			SDHC_RESP[2]	[0x0014]
Bits	**Name**	**Access**	**Reset**	**Description**
15:0	cmd_resp	RO	0	**Response Register 2** Response 7 to Response 0 registers are referenced as a contiguous, single register in the SD Host Controller Spec V3.0. *Table 27-20* shows the mapping from the Response Registers to the SD Host Controller Standard Specification REP[127:0] notation. *Table 27-21* shows the SD types of response mapped to the response registers.

Table 27-15: SDHC Response 3 Register

Response 3 Register			SDHC_RESP[3]	[0x0016]
Bits	**Name**	**Access**	**Reset**	**Description**
15:0	cmd_resp	RO	0	**Response Register 3** Response 7 to Response 0 registers are referenced as a contiguous, single register in the SD Host Controller Spec V3.0. *Table 27-20* shows the mapping from the Response Registers to the SD Host Controller Standard Specification REP[127:0] notation. *Table 27-21* shows the SD types of response mapped to the response registers.

Table 27-16: SDHC Response 4 Register

Response 4 Register			**SDHC_RESP[4]**	**[0x0018]**
Bits	**Name**	**Access**	**Reset**	**Description**
15:0	cmd_resp	RO	0	**Response Register 4** Response 7 to Response 0 registers are referenced as a contiguous, single register in the SD Host Controller Spec V3.0. *Table 27-20* shows the mapping from the Response Registers to the SD Host Controller Standard Specification REP[127:0] notation. *Table 27-21* shows the SD types of response mapped to the response registers.

Table 27-17: SDHC Response 5 Register

Response 5 Register			**SDHC_RESP[5]**	**[0x001A]**
Bits	**Name**	**Access**	**Reset**	**Description**
15:0	cmd_resp	RO	0	**Response Register 5** Response 7 to Response 0 registers are referenced as a contiguous, single register in the SD Host Controller Spec V3.0. *Table 27-20* shows the mapping from the Response Registers to the SD Host Controller Standard Specification REP[127:0] notation. *Table 27-21* shows the SD types of response mapped to the response registers.

Table 27-18: SDHC Response 6 Register

Response 6 Register			**SDHC_RESP[6]**	**[0x001C]**
Bits	**Name**	**Access**	**Reset**	**Description**
15:0	cmd_resp	RO	0	**Response Register 6** Response 7 to Response 0 registers are referenced as a contiguous, single register in the SD Host Controller Spec V3.0. *Table 27-20* shows the mapping from the Response Registers to the SD Host Controller Standard Specification REP[127:0] notation. *Table 27-21* shows the SD types of response mapped to the response registers.

Table 27-19: SDHC Response 7 Register

Response 7 Register			**SDHC_RESP[7]**	**[0x001E]**
Bits	**Name**	**Access**	**Reset**	**Description**
15:0	cmd_resp	RO	0	**Response Register 7** Response 7 to Response 0 registers are referenced as a contiguous, single register in the SD Host Controller Spec V3.0. *Table 27-20* shows the mapping from the Response Registers to the SD Host Controller Standard Specification REP[127:0] notation. *Table 27-21* shows the SD types of response mapped to the response registers.

Table 27-20: SDHC Response Register Mapping to SD Host Controller Response Register Convention

Register	**Register Name**	**Register Offset**	**SDHC REP[] Bit Mapping**
SDHC_RESP[0]	Response 0	0x10	REP[15:0]
SDHC_RESP[1]	Response 1	0x12	REP[31:16]
SDHC_RESP[2]	Response 2	0x14	REP[47:32]
SDHC_RESP[3]	Response 3	0x16	REP[63:48]
SDHC_RESP[4]	Response 4	0x18	REP[79:64]
SDHC_RESP[5]	Response 5	0x1A	REP[95:80]
SDHC_RESP[6]	Response 6	0x1C	REP[111:96]
SDHC_RESP[7]	Response 7	0x1E	REP[127:112]

Table 27-21: Kind of SD Card Response Mapping to SDHC Response Registers

Kind of Response	Meaning of Response	REP[] Specification Mapping	SDHC Response Register MSW	SDHC Response Register LSW
R1, R1b (normal response)	Card Status	REP[31:0]	*SDHC_RESP[1]*	*SDHC_RESP[0]*
R1b (Auto CMD12 response)	Card Status for Auto CMD12	REP[127:96]	*SDHC_RESP[7]*	*SDHC_RESP[6]*
R1 (Auto CMD23 response)	Card Status for Auto CMD23	REP[127:96]	*SDHC_RESP[7]*	*SDHC_RESP[6]*
R2 (CID, CSD register)	CID or CSD reg. incl.	REP [119:0]	*SDHC_RESP[7]*	*SDHC_RESP[0]*
R3 (OCR register)	OCR register for memory	REP [31:0]	*SDHC_RESP[1]*	*SDHC_RESP[0]*
R4 (OCR register)	OCR register for I/O, etc.	REP [31:0]	*SDHC_RESP[1]*	*SDHC_RESP[0]*
R5, R5b	SDIO response	REP [31:0]	*SDHC_RESP[1]*	*SDHC_RESP[0]*
R6 (Published RCA response)	Newly published RCA[31:16], etc.	REP [31:0]	*SDHC_RESP[1]*	*SDHC_RESP[0]*

Table 27-22: SDHC Buffer Data Port Register

<table>
<tr><th colspan="3">Buffer Data Port Register</th><th colspan="1">SDHC_BUFFER</th><th>[0x0020]</th></tr>
<tr><th>Bits</th><th>Name</th><th>Access</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:0</td><td>data</td><td>R/W</td><td>0</td><td><b>Buffer Data</b>
Pointer to the SDHC internal data buffer.</td></tr>
</table>

Table 27-23: SDHC Present State Register

<table>
<tr><th colspan="3">Present State Register</th><th colspan="1">SDHC_PRESENT</th><th>[0x0024]</th></tr>
<tr><th>Bits</th><th>Name</th><th>Access</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:25</td><td>-</td><td>RO</td><td>0</td><td><b>Reserved</b></td></tr>
<tr><td>24</td><td>cmd_signal_level</td><td>RO</td><td>0</td><td><b>CMD Line Signal Level</b>
Indicates the CMD line level for error recovery and debugging.</td></tr>
<tr><td>23:20</td><td>dat_signal_level</td><td>RO</td><td>-</td><td><b>SDHC_DAT[3:0] Line Signal Level</b>
Indicates the DAT line level for error recovery and debugging. Use to detect the busy signal level as indicated on SDHC_DAT[0].</td></tr>
<tr><td>19</td><td>wp</td><td>RO</td><td>-</td><td><b>Write Protect Switch Pin Level</b>
The write protect switch is supported for memory and combo cards. This bit reflects the state of the SDHC_WP pin.
1: Write enabled (SDHC_WP = 1).
0: Write protected (SDHC_WP = 0).</td></tr>
<tr><td>18</td><td>card_detect</td><td>RO</td><td>-</td><td><b>Card Detect Pin Level</b>
This bit reflects the inverted state of the SDHC_CDN pin. Debouncing is not performed on this bit. When Card State Stable is set to 1, this bit might be valid, but is not guaranteed. To use this bit, the card driver must debounce the bit.
1: Card present (SDHC_CDN = 0).
0: No card present (SDHC_CDN = 1).</td></tr>
<tr><td>17</td><td>card_state</td><td>RO</td><td>-</td><td><b>Card State Stable</b>
Used for debugging only. If this bit reads 0, the SDHC_CDN pin level is not stable. If this bit reads 1, the SDHC_CDN pin level is stable.
1: No card or card inserted.
0: Reset or debouncing.
<i>Note: This bit is not valid unless the SDHC_PRESENT.card_inserted bit reads 1.</i></td></tr>
</table>

Present State Register			SDHC_PRESENT	[0x0024]
Bits	**Name**	**Access**	**Reset**	**Description**
16	card_inserted	RO	-	**Card Inserted** Indicates if a card is inserted. This signal is debounced by the SDHC hardware. A change in state from 0 to 1 on this bit generates an SDHC_IRQn with the *SDHC_INT_STAT.card_insertion* flag set. Conversely, a transition of this bit from a 1 to a 0 generates an SDHC_IRQn interrupt with the *SDHC_INT_STAT.card_removal* field set. 1: Card Inserted. 0: Reset, debouncing, or no card inserted.
15:12	-	RO	0	**Reserved**
11	buffer_read	RO	0	**Buffer Read Status** If this bit reads 1, then data is available in the buffer for non-DMA transfers. This bit is cleared when all available block data is read from the buffer. This bit transitions from 0 to 1 when block data is ready in the buffer resulting in a SDHC_IRQn interrupt, if enabled, with the *SDHC_INT_STAT.buff_rd_ready* flag set. 1: Read data available. 0: No data to read.
10	buffer_write	RO	0	**Buffer Write Status** If this bit reads 1, then space is available in the buffer for write data. This bit is cleared when no space is available in the buffer. This bit transitions from a 0 to a 1 when top-of-block data is written to the buffer, resulting in a SDHC_IRQn interrupt, if enabled, with the *SDHC_INT_STAT.buff_wr_ready* flag set. 1: Space available in the buffer for write data. 0: No space available in the buffer for write data.
9	read_transfer	RO	0	**Read Transfer Active** Indicates completion of a read transfer. This bit is set to 1 for either of the following conditions: 1) After the end bit of a Read command. 2) When a read operation is restarted by setting the *SDHC_BLK_GAP.cont* bit (Continue Request). This bit is set to 0 for either of the following conditions: 1) The last data block as specified by the block length is transferred to the SDHC. 2) When all valid data blocks are transferred to the system, and no current block transfers are sent because the Stop At Block Gap Request register field is set to 1. A SDHC_IRQn interrupt is generated, if enabled. 1: Transferring data. 0: No valid data.

Present State Register			**SDHC_PRESENT**	**[0x0024]**
Bits	**Name**	**Access**	**Reset**	**Description**
8	write_transfer	RO	0	**Write Transfer Active** This bit is set to 1 for either of the following conditions: 1) After the end bit of the Write command. 2) When a write operation is restarted by setting the *SDHC_BLK_GAP.cont* bit to 1. This bit is cleared to 0 for either of the following conditions: 1) After getting the CRC status of the last data block transfer as specified by the transfer count, single and multiple block, *SDHC_BLK_CNT* register. 2) After getting the CRC status of any block where data transmission is stopped by a Stop At Block Gap Request (*SDHC_BLK_GAP.stop*). When *SDHC_BLK_GAP.stop* (stop at block gap request) is set, a change in *write_transfer* from 1 to 0 causes an SDHC_IRQn interrupt, if enabled, with the *SDHC_INT_STAT.blk_gap_event* flag set to 1. The *blk_gap_event* field indicates to the card driver that a non-DAT command can be issued during an active write. 1: Transferring data. 0: No valid data for transfer.
7:4	-	RO	0	**Reserved**
3	retuning	RO	0	**Re-Tuning Request** If this field reads 1, a retuning request was received from the external device. 0: Re-tuning request has not been received. 1: Re-tuning request received.
2	dat_line_active	RO	0	**DAT Line Active** A value of 1 indicates one or more DAT lines (SDHC_DAT[3:0]) are in use on the SD Bus. 0: No SD Bus DAT lines in use. 1: One or more DAT lines are in use.
1	dat	RO	0	**Command Inhibit (DAT)** This bit is set if DAT Line Active or the Read Transfer Active bits are set. A SDHC_IRQn interrupt is generated, if enabled, when this bit transitions from a 1 to a 0 with the *SDHC_INT_STAT.trans_comp* flag set. The card driver can save registers in the range of 0x000 to 0x00D for a suspend transaction after the *SDHC_INT_STAT.trans_comp* interrupt event. 1: Command that uses DAT line cannot be issued. 0: Command that uses DAT line can be issued.
0	cmd	RO	0	**Command Inhibit (CMD)** If this bit reads 0, the CMD line is not in use. This bit is set to 1 by the SDHC immediately after the *SDHC_CMD* register is written, and the bit is cleared to 0 when the Command Response is received. Auto CMD12 and Auto CMD23 consist of two responses, and this bit is not cleared until the read/write portion of the sequence is complete. 1: Command cannot be issued. 0: Can issue command using only CMD line.

Table 27-24: SDHC Host Control 1 Register

Host Control 1 Register			SDHC_HOST_CN_1	[0x0028]
Bits	**Name**	**Access**	**Reset**	**Description**
7	card_detect_signal	R/W	0	**Card Detect Signal Selection** 1: The Card Detect Test Level is selected (for test purposes). 0: SDHC_CDN is used for card detection (normal operation). *Note: Disable the Card Detect Interrupt when changing this bit.*
6	card_detect_test	R/W	-	**Card Detect Test Level** This bit is enabled when the Card Detect Signal Selection, *SDHC_HOST_CN_1.card_detect_signal*, field is set to 1. 1: Card Inserted. 0: No card inserted.
5	ext_data_transfer_width	R/W	0	**Extended Data Transfer Width** Extended data transfer width is not supported on the MAX78002. Always reads 0. 0: Bus width is selected by *SDHC_HOST_CN_1.data_transfer_width* field.
4:3	dma_select	R/W	0	**DMA Select** Sets the DMA mode. 0b00: SDMA mode. 0b01: Reserved. 0b10: 32-bit address ADMA2 mode. 0b11: Reserved.
2	hs_en	R/W	0	**High Speed Enable** 1: High-speed mode. 0: Normal-speed mode.
1	data_transfer_width	R/W	0	**Data Transfer Width** Sets the data transfer width of the SDHC. 1: 4-bit mode. 0: 1-bit mode.
0	led_cn	R/W	0	**LED Control** 1: LED on. 0: LED off.

Table 27-25: SDHC Power Control Register

Power Control Register			SDHC_PWR	[0x0029]
Bits	**Name**	**Access**	**Reset**	**Description**
7:4	-	RO	0	**Reserved**
3:1	bus_volt_sel	R/W	6	**SD Bus Voltage Select** Sets the voltage level for the SD card. Validate the setting against the Capabilities Register (*SDHC_CFG_0*). 0 - 4: Reserved. 5: 1.8V typical. 6: 3.0V typical. 7: 3.3V typical.

Power Control Register		SDHC_PWR		[0x0029]
Bits	**Name**	**Access**	**Reset**	**Description**
0	bus_power	R/W	0	**SD Bus Power** Before setting this bit, configure the *SDHC_PWR.bus_volt_sel* field. If no card is detected, then this bit is automatically set to 0 by the SDHC. 0: Power disabled. 1: Power enabled.

Table 27-26: SDHC Block Gap Control Register

Block Gap Control Register		SDHC_BLK_GAP		[0x002A]
Bits	**Name**	**Access**	**Reset**	**Description**
7:4	-	RO	0	**Reserved**
3	intr	R/W	0	**Interrupt at Block Gap** Setting this bit to 1 enables interrupt detection at the block gap for a multiple block transfer. 1: Enabled. 0: Disabled. *Note: This bit is only valid if SDHC_HOST_CN_1.data_transfer_width=1 (4-bit mode).*
2	read_wait	R/W	0	**Read Wait Control** If the card supports read wait (optional for SDIO cards), setting this bit enables use of the read wait protocol to stop reading data using the SDHC_DAT[2] line. If the card does not support read wait, the SDHC stops the SD Clock to hold read data, preventing command generation. When a card is inserted, the card driver must set this field based on the CCCR of the SDIO card inserted. Suspend/Resume is not supported when this bit is set to 0. 1: Enable Read Wait Control.. 0: Disable Read Wait Control *Note: If the SDIO card does not support read wait, then you must not set this bit to 1. Setting it to 1 when read wait is not supported might cause a SDHC_DAT line conflict.*
1	cont	R/W	0	**Continue Request** This bit is used to restart a transaction that was stopped using the Stop At Block Gap Request (*SDHC_BLK_GAP.stop*). To cancel a stop at the block gap, set *SDHC_BLK_GAP.stop* to 0, and set this bit, *SDHC_BLK_GAP.cont*, to 1 to restart the transfer. This bit is automatically cleared by hardware for either of the following conditions: • During a read transaction, the DAT Line Active changes from 0 to 1 as the write transaction restarts. • During a write transaction, the Write Transfer Active changes from 0 to 1 as the write transaction restarts. 1: Restart. 0: No effect.

Block Gap Control Register		SDHC_BLK_GAP		[0x002A]
Bits	**Name**	**Access**	**Reset**	**Description**
0	stop	R/W	0	**Stop At Block Gap Request** Setting this bit stops executing read and write transactions at the next block gap for non-DMA, SDMA, and ADMA transfers. This bit must remain set to 1 until the SDHC_INT_STAT.*trans_comp* bit is set to 1. For write transfers where the card driver writes data to the Buffer Data Port Register (SDHC_BUFFER), the card driver must set this bit after all block data is written. 1: Stop. 0: Transfer. This bit affects the following fields: • Read Transfer Active, SDHC_PRESENT.*read_transfer.* • Write Transfer Active, SDHC_PRESENT.*write_transfer.* • SDHC_DAT Line Active, SDHC_PRESENT.*dat_line_active.* • Command Inhibit (DAT), SDHC_PRESENT.*dat.* *Note: If this bit is set to 1, the card driver must not write data to the Buffer Data Port Register (SDHC_BUFFER).* *Note: Clearing both the SDHC_BLK_GAP.stop and SDHC_BLK_GAP.cont fields does not cause a transaction to restart.* *Note: You can set this bit to 1 regardless of whether the card inserted supports Read Wait Control. The SDHC stops the card through Read Wait Control or by stopping the SD clock.*

Table 27-27: SDHC Wakeup Control Register

Wakeup Control Register		SDHC_WAKEUP		[0x002B]
Bits	**Name**	**Access**	**Reset**	**Description**
7:3	-	RO	0	**Reserved**
2	card_rem	R/W	0	**Wakeup Event Enable on SD Card Removal** Enable wakeup event interrupt when the SDHC_INT_STAT.*card_removal* flag occurs. 1: Enable Interrupt. 0: Disable Interrupt.
1	card_ins	R/W	0	**Wakeup Event Enable on SD Card Insertion** Enable wakeup event interrupt when the SDHC_INT_STAT.*card_insertion* flag occurs. 1: Enable Interrupt. 0: Disable Interrupt.
0	card_int	R/W	0	**Wakeup Event Enable On Card Interrupt** Enable wakeup event interrupt when the SDHC_INT_STAT.*card_intr* flag occurs.

Table 27-28: SDHC Clock Control Register

<table>
<tr><td colspan="4"><b>Clock Control Register</b></td><td><b>SDHC_CLK_CN</b> <b>[0x002C]</b></td></tr>
<tr><th>Bits</th><th>Name</th><th>Access</th><th>Reset</th><th>Description</th></tr>
<tr><td>15:8</td><td>sdclk_freq_sel</td><td>R/W</td><td>0</td><td><b>SDCLK Frequency Select</b>
Selects the SD Clock Frequency output on the SDHC_CLK pin.
The SD Clock Frequency Select is a total of 10 bits. The divisors shown below consist of the *upper_sdclk_freq_sel* bits as bits 9:8, and the *sdclk_freq_sel* bits as bits 7:0 of the divisor.
<table>
<tr><th>upper_sdclk_freq_sel</th><th>sdclk_freq_sel</th><th>SDCLK Divisor (N)</th></tr>
<tr><td>0b11</td><td>0b11111111</td><td>1023</td></tr>
<tr><td>0b11</td><td>0b00000000</td><td>768</td></tr>
<tr><td>…</td><td>…</td><td>…</td></tr>
<tr><td>0b10</td><td>0b01010101</td><td>597</td></tr>
<tr><td>…</td><td>…</td><td>…</td></tr>
<tr><td>…</td><td>…</td><td>N</td></tr>
<tr><td>…</td><td>…</td><td>…</td></tr>
<tr><td>0b00</td><td>0b00000010</td><td>2</td></tr>
<tr><td>0b00</td><td>0b00000001</td><td>1</td></tr>
<tr><td>0b00</td><td>0b00000000</td><td>0 (MAX)</td></tr>
</table>
Setting *upper_sdclk_freq_sel* and *sdclk_freq_sel* to 0 results in the maximum SDCLK frequency of $f_{SDHC\_CLK\_FRQ}$. All other settings for *upper_sdclk_freq_sel* and *sdclk_freq_sel* follow the equation below:
$$SDHC\_CLK = {f_{SDHC\_CLK\_FRQ}}\Big/{(2 \times N)}$$
Note: The SD Clock Enable must be disabled (SDHC_CLK_CN.sd_clk_en = 0) prior to modification of this field.</td></tr>
<tr><td>7:6</td><td>upper_sdclk_freq_sel</td><td>R/W</td><td>0</td><td><b>Upper Bits of SDCLK Frequency Select</b>
Bits 9 and 8 of the 10-bit SDCLK frequency select. See the *SDHC_CLK_CN.sdclk_freq_sel* field for details about the clock select calculation.
Note: The SD Clock Enable must be disabled (SDHC_CLK_CN.sd_clk_en = 0) prior to modification of this field.</td></tr>
<tr><td>5</td><td>clk_gen_sel</td><td>RO</td><td>0</td><td><b>Clock Generator Select</b>
Reads 0 indicating Divided Clock mode only for SD Clock Frequency generation.
0: Divided clock mode.</td></tr>
<tr><td>4:3</td><td>-</td><td>RO</td><td>0</td><td><b>Reserved</b></td></tr>
<tr><td>2</td><td>sd_clk_en</td><td>R/W</td><td>0</td><td><b>SD Clock Enable</b>
Enable/disable SD Clock generation.
1: Enable the SD Clock and output on the SDHC_CLK pin.
0: SD Clock is disabled.
Note: This bit is cleared by the SDHC if the card-inserted field in the Present State register is cleared.
Note: The internal_clk_en bit must be set to 1, and the internal_clk_stable bit must read 1 prior to setting this bit to 1.</td></tr>
</table>

Clock Control Register		SDHC_CLK_CN		[0x002C]
Bits	**Name**	**Access**	**Reset**	**Description**
1	internal_clk_stable	RO	0	**Internal Clock Stable** This bit is set to 1 when the internal clock is stable. *Note: The internal clock must be enabled (SDHC_CLK_CN.internal_clk_en = 1) before this field is used.*
0	internal_clk_en	R/W	0	**Internal Clock Enable** Enable the internal clock. *Note: This bit must be set, and the internal_clk_stable bit must read 1 prior to setting the SD Clock Enable (SDHC_CLK_CN.sd_clk_en) bit.* *Note: This bit is set to 0 by the SDHC if waiting for a wakeup interrupt.*

Table 27-29: SDHC Timeout Control Register

<table>
<tr><th colspan="2">Timeout Control Register</th><th colspan="2">SDHC_TO</th><th>[0x002E]</th></tr>
<tr><th>Bits</th><th>Name</th><th>Access</th><th>Reset</th><th>Description</th></tr>
<tr><td>7:4</td><td>-</td><td>RO</td><td>0</td><td><b>Reserved</b></td></tr>
<tr><td>3:0</td><td>data_count_value</td><td>R/W</td><td>0</td><td><b>Data Timeout Counter Value</b>
Determines the interval for DAT line timeout detection. The timeout clock frequency is generated by dividing PCLK by the value calculated using this register. See Capabilities 0 Register (SDHC_CFG_0) for the definition of TMCLK.
The calculation for Data Timeout is shown in the following equation:
$$Data\,Timeout = TMCLK \times 2^{(13+data\_count\_value)}$$
<table>
<tr><th>Setting</th><th>Data Timeout</th></tr>
<tr><td>0b1111</td><td>Reserved</td></tr>
<tr><td>0b1110</td><td>$TMCLK \times 2^{(27)}$</td></tr>
<tr><td>0b1101</td><td>$TMCLK \times 2^{(26)}$</td></tr>
<tr><td>…</td><td>…</td></tr>
<tr><td>0b0010</td><td>$TMCLK \times 2^{(15)}$</td></tr>
<tr><td>0b0001</td><td>$TMCLK \times 2^{(14)}$</td></tr>
<tr><td>0b0000</td><td>$TMCLK \times 2^{(13)}$</td></tr>
</table>
<i>Note: Disable the Data Timeout Error Status Enable in the Error Interrupt Status Enable register (SDHC_ER_INT_EN.data_to).</i></td></tr>
</table>

Table 27-30: SDHC Software Reset Register

Software Reset Register		SDHC_SW_RESET		[0x002F]
Bits	**Name**	**Access**	**Reset**	**Description**
7:3	-	RO	0	**Reserved**

<table>
<tr><th colspan="2">Software Reset Register</th><th colspan="2">SDHC_SW_RESET</th><th>[0x002F]</th></tr>
<tr><th>Bits</th><th>Name</th><th>Access</th><th>Reset</th><th>Description</th></tr>
<tr><td>2</td><td>reset_dat</td><td>R/WC</td><td>0</td><td><b>Software Reset for DAT Line</b>
1: Reset.
0: Ready.
The following registers and fields are cleared/initialized when this bit is set:
<table>
<tr><th>Register</th><th>Field</th></tr>
<tr><td><i>SDHC_BUFFER</i></td><td><i>data</i></td></tr>
<tr><td rowspan="7"><i>SDHC_PRESENT</i></td><td><i>buffer_read</i></td></tr>
<tr><td><i>buffer_write</i></td></tr>
<tr><td><i>read_transfer</i></td></tr>
<tr><td><i>write_transfer</i></td></tr>
<tr><td><i>dat_line_active</i></td></tr>
<tr><td><i>dat</i></td></tr>
<tr><td><i>cmd</i></td></tr>
<tr><td rowspan="2"><i>SDHC_BLK_GAP</i></td><td><i>cont</i></td></tr>
<tr><td><i>stop</i></td></tr>
<tr><td rowspan="5"><i>SDHC_INT_STAT</i></td><td><i>buff_rd_ready</i></td></tr>
<tr><td><i>buff_wr_ready</i></td></tr>
<tr><td><i>dma</i></td></tr>
<tr><td><i>blk_gap_event</i></td></tr>
<tr><td><i>trans_comp</i></td></tr>
</table>
<i>Note: After setting this bit to 1, the Card Driver must poll this bit until it reads 0 to determine reset completion.</i></td></tr>
<tr><td>1</td><td>reset_cmd</td><td>R/WC</td><td>0</td><td><b>Software Reset for CMD Line</b>
1: Reset.
0: Ready.
The following registers and fields are cleared by setting this bit.
<table>
<tr><th>Register</th><th>Field</th></tr>
<tr><td><i>SDHC_PRESENT</i></td><td><i>cmd</i></td></tr>
<tr><td><i>SDHC_INT_STAT</i></td><td><i>cmd_comp</i></td></tr>
</table>
<i>Note: After setting this bit to 1, the card driver must poll this bit for 0 to determine when the reset is complete.</i></td></tr>
<tr><td>0</td><td>reset_all</td><td>R/WC</td><td>0</td><td><b>Software Reset for All</b>
Reset the SDHC except for the card detection interface. All registers are reset to their Reset/POR state.
1: Reset.
0: Ready.
<i>Note: After the Card Driver sets this bit to 1, the Card Driver should poll this bit until it reads 0 to determine when the SDHC completes the reset all request.</i></td></tr>
</table>

27.6.2 Normal Interrupt Status Register

The Normal Interrupt Status Enable affects reads of this register, but Normal Interrupt Signal Enable does not. An interrupt is generated when the Normal Interrupt Signal Enable is enabled, and at least one of the status bits is set to 1. W The Card

Interrupt (SDHC_INT_STAT.*card_intr*) is cleared when the card stops asserting the interrupt after the Card Driver services the interrupt condition.

Table 27-31: SDHC Normal Interrupt Status Register

Normal Interrupt Status Register		SDHC_INT_STAT		[0x0030]
Bits	**Name**	**Access**	**Reset**	**Description**
15	err_intr	ROC	0	**Error Interrupt** If any of the bits in the Error Interrupt Status register are set, then this bit is set. Therefore, the Host Driver can efficiently test for an error by checking this bit first. This bit is read only. 1: Error. 0: No Error.
14:13	-	RO	0	**Reserved**
12	retuning	ROC	0	**Re-Tuning Event** This status is set if the Re-Tuning Request bit in the Present State register changes from 0 to 1. The SDHC requests the Host Driver to perform re-tuning for the next data transfer. However, you can complete the current data transfer (not large block count) without re-tuning. 1: Perform re-tuning before the next data transfer. 0: Re-tuning is not required.
11:9	-	RO	0	**Reserved**
8	card_intr	ROC	0	**Card Interrupt** In one-bit mode, the SDHC detects the Card Interrupt without the SD Clock to support wakeup. In four-bit mode, the card interrupt signal is sampled during the interrupt cycle resulting in a delay between the interrupt signal from the memory card and the interrupt signal to the host driver. 1: Generate Card Interrupt. 0: No Card Interrupt. *Note: Writing a 1 to this bit does not clear this bit. It is cleared by resetting the SDHC_INT_EN.card_int flag.*
7	card_removal	R/W1C	0	**Card Removal** Set if the Card Inserted field in the Present State register (SDHC_PRESENT.*card_inserted*) changes from 1 to 0. 1: Card removed. 0: Card state stable or hardware debouncing.
6	card_insertion	R/W1C	0	**Card Inserted** Set if the Card Inserted field in the Present State register (SDHC_PRESENT.*card_inserted*) changes from 0 to 1. 1: Card inserted. 0: Card state stable or hardware debouncing.
5	buff_rd_ready	R/W1C	0	**Buffer Read Ready** Set if the Buffer Read Enable field in the Present State register (SDHC_PRESENT.*buffer_read*) changes from 0 to 1. 1: Ready to read buffer. 0: Not ready to read buffer. *Note: This field is set to 1 for every CMD19 execution while performing a tuning procedure (SDHC_HOST_CN_2.excute = 1).*

Normal Interrupt Status Register			SDHC_INT_STAT	[0x0030]
Bits	**Name**	**Access**	**Reset**	**Description**
4	buff_wr_ready	R/W1C	0	**Buffer Write Ready** Set if the Buffer Write Enable field in the Present State register (*SDHC_PRESENT.buffer_write*) changes from 0 to 1. 1: Ready to write buffer. 0: Not ready to write buffer.
3	dma	R/W1C	0	**DMA Interrupt** Set when the SDHC encounters the DMA buffer boundary set in the *SDHC_BLK_SIZE*.trans field during a SDMA transfer. The Card Driver must update the *SDHC_SDMA* register with the address of the next block to transfer before the SDHC continues the transfer. 1: SDHC DMA Interrupt is generated. 0: No SDHC DMA Interrupt.
2	blk_gap_event	R/W1C	0	**Block Gap Event** If the Stop at Block Gap Request field is set in the Block Gap Control register (*SDHC_BLK_GAP*.stop), this bit is set when a read or write transaction is stopped at a block gap. If Stop at Block Gap Request is not set to 1, then this bit is not meaningless. 1: Transaction stopped at block gap. 0: No block gap event.
1	trans_comp	R/W1C	0	**Transfer Complete** Set when a read/write transfer and a command with busy is complete. This bit has higher priority than Data Timeout Error. If both bits are set to 1, execution of a command is complete. See *Table 27-32* for Transfer Complete and Data Timeout Error priority and meaning. 1: Command execution is complete. 0: Not complete. *Note: This field is not set while performing a tuning procedure (SDHC_HOST_CN_2.excute = 1).*
0	cmd_comp	R/W1C	0	**Command Complete** Set when the end bit of the command response is received. Auto CMD12 and Auto CMD23 consist of two responses. This flag is not set by the card's response to the CMD12 or CMD23, but by the card's response to the read or write command you send to complete the Auto CMD12 or Auto CMD23. See Command Inhibit (*SDHC_PRESENT.cmd*) for how to control this bit. *Table 27-33* illustrates the relationship between Command Complete and Command Timeout Error bits. If both bits are set, then the response was not received within 64 SD clock cycles. 1: Command execution is complete. 0: Not complete.

Table 27-32: Transfer Complete and Data Timeout Error Priority and Status

Transfer Complete ***SDHC_INT_STAT.trans_comp***	**Data Timeout Error** ***SDHC_ER_INT_STAT.data_to***	**Status**
0	0	Interrupted by another event.
0	1	Timeout occurred during transfer.
1	N/A	Command execution complete.

Table 27-33: Command Complete and Command Timeout Error Priority and Status

Transfer Complete ***SDHC_INT_STAT.cmd_comp***	**Data Time Error** ***SDHC_ER_INT_STAT.cmd_to***	**Status**
0	0	Interrupted by another event.
N/A	1	Response not received within 64 SD Clock cycles.
1	0	Response received.

27.6.3 Error Interrupt Status Register

The interrupts defined in this register are enabled by the corresponding fields in the Error Interrupt Status Enable (*SDHC_ER_INT_EN*) register. Setting any field in the *SDHC_ER_INT_SIGNAL* register enables SDHC error interrupt generation using the SDHC_IRQn. The interrupt occurs when any field in the *SDHC_ER_INT_STAT* register is set to 1.

Table 27-34: SDHC Error Interrupt Status Register

Error Interrupt Status Register		**SDHC_ER_INT_STAT**		**[0x0032]**
Bits	**Name**	**Access**	**Reset**	**Description**
15:13	-	RO	0	**Reserved**
12	dma	R/W1C	0	**DMA Error** Error in SDMA transaction 1: Error. 0: No error.
11:10	-	RO	0	**Reserved**
9	adma	R/W1C	0	**ADMA Error** Set when the SDHC detects an error during an ADMA data transfer. The state of the ADMA when the error occurs is saved in the ADMA Error Status (*SDHC_ADMA_ER*) register. This bit is also set if the SDHC detects invalid descriptor data. If the *SDHC_ADMA_ER* register indicates an ADMA Error State, then an invalid descriptor was detected. 1: Error. 0: No error.
8	auto_cmd_12	R/W1C	0	**Auto CMD Error** Auto CMD12 and Auto CMD23 use this error status. This bit is set when detecting that one of the bits D00 - D04 in the Auto CMD Error Status (*SDHC_AUTO_CMD_ER*) register changed from a 0 to a 1. 1: Error. 0: No error. *Note: For Auto CMD12, this bit is set to 1 not only when an error occurs in Auto CMD12, but also when Auto CMD12 is not executed due to a previous command error.*
7	current_limit	R/W1C	0	**Current Limit Error** Not supported.
6	data_end_bit	R/W1C	0	**Data End Bit Error** Set if a 0 is detected at the end bit position of read data that uses the DAT line or the end-bit position of the CRC status. 1: Error. 0: No error.

Error Interrupt Status Register		SDHC_ER_INT_STAT		[0x0032]
Bits	**Name**	**Access**	**Reset**	**Description**
5	data_crc	R/W1C	0	**Data CRC Error** Set when a CRC error is detected when receiving read data that uses the DAT line or when detecting a Write CRC status with a value other than 010. 1: Error. 0: No error.
4	data_to	R/W1C	0	**Data Timeout Error** Set for any of the following timeout conditions: • Busy timeout for R1b and R5b response types. See *Table 27-11* for more information about response types. • Busy timeout after Write CRC status • Write CRC status Timeout • Read Data Timeout 1: Error. 0: No error.
3	cmd_idx	R/W1C	0	**Command Index Error** Set if a Command Index error is detected in the Command Response. 1: Error. 0: No error.
2	cmd_end_bit	R/W1C	0	**Command End Bit Error** Set if the end bit of a Command Response is 0. 1: Error. 0: No error.
1	cmd_crc	R/W1C	Table	**Command CRC Error** Set for the following cases: 1) If a response is returned, and the Command Timeout Error is set to 0, then this error flag is set if a CRT error is detected in the Command Response. 2) The SDHC detects a CMD-line conflict by monitoring the *SDHC_CMD* line when a command is issued. The SDHC sets the Command Timeout Error flag if a CMD line conflict is detected. A CMD line conflict indicates the CMD line was driven to 1, and the SDHC detected a 0 on the CMD line on the next SDCLK. 1: Error. 0: No error.
	cmd_to	R/W1C	0	**Command Timeout Error** Set if there is not response within 64 SDCLK cycles from the end bit of a command. 1: Error. 0: No error. *Note: If both the SDHC_ER_INT_STAT.cmd_crc and SDHC_ER_INT_STAT.cmd_to flags are set, then the SDHC detected a CMD-line conflict. See SDHC_ER_INT_STAT.cmd_crc for more information about a CMD-line conflict.*

Table 27-35: SDHC Normal Interrupt Status Register

Normal Interrupt Status Enable Register		**SDHC_INT_EN**		**[0x0034]**
Bits	**Name**	**Access**	**Reset**	**Description**
15:13	-	RO	0	**Reserved**
12	retuning	R/W		**Re-Tuning Event Status Enable** 1: Enabled. 0: Disabled.
11:9	-	RO	0	**Reserved**
8	card_int	R/W	0	**Card Interrupt Status Enable** Set to enable card-interrupt detection. The Card Driver should clear this bit prior to servicing a card interrupt status event and re-enable this bit after all interrupts from the card are serviced. 1: Enabled. 0: Disabled.
7	card_removal	R/W	0	**Card Removal Status Enable** Set to enable card removal event. 1: Enabled. 0: Disabled.
6	card_insert	R/W	0	**Card Insertion Status Enable** Set to enable card insertion event. 1: Enabled. 0: Disabled.
5	buffer_rd	R/W	0	**Buffer Read Ready Status Enable** Set to enable Buffer Read Ready status. 1: Enabled. 0: Disabled.
4	buffer_wr	R/W	0	**Buffer Write Ready Status Enable** Set to enable Buffer Write Ready status. 1: Enabled. 0: Disabled.
3	dma	R/W	0	**DMA Interrupt Status Enable** Set to enable DMA status. 1: Enabled. 0: Disabled.
2	blk_gap	R/W	0	**Block Gap Event Status Enable** Set to enable Block Gap status. 1: Enabled. 0: Disabled.
1	trans_comp	R/W	0	**Transfer Complete Status Enable** Set to enable Transfer Complete status. 1: Enabled. 0: Disabled.
0	cmd_comp	R/W	0	**Command Complete Status Enable** Set to enable Command Complete status. 1: Enabled. 0: Disabled.

Table 27-36: SDHC Error Interrupt Status Enable Register

Error Interrupt Status Enable Register			**SDHC_ER_INT_EN**	**[0x0036]**
Bits	**Name**	**Access**	**Reset**	**Description**
15:13	-	RO	0	**Reserved**
12	vendor	R/W	0	**Target Response Error/Host Error Status Enable** Set to enable Target Response/Host Error status interrupts. 1: Enabled. 0: Disabled.
11	-	RO	0	**Reserved**
10	tuning	R/W	0	**Tuning Error Status Interrupt Enable** 1: Enabled. 0: Disabled.
9	adma	R/W	0	**ADMA Error Status Interrupt Enable** 1: Enabled. 0: Disabled.
8	auto_cmd	R/W	0	**Auto CMD12 Error Status Interrupt Enable** 1: Enabled. 0: Disabled.
7	-	RO	0	**Reserved**
6	data_end_bit	R/W	0	**Data End Bit Error Status Interrupt Enable** 1: Enabled. 0: Disabled.
5	data_crc	R/W	0	**Data CRC Error Status Interrupt Enable** 1: Enabled. 0: Disabled.
4	data_to	R/W	0	**Data Timeout Error Status Interrupt Enable** 1: Enabled. 0: Disabled.
3	cmd_idx	R/W	0	**Command Index Error Status Interrupt Enable** 1: Enabled. 0: Disabled.
2	cmd_end_bit	R/W	0	**Command End Bit Error Status Interrupt Enable** 1: Enabled. 0: Disabled.
1	cmd_crc	R/W	0	**Command CRC Error Status Interrupt Enable** 1: Enabled. 0: Disabled.
0	cmd_to	R/W	0	**Command Timeout Error Status Interrupt Enable** 1: Enabled. 0: Disabled.

Table 27-37: SDHC Normal Interrupt Signal Enable Register

Normal Interrupt Signal Enable Register			**SDHC_INT_SIGNAL**	**[0x0038]**
Bits	**Name**	**Access**	**Reset**	**Description**
15:13	-	RO	0	**Reserved**

Normal Interrupt Signal Enable Register			**SDHC_INT_SIGNAL**	**[0x0038]**
Bits	**Name**	**Access**	**Reset**	**Description**
12	retuning	R/W	0	**Re-Tuning Event Signal Enable** 1: Enabled. 0: Disabled.
11:9	-	RO	0	**Reserved**
8	card_int	R/W	0	**Card Interrupt Signal Enable** 1: Enabled. 0: Disabled.
7	card_removal	R/W	0	**Card Removal Signal Enable** 1: Enabled. 0: Disabled.
6	card_insert	R/W	0	**Card Insertion Signal Enable** 1: Enabled. 0: Disabled.
5	buffer_rd	R/W	0	**Buffer Read Ready Signal Enable** 1: Enabled. 0: Disabled.
4	buffer_wr	R/W	0	**Buffer Write Ready Signal Enable** 1: Enabled. 0: Disabled.
3	dma	R/W	0	**DMA Interrupt Signal Enable** 1: Enabled. 0: Disabled.
2	blk_gap	R/W	0	**Block Gap Signal Enable** 1: Enabled. 0: Disabled.
1	trans_comp	R/W	0	**Transfer Complete Signal Enable** 1: Enabled. 0: Disabled.
0	cmd_comp	R/W	0	**Command Complete Signal Enable** 1: Enabled. 0: Disabled.

Table 27-38: SDHC Error Interrupt Signal Enable Register

Error Interrupt Signal Enable Register			**SDHC_ER_INT_SIGNAL**	**[0x003A]**
Bits	**Name**	**Access**	**Reset**	**Description**
15:13	-	RO	0	**Reserved**
12	tar_resp	R/W	0	**Target Response Error Signal Enable** 1: Enabled. 0: Disabled.
11	-	RO	0	**Reserved**
10	tuning	R/W	0	**Tuning Error Signal Enable** 1: Enabled. 0: Disabled.

Error Interrupt Signal Enable Register			SDHC_ER_INT_SIGNAL	[0x003A]
Bits	**Name**	**Access**	**Reset**	**Description**
9	adma	R/W	0	**ADMA Error Signal Enable** 1: Enabled. 0: Disabled.
8	auto_cmd	R/W	0	**Auto CMD12 Error Signal Enable** 1: Enabled. 0: Disabled.
7	curr_lim	R/W	0	**Current Limit Error Signal Enable** 1: Enabled. 0: Disabled.
6	data_end_bit	R/W	0	**Data End Bit Error Signal Enable** 1: Enabled. 0: Disabled.
5	data_crc	R/W	0	**Data CRC Error Signal Enable** 1: Enabled. 0: Disabled.
4	data_to	R/W	0	**Data Timeout Error Signal Enable** 1: Enabled. 0: Disabled.
3	cmd_idx	R/W	0	**Command Index Error Signal Enable** 1: Enabled. 0: Disabled.
2	cmd_end_bit	R/W	0	**Command End Bit Error Signal Enable** 1: Enabled. 0: Disabled.
1	cmd_crc	R/W	0	**Command CRC Error Signal Enable** 1: Enabled. 0: Disabled.
0	cmd_to	R/W	0	**Command Timeout Error Signal Enable** 1: Enabled. 0: Disabled.

27.6.4 Auto CMD Error Status Register

This register is used to indicate response errors for Auto CMD12 and Auto CMD23. The contents of this register are only valid when the Auto CMD Error is set (*SDHC_ER_INT_STAT.auto_cmd_12*). For Auto CMD23 errors, the error code is stored in *SDHC_AUTO_CMD_ER*[4:1].

Table 27-39: SDHC Auto CMD Error Status Register

Auto CMD Error Status Register			SDHC_AUTO_CMD_ER	[0x003C]
Bits	**Name**	**Access**	**Reset**	**Description**
15:8	-	RO	0	**Reserved**
7	not_issued	ROC	0	**Command Not Issued by Auto CMD12 Error** 1: Command not issued due to Auto CMD12 error as indicated in bits 4:1 of this register. 0: Auto CMD Error issued by Auto CMD23.
6:5	-	RO	0	**Reserved**

Auto CMD Error Status Register			SDHC_AUTO_CMD_ER	[0x003C]
Bits	**Name**	**Access**	**Reset**	**Description**
4	index	ROC	0	**Auto CMD Index Error** Command Index error occurred in response to a command. 1: Command Index Error. 0: No Error.
3	end_bit	ROC	0	**Auto CMD End Bit Error** Set if the end bit of the Command Response is 0. 1: End Bit Error. 0: No Error.
2	crc	ROC	0	**Auto CMD CRC Error** Set if CRC error in command response. 1: CRC Error. 0: No Error. *Note: If both SDHC_AUTO_CMD_ER.crc and SDHC_AUTO_CMD_ER.to are set, then a CMD-line conflict occurred.*
1	to	ROC	0	**Auto CMD Timeout Error** Set if no response is returned within 64 SDCLK cycles from the end bit of the command. If set, then ignore bits 4:2 of this register. 1: Timeout Error. 0: No Error. *Note: If both SDHC_AUTO_CMD_ER.crc and SDHC_AUTO_CMD_ER.to are set, then a CMD-line conflict occurred.*
0	not_excuted	ROC	0	**Auto CMD12 Not Executed Error** Auto CMD12 was not issued to stop a multi-block memory transfer due to an error with a prior command. 1: Not Executed. 0: No Error or error generated by Auto CMD23.

Table 27-40: SDHC Host Control 2 Register

Host Control 2 Register			SDHC_HOST_CN_2	[0x003E]
Bits	**Name**	**Access**	**Reset**	**Description**
15	preset_val_en	R/W	0	**Preset Value Enable** When set to 0, the following fields must be set by the Card Driver: • SDCLK Frequency Select (*SDHC_CLK_CN.sdclk_freq_sel*). • Clock Generator Select (*SDHC_CLK_CN.clk_gen_sel*). • Driver Strength Select (*SDHC_HOST_CN_2.driver_strength*). If set to 1, the Host Controller hardware sets the above fields based on the values in the Preset Value registers. 0: Card Driver must set the SDCLK Frequency Select, Clock Generator Select and Driver Strength Select fields. 1: The Host Controller hardware sets the above fields using the Preset Value register settings.
14	asynch_int	R/W	0	**Asynchronous Interrupt Enable** Always reads 0. Asynchronous Interrupt Enable is not supported by the MAX78002. Writes to this field have no effect.

Host Control 2 Register			SDHC_HOST_CN_2	[0x003E]
Bits	**Name**	**Access**	**Reset**	**Description**
13:8	-	RO	0	**Reserved**
7	sampling_clk	R/W	0	**Sampling Clock Select** This field is automatically set by hardware when Execute Tuning (*SDHC_HOST_CN_2.excute*) is cleared. 0: The fixed clock is used to sample data. 1: The tuned clock is used to sample data. *Note: The Card Driver cannot write 1 to this bit. Writing this bit to 0 can only be done if the Host Controller is not receiving a response or reading a data block.*
6	excute	R/WAC	0	**Execute Tuning** Setting this bit to 1 starts the tuning procedure and the bit is automatically cleared when the Host Controller completes the tuning procedure. Writing a 0 to this bit when it is set to 1 aborts the tuning procedure. 1: Execute tuning. 0: Tuning complete or not tuned.
5:4	driver_strength	R/W	0	**Driver Strength Select** If using 3.3V signaling, this field is ignored. For 1.8V signaling, the output driver strength is set using this field. If *SDHC_HOST_CN_2.preset_val_en* = 0, this field is controlled by the Host Driver. If *SDHC_HOST_CN_2.preset_val_en* = 1, this field is automatically set by the hardware using the Preset Value registers. 0: Driver Type B is selected. 1: Driver Type A is selected. 2: Driver Type C is selected. 3: Driver Type D is selected.
3	signal_v1_8	R/W	0	**1.8V Signaling Enable** If the card inserted supports UHS-I, this bit can be set to 1. No matter the value set, 3.3V is used for the card's supply. 1: 1.8V signaling. 0: 3.3V signaling.
2:0	uhs	R/W	0	**UHS Mode Select** Used to select the UHS-I mode. This field is only used if 1.8V signaling is set to 1 (*SDHC_HOST_CN_2.signal_v1_8* = 1). 0: SDR12. 1: SDR25. 2: SDR50. 3: SDR104 (Not supported). 4: DDR50. 5 – 7: Reserved.

Table 27-41: SDHC Capabilities Register 0

Capabilities Register 0			SDHC_CFG_0	[0x0040]
Bits	**Name**	**Access**	**Reset**	**Description**
31:30	slot_type	R	0	**Slot Type** 0: Support for a single slot with support for a removable card.
29	async_int	R	1	**Asynchronous Interrupt Support** 1: Asynchronous Interrupt Supported.

Capabilities Register 0			SDHC_CFG_0	[0x0040]
Bits	**Name**	**Access**	**Reset**	**Description**
28	bit_64_sys_bus	R	0	**64-bit System Bus Support** 0: 64-bit system bus not supported.
27	-	RO	0	**Reserved**
26	v1_8	R	1	**Voltage Support 1.8V** 1: 1.8V supported.
25	v3_0	R	1	**Voltage Support 3.0V** 1: 3.0V supported.
24	v3_3	R	1	**Voltage Support 3.3V** 1: 3.3V supported.
23	suspend	R	1	**Suspend/Resume Support** 1: Suspend / Resume functionality is supported.
22	sdma	R	1	**SDMA Support** SDMA is supported and can transfer data between system memory and the SDHC directly. 1: SDMA supported.
21	hs	R	1	**High Speed Support** The SDHC supports High Speed mode. 1: High speed mode supported.
20	-	RO	0	**Reserved**
19	adma2	R	1	**ADMA2 Support** The SDHC supports ADMA2. 1: ADMA2 supported.
18	bit_8	R	0	**8-bit Support for Embedded Device** The SDHC supports 8-bit bus width mode. 0: 8-bit Bus width not supported.
17:16	max_blk_len	R	0b10	**Max Block Length** This value indicates the maximum block size that the Host Driver can read and write to the buffer in the SDHC without wait cycles. The transfer block length is always 512 bytes for SD memory cards regardless of this field. 0b10: 2048 bytes.
15:8	clk_freq	R	0x00	**Base Clock Frequency for SD Clock** 0x00: Get information using another method.
7	to_clk_unit	R	1	**Timeout Clock Unit** 1: MHz base clock unit.
6	-	RO	0	**Reserved**
5:0	to_clk_freq	R	0x01	**Timeout Clock Frequency** The base clock frequency used to detect Data Timeout errors. The Timeout Clock Unit defines the units of this field's value. 1: 1 [MHz].

Table 27-42: SDHC Capabilities Register 1

Capabilities Register 1			**SDHC_CFG_1**	**[0x0044]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:24	-	RO	1	**Reserved**
23:16	clk_multi	R	0	**Clock Multiplier** Always reads 0x00. 0: Programmable clock generation is not supported.
15:14	retuning	R	0	**Re-Tuning Modes** Always reads 0b00. The SDHC supports Mode 1 Re-Tuning only with timer controlled by the host driver and a maximum of 4MB data length.
13	tuning_sdr50	R	0	**Use Tuning for SDR50** 1: Tuning required for SDR50. 0: SDR50 does not require tuning.
12	-	RO	0	**Reserved**
11:8	timer_cnt_tuning	R	0	**Timer Count for Re-Tuning** 0x0: Re-Tuning Timer disabled. 0x1: 1 second. 0x2: 2 seconds. 0x3: 4 seconds. 0x4: 8 seconds. ….: ……….. n: $2^{(n-1)}\ seconds.$ ….: ……….. 0xB: 1024 seconds. 0xC: Reserved. 0xD: Reserved. 0xE: Reserved. 0xF: Get information from another source.
7	-	RO	0	**Reserved**
6	driver_d	R	1	**Driver Type D Support** 1: Driver Type D is supported
5	driver_c	R	1	**Driver Type C Support** 1: Driver Type C is supported.
4	driver_a	R	1	**Driver Type A Support** 1: Driver Type A is supported.
3	-	RO	0	**Reserved**
2	ddr50	R	1	**DDR50 Support** 1: DDR50 is support.
1	sdr104	R	1	**SDR104** 1: SDR104 is supported.
0	sdr50	R	1	**SDR50** 1: SDR50 is supported.

Table 27-43: SDHC Maximum Current Capabilities Register

Maximum Current Capabilities Register			**SDHC_MAX_CURR_CFG**	**[0x0048]**
Bits	**Name**	**Access**	**Reset**	**Description**
31:24	-	RO	0	**Reserved**
23:16	v1_8	RO	0	**Maximum Current for 1.8V** 0x00: System dependent.
15:8	v3_0	RO	0	**Maximum Current for 3.0V** 0x00: System dependent.
7:0	v3_3	RO	0	**Maximum Current for 3.3V** 0x00: System dependent.

Table 27-44: SDHC Force Event Register for Auto CMD Error Status Register

Force Event Register for Auto CMD Error Status			**SDHC_FORCE_CMD**	**[0x0050]**
Bits	**Name**	**Access**	**Reset**	**Description**
15:8	-	RO	0	**Reserved**
7	not_issued	W	0	**Force Event for Command Not Issued By Auto CMD12 Error** 1: Interrupt is generated. 0: No interrupt generated.
6:5	-	RO	0	**Reserved**
4	index	W	0	**Force Event for Auto CMD Index Error** 1: Interrupt is generated. 0: No interrupt generated.
3	end_bit	W	0	**Force Event for Auto CMD End Bit Error** 1: Interrupt is generated. 0: No interrupt generated.
2	crc	W	0	**Force Event for Auto CMD CRC Error** 1: Interrupt is generated. 0: No interrupt generated.
1	to	W	0	**Force Event for Auto CMD Timeout Error** 1: Interrupt is generated. 0: No interrupt generated.
0	not_excu	W	0	**Force Event for Auto CMD12 Not Executed** 1: Interrupt is generated. 0: No interrupt generated.

Table 27-45: SDHC Force Event Register for Error Interrupt Status

Force Event Register for Error Interrupt Status			**SDHC_FORCE_EVENT_INT_STAT**	**[0x0052]**
Bits	**Name**	**Access**	**Reset**	**Description**
15:12	vendor	R/W	0	**Force Event for Vendor Specific Error Status** 1: Interrupt is generated. 0: No interrupt generated.
11:10	-	RO	0	**Reserved**
9	adma	R/W	0	**Force Event for ADMA Error** 1: Interrupt is generated. 0: No interrupt generated.

Force Event Register for Error Interrupt Status		SDHC_FORCE_EVENT_INT_STAT		[0x0052]
Bits	**Name**	**Access**	**Reset**	**Description**
8	auto_cmd	R/W	0	**Force Event for Auto CMD Error** 1: Interrupt is generated. 0: No interrupt generated.
7	curr_limit	R/W	0	**Force Event for Current Limit Error** 1: Interrupt is generated. 0: No interrupt generated.
6	data_end_bit	R/W	0	**Force Event for Data End Bit Error** 1: Interrupt is generated. 0: No interrupt generated.
5	data_crc	R/W	0	**Force Event for Data CRC Error** 1: Interrupt is generated. 0: No interrupt generated.
4	data_to	R/W	0	**Force Event for Data Timeout Error** 1: Interrupt is generated. 0: No interrupt generated.
3	cmd_index	R/W	0	**Force Event for Command Index Error** 1: Interrupt is generated. 0: No interrupt generated.
2	cmd_end_bit	R/W	0	**Force Event for Command End Bit Error** 1: Interrupt is generated. 0: No interrupt generated.
1	cmd_crc	R/W	0	**Force Event for Command CRC Error** 1: Interrupt is generated. 0: No interrupt generated.
0	cmd_to	R/W	0	**Force Event for Command Timeout Error** 1: Interrupt is generated. 0: No interrupt generated.

Table 27-46: SDHC ADMA Error Status Register

ADMA Error Status Register		SDHC_ADMA_ER		[0x0054]
Bits	**Name**	**Access**	**Reset**	**Description**
7:3	-	RO	0	**Reserved**
2	len_mismatch	ROC	0	**ADMA Length Mismatch Error** This error occurs in the following two cases: 1.) When setting Block Count Enable, the total data length specified by the Descriptor Table is different from that specified by the Block Count and Block Length fields. 2.) Total data length is not divisible by the Block Length field. 1: Error. 0: No Error.

<table>
<tr><td colspan="2"><b>ADMA Error Status Register</b></td><td colspan="2"><b>SDHC_ADMA_ER</b></td><td><b>[0x0054]</b></td></tr>
<tr><th>Bits</th><th>Name</th><th>Access</th><th>Reset</th><th>Description</th></tr>
<tr><td>1:0</td><td>state</td><td>ROC</td><td>0b00</td><td><b>ADMA Error State</b>
The state of the ADMA when the error condition occurred. Only valid during data transfer for ADMA.
The following table shows the possible state values, the associated ADMA Error State, and the contents of the <i>SDHC_SDMA</i> register.
<table>
<tr><th>state</th><th>ADMA Error State when the error occurred</th><th>SYS_SDR register contents</th></tr>
<tr><td>0b00</td><td>ST_STOP (Stop DMA)</td><td>Points next to the error descriptor.</td></tr>
<tr><td>0b01</td><td>ST_FDS (Fetch Descriptor)</td><td>Points to the error descriptor.</td></tr>
<tr><td>0b10</td><td>N/A</td><td>N/A</td></tr>
<tr><td>0b11</td><td>ST_TFR (Transfer Data)</td><td>Points next to the error descriptor.</td></tr>
</table>
<i>Note: 0b10 is not a valid error state and is never set.</i></td></tr>
</table>

Table 27-47: SDHC ADMA System Address Register 0

<table>
<tr><td colspan="2"><b>ADMA System Address Register 0</b></td><td colspan="2"><b>SDHC_ADMA_ADDR_0</b></td><td><b>[0x0058]</b></td></tr>
<tr><th>Bits</th><th>Name</th><th>Access</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:0</td><td>addr</td><td>R/W</td><td>0</td><td><b>ADMA System Address 0</b>
Holds the byte address of the executing command for the Descriptor Table. The Host Driver must set this address, made up of <i>SDHC_ADMA_ADDR_1</i>:<i>SDHC_ADMA_ADDR_0</i>, to the start address of the Descriptor Table. The ADMA increments this register address when fetching a descriptor line to point to the next address. When an ADMA Error Interrupt occurs, this register holds a valid descriptor address depending on the ADMA state. The following table shows the 64-bit System Address for ADMA using <<i>SDHC_ADMA_ADDR_1</i>>:<<i>SDHC_ADMA_ADDR_0</i>>.
<table>
<tr><th><i>SDHC_ADMA_ADDR_1</i></th><th><i>SDHC_ADMA_ADDR_0</i></th><th>64-Bit System Address</th></tr>
<tr><td>0x0000 0000</td><td>0x0000 0000</td><td>0x0000 0000 0000 0000</td></tr>
<tr><td>0x0000 0000</td><td>0x0000 0004</td><td>0x0000 0000 0000 0004</td></tr>
<tr><td>0x0000 0000</td><td>0x0000 0008</td><td>0x0000 0000 0000 0008</td></tr>
<tr><td>0x0000 0000</td><td>0x0000 000C</td><td>0x0000 0000 0000 000C</td></tr>
<tr><td>…</td><td>…</td><td>…</td></tr>
<tr><td>0xFFFF FFFF</td><td>0xFFFF FFFC</td><td>0xFFFF FFFF FFFF FFFC</td></tr>
</table>
<i>Note: The Host Driver must program the Descriptor Table on 32-bit boundaries and set the 32-bit boundary address to this register. ADMA2 ignores the lower two bits of this register, assuming it to be 0b00.</i></td></tr>
</table>

Table 27-48: SDHC ADMA System Address Register 1

<table>
<tr><td colspan="2"><b>ADMA System Address Register 1</b></td><td colspan="2"><b>SDHC_ADMA_ADDR_1</b></td><td><b>[0x005C]</b></td></tr>
<tr><th>Bits</th><th>Name</th><th>Access</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:0</td><td>addr</td><td>R/W</td><td>0</td><td><b>ADMA System Address 1</b>
Most-significant word for the 64-bit ADMA address. See <i>SDHC_ADMA_ADDR_0</i> for details.</td></tr>
</table>

27.6.5 Preset Value Registers

All preset value registers (*SDHC_PRESET_0* to *SDHC_PRESET_7*) contain the same fields as described in the *SDHC_PRESET_0* register. One of the Preset Value registers is automatically selected by the SDHC based on the selected bus-speed mode

Table 27-49 shows a group of preset values per card or device. One of the Preset Value registers (*SDHC_PRESET_1* – *SDHC_PRESET_7*) is selected by the SDHC hardware based on the Selected Bus Speed mode. *Table 27-50* defines the conditions to select one of the Preset Value registers.

Table 27-49: Preset Value Register Example

Offset	Preset Value Registers	Signal Voltage
[0x0060]	Preset Value for Initialization	3.3V or 1.8V
[0x0062]	Preset Value for Default Speed	3.3V
[0x0064]	Preset Value for High Speed	3.3V
[0x0066]	Preset Value for SDR12	1.8V
[0x0068]	Preset Value for SDR25	1.8V
[0x006A]	Preset Value for SDR50	1.8V
[0x006C]	Preset Value for SDR104	1.8V
[0x006E]	Preset Value for DDR50	1.8V

Table 27-50: Preset Value Register Selection Conditions

Selected Bus Speed Mode	1.8V Signaling Enable *SDHC_HOST_CN_2.signal_v1_8*	High Speed Enable *SDHC_HOST_CN_1.hs_en*	UHS-I Mode Selection *SDHC_HOST_CN_2.uhs*
Default Speed	0	0	N/A
High Speed	0	1	N/A
SDR12	1	N/A	0b000
SDR25	1	N/A	0b001
SDR50	1	N/A	0b010
SDR104	1	N/A	0b011
DDR50	1	N/A	0b100
Reserved	1	N/A	0b101 to 0b111

Table 27-51: SDHC Preset Value 0 to Preset Value 7 Registers

Preset Value 0 for Initialization		**SDHC_PRESET_0**		**[0x0060]**
Preset Value 1 for Initialization		**SDHC_PRESET_1**		**[0x0062]**
Preset Value 2 for Initialization		**SDHC_PRESET_2**		**[0x0064]**
Preset Value 3 for Initialization		**SDHC_PRESET_3**		**[0x0066]**
Preset Value 4 for Initialization		**SDHC_PRESET_4**		**[0x0068]**
Preset Value 5 for Initialization		**SDHC_PRESET_5**		**[0x006A]**
Preset Value 6 for Initialization		**SDHC_PRESET_6**		**[0x006C]**
Preset Value 7 for Initialization		**SDHC_PRESET_7**		**[0x006E]**
Bits	**Name**	**Access**	**Reset**	**Description**
15:14	driver_strength	RO	1	**Driver Strength Select Value** Driver strength is supported by 1.8V signaling bus speed modes. This field is not used for 3.3V signaling. 0b00: Driver Type B is selected. 0b01: Driver Type A is selected. 0b10: Driver Type C is selected. 0b11: Driver Type D is selected.
13:11	-	RO	0	**Reserved**
10	clk_gen	RO	0	**Clock Generator Select Value** 0: Programmable clock generator is not supported.
9:0	sdclk_freq	RO	-	**SDCLK Frequency Select Value** 10-bit preset value to set the SDCLK frequency select field in the clock control register (*SDHC_CLK_CN.upper_sdclk_freq_sel* and *SDHC_CLK_CN.sdclk_freq_sel*).

Table 27-52: SDHC Slot Interrupt Status Register

Slot Interrupt Status Register		**SDHC_SLOT_INT**		**[0x00FC]**
Bits	**Name**	**Access**	**Reset**	**Description**
15:1	-	RO	0	**Reserved**
0	int_signals	RO	0	**Interrupt Signals** Indicates the logical OR of Interrupt Signal and Wakeup Signal for the single slot. Only one slot is supported, slot 0. Reset by POR and by software reset for all (*SDHC_SW_RESET.reset_all*).

Table 27-53: SDHC Host Controller Version Register

Host Controller Version Register		**SDHC_HOST_CN_VER**		**[0x00FE]**
Bits	**Name**	**Access**	**Reset**	**Description**
15:8	vend_ver	RO	-	**Vendor Version** This status is reserved for the vendor version number. The Host Driver should not use this status.
7:0	spec_ver	RO	0x02	**Specification Version Number** This status indicates the Host Controller Specification Version. 0x02: SD Host Specification Version 3.00.

28. Convolutional Neural Network

The CNN accelerator consists of 64 parallel processors with 1.31MB 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 the data memory instance it shares with the three others.

The features of the CNN accelerator include:

- Data Storage
    - 1.31MB SRAM based data storage
    - Configured as 20Kx8 bit integers x64 channels or 80Kx8 bit integers x4 channels for input layers
    - Input Data Format - 8 bit signed values
    - Selectable Output Data Format - 8 bit signed integer or 32 bit signed integer
    - Arm AMBA APB accessible
    - Hardware CNN results data unload assist
- Weight Storage
    - SRAM based with selectable data retention mode
    - Configurable from 2M 8 bit integer weights to 16M 1 bit logical weights
    - Optional 4x processing mode splits each weight memory into 4 parallel memories with a common address generating 4x the number of masks each cycle
    - All processors include the following dedicated weight storage
        - 1x Processing Mode
            - 4096x9x8 bit weights
            - 8192x9x4 bit weights
            - 16384x9x2 bit weights
            - 32768x9x1 bit weights
        - 4x Processing Mode
            - 4x1024x9x8 bit weights
            - 4x2048x9x4 bit weights
            - 4x4096x9x2 bit weights
            - 4x8192x9x1 bit weights
    - The first processor in each x16 includes additional weight storage for input layer processing
        - 1x Processing Mode
            - 1024x9x8 bit weights
            - 2048x9x4 bit weights
            - 4096x9x2 bit weights
            - 8192x9x1 bit weights
        - 4x Processing Mode
            - 4x256x9x8 bit weights
            - 4x512x9x4 bit weights
            - 4x1024x9x2 bit weights
            - 4x2048x9x1 bit weights

- Programmable Per x16 processor weight RAM start address, start pointer, and mask count.
- Arm AMBA APB accessible
- Optional weight load hardware assist for packed weight storage

- 128 Independently configurable layers (Per x16 Processor)
  - Programmable start layer - any of the 128 layers
  - Linked layer mode allows arbitrary non-sequential layer execution
  - Configurable Per Layer Parameters
    - Processor and mask enables (16 channels)
    - Input data format - byte-wide input data or 4x8 bit wide input data (x16 processors 0, 4, 8, or 12 only)
    - Per layer data streaming
      - Up to Eight simultaneous streaming layers - available for the first eight layers
      - Optional FIFO input data paths (first layer only)
      - Selectable streaming termination layer - transition to non-stream processing mode
      - Programmable per stream configuration
        - Stream start - relative to prior stream
        - Three stream processing delay counters - 2 column counters for non-integer ratios, 1-row delta counter
    - Data SRAM circular buffer size
    - Programmable Input data size (separate Row, Column fields)\
    - Programmable Row and Column Padding - 0 to 3 bytes
    - Configurable Number of input channels 1 – 1024
    - Configurable Number of output channels 1 - 1024 (determined by the kernel count value)
    - Selectable Kernel bit width size (1, 2, 4, 8)
    - Selectable Kernel SRAM pointer start address and count
    - Optional In-flight Input Image Pooling
      - Pool Mode - None, Maximum or Average
      - Pool Size - 2x2 to 16x16 with independent row and column counts
      - Pool Dilation - 0 to 15
    - Programmable Stride - 1 to 4 common row/column stride value
    - Data SRAM read pointer base address
      - Configurable Read Pointer increment value for flexible input channel access
    - Data SRAM write pointer configuration
      - Base Address
      - Independent offsets for output channel storage in SRAM
      - Programmable stride increment offset
    - Bias - 8192 8 bit integers with an option for 1024 10 bit integers using multiple x16 processors
      - Optionally configurable as 4x2048x8 bit bias for 4x mode (with an option for 10-bit bias using multiple x16 processors)
    - Pre-activation output scaling - direction (up/down) and 0 to 15 bit shift magnitude
    - Output Activation - None, ReLU, Absolute Value
    - Passthru mode allows input data to be passed directly through to the output with programmable data relocation.

    - Element-wise operations (add, subtract, XOR, OR) with optional convolution - up to 16 elements
    - Deconvolution
    - Flattening - for MLP processing
    - Depthwise Convolution
    - Simple logic modes support single mask bit +1/-1, 0/-1 modes
    - No mask mode supports convolutions with a fixed mask value of one
- Processing
  - 64 parallel physical channel processors
    - Organized as 4 x 16 Processors
    - 8-bit integer data path with an option for 32-bit integers on the output layer
    - Per Channel Processor Enable/Disable
    - Expandable to 1024 parallel logical channel processors

  - Configurable 3x3 or 1x1 2D kernel size
  - Configurable 1D kernel size to 1x9
  - Full resolution sum-of-products arithmetic for 1024 8 bit integer (data and weight) channels
  - Two maximum operating frequency modes - up to 50MHz in non-pipelined mode or up to 200 MHz in pipeline mode
  - Up to 16 output channels per clock processing rate
  - Conditional execution allows early layer termination and branching based on the programmable address and/or data and/or count match
- Input Layer Maximum Input Size
  - 20K bytes, 64 channels, non-streaming, APB I/F
  - 80K bytes, 16 channels, non-streaming, APB I/F
  - 80K bytes, 4 channels, non-streaming, FIFO I/F
  - 2048x2048 bytes, 4 channels, streaming, FIFO I/F
- Hidden Layers Maximum Input Size
  - Up to 20K bytes per channel, x64 channels, non-streaming
  - 20K bytes can be split equally across 1-16 logical channels, non-streaming
  - 4M bytes per channel, x64 channels, streaming
  - 4M bytes can be split equally across 8 layers, streaming
- Optional Interrupt on CNN completion and FIFO full and empty statuses
- User accessible BIST on all internal memories
- User accessible Zeroization of all internal memories
- Single-step operation with full data SRAM access for CNN operation debug
- Power Management
  - Independent x16 processor supply enables
  - Independent x16 processor mask retention enables
  - Independent x16 data path clock enables
  - Functional APB clock gating with per x16 processor override - registers clocked only during APB write access.
  - CNN Clock frequency scaling (divide by 2, 4, 8, 16)
  - Chip level voltage control for power-performance optimization
- Input Data Row Buffer Memory (TRAM)
  - Organized as 12Kx16 or optionally as 4x3Kx16 in read-ahead mode
  - Programmable per layer TRAM read/write pointer start and rollover values
  - Automatically allocates memory based on the programmed number of input channels
- Read Ahead input processing mode allows the next input data byte to be pre-processed while the current input byte output channel generation is active.

a

29. 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 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 SBL.
- Checksum verification of ROM image before further ROM execution.
- SWD disabled in LOCKED and PERMLOCKED states.
- Programmable through UART or SWD interface.
- UART operates at 115,200bps.
- LOCKED mode disables SWD and disallows any change to flash through 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 SHA-256 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.

29.1 Instances

The dedicated pins and features of the bootloader are shown *Table 29-1*.

Table 29-1: MAX78002 Bootloader Instances

Part Number	Activation Pins		Bootloader	Secure Bootloader	Secure Boot	Flash Memory Page Size
	UART0 RX	SWDCLK				
MAX78002GXE+	UART0 RX	SWDCLK	Yes	No	No	16KB

29.2 Bootloader Operating States

Each bootloader supports the modes shown in *Table 29-2*. Each bootloader mode has a unique prompt.

Table 29-2: MAX78002 Bootloader Operating States and Prompts

State	Bootloader	Secure Bootloader	Recognized Commands	Prompt
UNLOCKED	Yes	Yes	All Commands U/L/P	“ULDR> “ <0x55> <0x4C> <0x44> <0x52> <0x3E> <0x20>
LOCKED	Yes	Yes	Only L/P	“LLDR> “ <0x4C> <0x4C> <0x44> <0x52> <0x3E> <0x20>

PERMLOCK	Yes	Yes	Only P	"PLLDR> " <0x50> <0x4C> <0x4C> <0x44> <0x52> <0x3E> <0x20>
CHALLENGE	No	Yes	*GC – Get Challenge* *SR – Send Response*	"<CR> " <0x43> <0x52> <0x3E> <0x20>
APPVERIFY	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.

29.2.1 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.

29.2.2 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.

29.2.3 PERMLOCKED

The PERMLOCKED state disables access to program memory other than to verify it with a simple SHA-256 hash. The commands available in the PERMLOCKED state are shown in *Table 29-3*.

Table 29-3: PERMLOCK Command Summary

Command
H – Check Device
I – Get ID

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.

29.2.4 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 SHA-256 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.

29.2.5 APPVERIFY (Secure Bootloader only)

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. The device will automatically verify program memory on all subsequent resets and attempts to execute the Lock and Plock commands.

29.3 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 the 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 SHA-256 using the secret key over the length of the padded binary image.
5. Append 32-byte HMAC SHA-256 to the binary image, after the last pad byte.
6. Generate Motorola SREC file.

29.4 Bootloader Activation

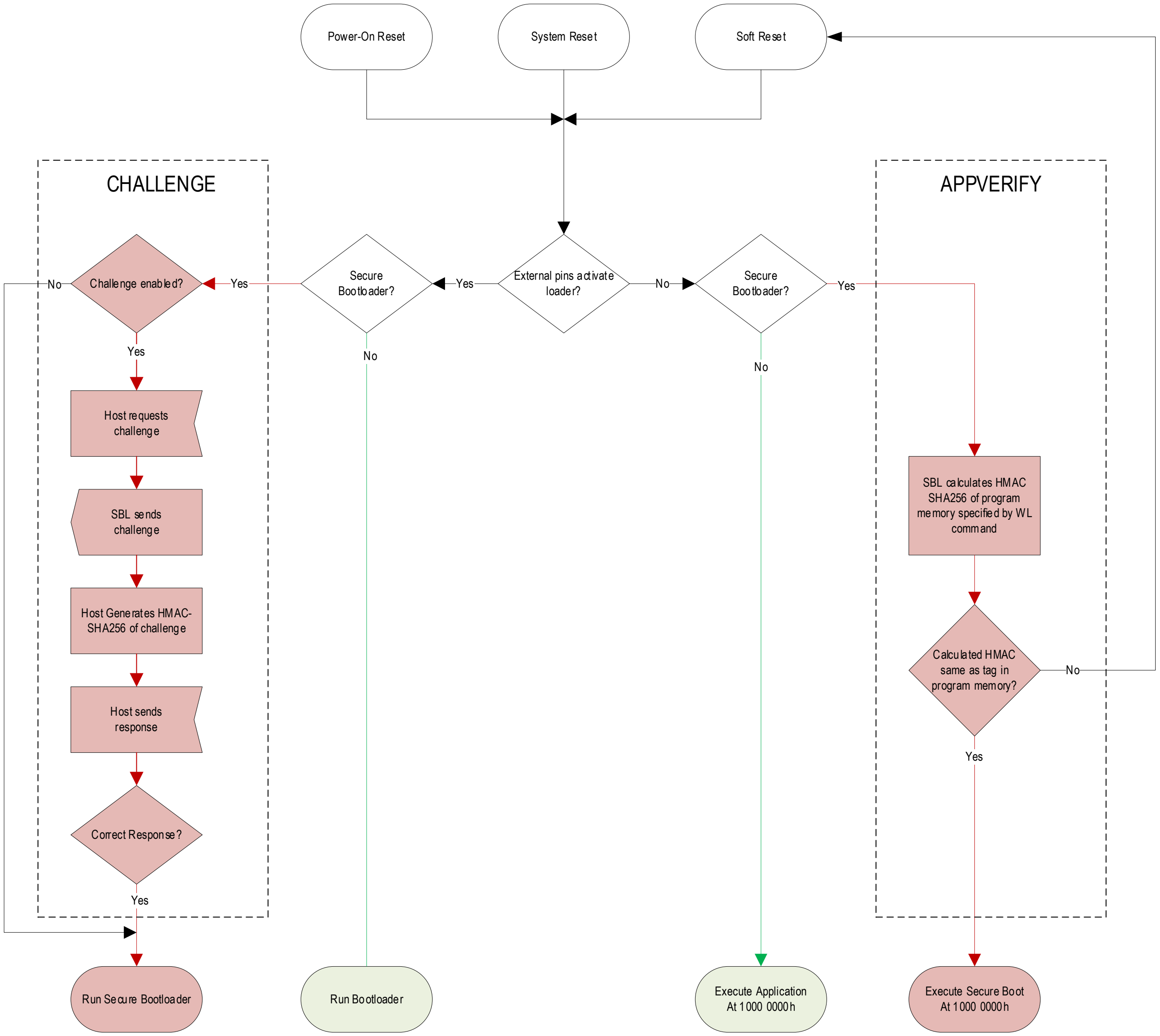
Perform the following sequence to activate the bootloader:

1. The host asserts the UART0 Rx pin and SWDCLK pins low as shown in *Table 29-1*.
2. The host asserts RSTN pin low.
3. The 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 115,200bps.
6. The bootloader outputs the status prompt on the UART0 Tx pin. The prompt is unique for each bootloader state as shown in *Table 29-2*.
7. The host releases the UART0 Rx and SWDCLK pins once the host confirms the correct bootloader prompt.
8. The host begins bootloader session by sending commands on the UART0 Rx pin.

29.5 Bootloader

The bootloader is invoked following a reset when the bootloader activation pin is asserted. The flow chart of the operation following a reset of the device is shown in *Figure 29-1*. Features exclusive to the SBL are highlighted in red.

Figure 29-1: MAX78002 Combined Bootloader Flow


Power-On Reset
System Reset
Soft Reset
CHALLENGE
APPVERIFY
Challenge enabled?
No
Yes
Secure Bootloader?
Yes
External pins activate loader?
No
Secure Bootloader?
Yes
Yes
No
No
Host requests challenge
SBL sends challenge
Host Generates HMAC-SHA256 of challenge
Host sends response
Correct Response?
Yes
SBL calculates HMAC SHA256 of program memory specified by WL command
Calculated HMAC same as tag in program memory?
No
Yes
Run Secure Bootloader
Run Bootloader
Execute Application At 1000 0000h
Execute Secure Boot At 1000 0000h

29.6 Secure Bootloader

The secure version of the bootloader provides additional features for secure and authenticated loading. These features are highlighted in *Figure 29-1*.

29.6.1 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.

29.6.2 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 29-2*.

Only two commands are available in the CHALLENGE state.

Table 29-4: 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 SHA-256 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. The host generates the challenge/response HMAC SHA-256 secret key.
2. The 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. The 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. The host executes the GC command.
2. The bootloader generates a 128-bit challenge and sends it to the host.
3. The 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.

29.7 Command Protocol

The bootloader presents a mode-specific prompt based on the current state of the loader as shown as in *Table 29-2*. 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.

29.8 General Commands

Table 29-5: MAX78002 General Command Summary

Command
L - Load
P – Page Erase
V – Verify
LOCK – Lock Device
PLOCK – Permanent Lock
UNLOCK – Unlock Device
H – Check Device
I – Get ID
S – Status
Q – Quit

29.8.1 General Command Details

Table 29-6: 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 is 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 29-7: 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 29-8: 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 29-9: LOCK – Lock Device

LOCK – Lock Device	Lock Device
Description	Locks the device and disables SWD on the next device reset. See *LOCKED* 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 *S – Status* 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 29-10: 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 *S – Status* 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 29-11: 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 29-12: 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	yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy<0x0D><0x0A>
Response: Failure	<0x0D><0x0A>

Table 29-13: 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 29-14: S – Status

S – Status	Read Device Status
Description	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: 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. 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. In addition, the SBL will display: 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. 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. 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.
Modes	U

Command	`S<0x0D><0x0A>`
	`Status<0x0D><0x0A>`
	`Lock: <response><0x0D><0x0A>`
	`PLock: <response><0x0D><0x0A>`
	`Application Length: <response><0x0D><0x0A>`
	`Application Key: <response><0x0D><0x0A>`
	`Challenge Key: <response><0x0D><0x0A>`

Response: Success	None.
Response: Failure	None.

Table 29-15: 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<0x0D><0x0A>`
Response: Success	None
Response: Failure	None

29.9 Secure Commands

Table 29-16: MAX78002 Secure Command Summary

Command
LK – Load Application Key
LC – Load Challenge Key
VK – Verify Application Key
VC – Verify Challenge Key
AK – Activate Application Key
AC – Activate Challenge
WL – Write Code Length

29.9.1 Secure Command Details

Table 29-17: LK – Load Application Key

LK – Load Application Key	Load Application HMAC SHA-256 Key
Description	Write 128-bit HMAC secret key to nonvolatile memory.
Modes	U
Command	`LK yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy<0x0D><0x0A>`
Response: Success	`OK<0x0D><0x0A>`

Response: Failure	`Bad key input<0x0D><0x0A>`
	or
	`Key already loaded<0x0D><0x0A>`

Table 29-18: LK – Load Challenge Key

LC – Load Challenge Key	Load Challenge Key
Description	Write 128-bit challenge key to nonvolatile memory.
Modes	U
Command	`LC yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy<0x0D><0x0A>`
Response: Success	`OK<0x0D><0x0A>`

Response: Failure	`Bad key input<0x0D><0x0A>`
	`or`
	`Key already loaded<0x0D><0x0A>`

Table 29-19: 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 yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy<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 29-20: 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 yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy<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 29-21: AK – Activate Application Key

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 29-22: 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 29-23: 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>`

29.10 Challenge/Response Commands

Table 29-24: MAX78002 Challenge/Response Command Summary

Command
GC – Get Challenge
SR – Send Response

29.10.1 Challenge/Response Command Details

Table 29-25: 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 29-26: 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>`

30. Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
a0	1/05/2022	Preliminary Release 0	-
a1	04/01/2022	Preliminary Release 1	ALL

Information furnished by Analog Devices is believed to be accurate and reliable. However, no responsibility is assumed by Analog Devices for its use, nor for any infringements of patents or other rights of third parties that may result from its use. Specifications subject to change without notice. No license is granted by implication or otherwise under any patent or patent rights of Analog Devices.