



# MAX78000 User Guide

*UGXXXX; Rev 0.1; 5/2021*

Preliminary Draft 05/21/2021

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

# MAX78000 User Guide

## Table of Contents

MAX78000 User Guide -----	2
1. Overview -----	20
1.1 Block Diagram -----	21
2. Memory, Register Mapping, and Access -----	22
2.1 Memory, Register Mapping, and Access Overview-----	22
2.2 Field Access Definitions -----	27
2.3 Standard Memory Regions-----	27
2.4 AHB Interfaces-----	31
2.4.1.1 I-Code-----	31
2.4.1.2 D-Code-----	31
2.4.1.3 System-----	31
2.4.2.1 Standard DMA-----	31
2.4.2.2 CNN and CNN TX FIFO-----	31
2.4.2.3 SPIO-----	31
2.5 Peripheral Register Map-----	32
2.6 Error Correction Coding (ECC) Module-----	33
3. System, Power, Clocks, Reset -----	34
3.1 Oscillator Sources-----	34
3.2 System Oscillator (SYS_OSC)-----	35
3.3 Operating Modes-----	38
3.3.2.1 SLEEP-----	38
3.3.2.2 LPM-----	40
3.3.2.3 UPM-----	42
3.3.2.4 STANDBY-----	44
3.3.2.5 BACKUP -----	45
3.3.2.6 PDM-----	48
3.4 Operating Modes Wakeup Sources-----	50
3.5 Device Resets -----	50
3.6 Unified Internal Cache Controllers -----	51
3.7 RAM Memory Management-----	53
3.8 Miscellaneous Control Registers (MCR) -----	54
3.9 Single Inductor Multiple Output Power Supply (SIMO)-----	57
3.10 Low-Power General Control Registers (LPGCR)-----	64
3.11 Power Sequencer Registers (PWRSEQ)-----	65
3.12 Trim System Initialization Registers (TRIMSIR) -----	72
3.13 Global Control Registers (GCR)-----	74

3.14	<i>Error Correction Coding (ECC)</i>	90
3.15	<i>System Initialization Registers (SIR)</i>	91
3.16	<i>Function Control Registers (FCR)</i>	92
3.17	<i>General Control Function Registers (GCFR)</i>	95
4.	Interrupts and Exceptions	98
4.1	<i>CM4 Interrupt and Exception Features</i>	98
4.2	<i>CM4 Interrupt Vector Table</i>	98
4.3	<i>RV32 Interrupt Vector Table</i>	100
5.	General-Purpose I/O and Alternate Function Pins (GPIO)	102
5.1	<i>Instances</i>	102
5.2	<i>Configuration</i>	103
5.3	<i>Reference Tables</i>	105
5.4	<i>Usage</i>	106
5.5	<i>Configuring GPIO (External) Interrupts</i>	107
5.6	<i>Registers</i>	109
6.	Flash Controller (FLC)	118
6.1	<i>Instances</i>	118
6.2	<i>Usage</i>	118
6.3	<i>Registers</i>	120
7.	Debug Access Port (DAP)	126
7.1	<i>Instances</i>	126
7.2	<i>Access Control</i>	126
7.3	<i>Pin Configuration</i>	126
8.	Semaphores	127
8.1	<i>Instances</i>	127
8.2	<i>Multiprocessor Communications</i>	127
8.3	<i>Registers</i>	128
9.	Standard DMA (DMA)	133
9.1	<i>Instances</i>	133
9.2	<i>DMA Channel Operation (DMA_CH)</i>	133
9.3	<i>Usage</i>	137
9.4	<i>Count-To-Zero (CTZ) Condition</i>	137
9.5	<i>Chaining Buffers</i>	138
9.6	<i>DMA Interrupts</i>	140
9.7	<i>Channel Timeout Detect</i>	140
9.8	<i>Memory-to-Memory DMA</i>	141
9.9	<i>Registers</i>	141

9.10	DMA Channel Registers	142
10.	Analog to Digital Converter (ADC) and Comparators	148
10.1	Features	148
10.2	Instances	148
10.3	Architecture	149
10.4	Clock Configuration	150
10.5	Power-Up Sequence	151
10.6	Conversion	151
10.7	Reference Scaling and Input Scaling	152
10.8	Data Limits and Out of Range Interrupts	153
10.9	Power-Down Sequence	155
10.10	Comparator Operation	155
10.11	ADC Registers	156
10.12	Low-Power Comparator Registers	160
11.	UART (UART)	162
11.1	Instances	163
11.2	DMA	164
11.3	UART Frame	164
11.4	FIFOs	164
11.5	Interrupt Events	165
11.6	LPUART Wakeup Events	167
11.7	Inactive State	167
11.8	Receive Sampling	168
11.9	Baud Rate Generation	168
11.9.4.1	Configuring a LPUART for Low-Power Modes of Operation	170
11.10	Hardware Flow Control	170
11.10.2.1	RTC/CTS Handling for Application Controlled HFC	172
11.11	Registers	173
12.	Serial Peripheral Interface (SPI)	180
12.1	Instances	181
12.2	Format	182
12.3	Pin Configuration	184
12.4	Clock Configuration	185
12.5	Registers	188
13.	I <sup>2</sup> C Master/Slave Serial Communications Peripheral (I <sup>2</sup> C)	199
13.1	I <sup>2</sup> C Master/Slave Features	199
13.2	Instances	199
13.3	I <sup>2</sup> C Overview	200

13.4	<i>Configuration and Usage</i>	202
13.4.4.1	Hs-Mode Timing	203
13.4.4.2	Hs-Mode Clock Configuration	203
13.4.6.1	I <sup>2</sup> C Master Mode Receiver Operation	206
13.4.6.2	I <sup>2</sup> C Master Mode Transmitter Operation	206
13.4.6.3	I <sup>2</sup> C Multi-Master Operation	206
13.4.7.1	Slave Transmitter	208
13.4.7.2	Slave Receivers	211
13.5	<i>Registers</i>	216
14.	Inter-Integrated Sound Interface (I <sup>2</sup> S)	232
14.1	<i>Instances</i>	232
14.2	<i>Details</i>	233
14.3	<i>Master and Slave Mode Configuration</i>	234
14.4	<i>Clocking</i>	234
14.5	<i>Data Formatting</i>	236
14.6	<i>Transmit and Receive FIFOs</i>	239
14.7	<i>Interrupt Events</i>	241
14.8	<i>Direct Memory Access</i>	242
14.9	<i>Block Operation</i>	243
14.10	<i>Registers</i>	243
15.	Camera Interface (CAMERAIF)	249
15.1	<i>Instances</i>	249
15.2	<i>Capture Modes</i>	250
15.3	<i>Timing Modes</i>	250
15.4	<i>Data Width</i>	250
15.5	<i>Data FIFO</i>	252
15.6	<i>Usage</i>	252
15.7	<i>Camera Registers</i>	253
16.	1-Wire Master (OWM)	258
16.1	<i>1-Wire Master Features</i>	258
16.2	<i>1-Wire Pins and Configuration</i>	258
16.3	<i>1-Wire Protocol</i>	259
16.3.1.1	Physical Layer	260
16.3.1.2	Link Layer	260
16.3.1.3	Network Layer	263
16.4	<i>1-Wire Operation</i>	266
16.5	<i>1-Wire Data Reads</i>	267
16.6	<i>Registers</i>	268
17.	Real-Time Clock (RTC)	273
17.1	<i>Overview</i>	273

17.2	<i>Instances</i>	274
17.3	<i>Register Access Control</i>	274
17.4	<i>RTC Alarm Functions</i>	275
17.5	<i>RTC Calibration</i>	277
17.6	<i>Registers</i>	279
18.	Timers (TMR/LPTMR)	284
18.1	<i>Instances</i>	285
18.2	<i>Basic Timer Operation</i>	285
18.3	<i>32-Bit Single / 32-Bit Cascade / Dual 16-Bit</i>	286
18.4	<i>Timer Clock Sources</i>	286
18.5	<i>Timer Pin Functionality</i>	287
18.6	<i>Wakeup Events</i>	289
18.7	<i>Operating Modes</i>	290
18.7.5.1	Capture Event	300
18.7.5.2	Rollover Event	300
18.8	<i>Registers</i>	308
19.	Wakeup Timer (WUT)	317
19.1	<i>Basic Operation</i>	317
19.2	<i>One-Shot Mode (0)</i>	318
19.3	<i>Continuous Mode (1)</i>	319
19.4	<i>Registers</i>	320
20.	Watchdog Timer (WDT)	323
20.1	<i>Instances</i>	324
20.2	<i>Usage</i>	324
20.3	<i>WDT Feed Sequence</i>	325
20.4	<i>WDT Events</i>	326
20.5	<i>Initializing the WDT</i>	329
20.6	<i>Resets</i>	329
20.7	<i>Registers</i>	329
21.	Pulse Train Engine (PT)	334
21.1	<i>Instances</i>	334
21.2	<i>Features</i>	334
21.3	<i>Engine</i>	334
21.3.1.1	Pulse Train Mode	334
21.3.1.2	In Pulse Train Mode, Set the Bit Pattern	335
21.3.1.3	Synchronize Two or More Outputs, if Needed	335
21.3.1.4	Pulse Train Loop Mode	335
21.3.1.5	Pulse Train Loop Delay	335
21.3.1.6	Pulse Train Automatic Restart Mode	335

21.4	<i>Enabling and Disabling a Pulse Train Output</i>	336
21.5	<i>Atomic Pulse Train Output Enable and Disable</i>	336
21.6	<i>Halt and Disable</i>	337
21.7	<i>Interrupts</i>	337
21.8	<i>Registers</i>	337
21.8.1.1	Pulse Train Engine Global Enable/Disable Register	338
21.8.1.2	Pulse Train Engine Safe Enable Register	341
21.8.1.3	Pulse Train Engine Safe Disable Register	341
21.8.1.4	Pulse Train Registers	342
22.	Cyclic Redundancy Check (CRC)	345
22.1	<i>Instances</i>	345
22.2	<i>Usage</i>	345
22.3	<i>Polynomial Generation</i>	346
22.4	<i>Calculations Using Software Writes to the FIFO</i>	346
22.5	<i>Calculations Using DMA</i>	347
22.6	<i>Registers</i>	348
23.	Advanced Encryption Standard (AES)	350
23.1	<i>Instances</i>	350
23.2	<i>Encryption of 128-Bit Blocks of Data Using FIFO</i>	350
23.3	<i>Encryption of 128-Bit Blocks Using DMA</i>	350
23.4	<i>Encryption of Blocks Less Than 128-Bits</i>	352
23.5	<i>Decryption</i>	352
23.6	<i>Interrupt Events</i>	352
23.7	<i>Registers</i>	353
24.	TRNG Engine	356
24.1	<i>Registers</i>	356
25.	Bootloader	358
25.1	<i>Instances</i>	358
25.2	<i>Bootloader Operating States</i>	359
25.3	<i>Creating the Motorola SREC File</i>	360
25.4	<i>Bootloader Activation</i>	361
25.5	<i>Bootloader</i>	361
25.6	<i>Secure Bootloader</i>	362
25.7	<i>Command Protocol</i>	363
25.8	<i>General Commands</i>	364
25.9	<i>Secure Commands</i>	368
25.10	<i>Challenge/Response Commands</i>	370
26.	Convolutional Neural Network (CNN)	371

26.1	<i>Overview</i>	371
26.2	<i>Instances</i>	375
26.3	<i>Memory Configuration</i>	376
26.4	<i>CNN Global Registers (CNN)</i>	381
26.5	<i>CNNx16 Processor Array (CNNx16_n) Registers</i>	386
26.5.2.1	CNNx16_n Per Layer Registers (CNNx16_n_Ly)	396
26.5.2.2	CNNx16_n Processor Stream Registers	407
27.	Revision History	410

Preliminary Draft 05/21/2021

## List of Figures

Figure 1-1: MAX78000 Block Diagram .....	21
Figure 2-1: CM4 Code Memory Mapping .....	23
Figure 2-2: RISC-V IBUS Code Memory Mapping .....	24
Figure 2-3: CM4 Peripheral and Data Memory Mapping.....	25
Figure 2-4: RV32 Peripheral and Data Memory Mapping.....	26
Figure 2-5: Unique Serial Number Format.....	28
Figure 3-1: MAX78000 Clock Block Diagram.....	37
Figure 3-2: SLEEP Mode Clock Control.....	39
Figure 3-3: Low Power Mode (LPM) Clock and State Retention Diagram .....	41
Figure 3-4: Micro Power Mode (UPM) Clock and State Retention Block Diagram .....	43
Figure 3-5: STANDBY Mode Clock and State Retention Block Diagram.....	45
Figure 3-6: BACKUP Mode Clock and State Retention Block Diagram.....	47
Figure 3-7: Power Down Mode (PDM) Clock and State Retention Block Diagram .....	49
Figure 9-1: DMA Block-Chaining Flowchart .....	139
Figure 10-1: Analog to Digital Converter Block Diagram .....	149
Figure 10-2: ADC Limit Engine .....	154
Figure 11-1: UART Block Diagram .....	163
Figure 11-2: UART Frame Structure .....	164
Figure 11-3: UART Interrupt Functional Diagram .....	165
Figure 11-4: Oversampling Example .....	168
Figure 11-5: UART Baud Rate Generation .....	168
Figure 11-6: LPUART Timing Generation .....	169
Figure 11-7: Hardware Flow Control Physical Connection .....	171
Figure 11-8: Hardware Flow Control Signaling for Transmitting to an External Receiver .....	172
Figure 12-1: SPI Block Diagram .....	181
Figure 12-2: 4-Wire SPI Connection Diagram .....	183
Figure 12-3: Generic 3-Wire SPI Master to Slave Connection .....	184
Figure 12-4: Dual Mode SPI Connection Diagram.....	185
Figure 12-5: SCK Clock Rate Control .....	186
Figure 12-6: SPI Clock Polarity .....	187
Figure 13-1: I <sup>2</sup> C Write Data Transfer.....	201
Figure 13-2: I <sup>2</sup> C SCL Timing for Standard, Fast and Fast-Plus Modes .....	202
Figure 14-1: I <sup>2</sup> S Master Mode, Full Duplex Connection .....	233
Figure 14-2: I <sup>2</sup> S Slave Mode .....	234
Figure 14-3: Audio Interface I <sup>2</sup> S Signal Diagram .....	234
Figure 14-4: Audio Mode with Inverted Word Select Polarity.....	236
Figure 14-5: Audio Master Mode Left-Justified First Bit Location .....	237
Figure 14-6: MSB Adjustment when Sample Size is Less Than Bits Per Word .....	237
Figure 14-7: LSB Adjustment when Sample Size is Less Than Bits Per Word.....	238
Figure 14-8: I <sup>2</sup> S Mono Left Mode.....	238
Figure 14-9: I <sup>2</sup> S Mono Right Mode.....	239
Figure 15-1: Horizontal and Vertical Synchronization Timing Mode with 8-Bit Data Width .....	251
Figure 15-2: Data Stream Timing Mode with 8-Bit Data Width.....	251
Figure 15-3: 10 or 12-bit PCIF_VSYNC/PCIF_HSYNC .....	252
Figure 16-1: 1-Wire Signal Interface .....	260
Figure 16-2: 1-Wire Reset Pulse.....	261
Figure 16-3: 1-Wire Write Time Slot .....	262
Figure 16-4: 1-Wire Read Time Slot .....	262

Figure 16-5: 1-Wire ROM ID Fields .....	263
Figure 17-1: MAX78000 RTC Block Diagram (12-bit Sub-Second Counter) .....	273
Figure 17-2: RTC Interrupt/Wakeup Diagram Wakeup Function.....	276
Figure 17-3: Internal Implementation of 4kHz Digital Trim .....	278
Figure 18-1: MAX78000 TimerA Output Functionality, Modes 0/1/3/5..	288
Figure 18-2: MAX78000 TimerA Input Functionality, Modes 2/4/6/7/8/14.....	289
Figure 18-3: Timer I/O Signal Naming Conventions.....	290
Figure 18-4: One-Shot Mode Diagram.....	293
Figure 18-5: Continuous Mode Diagram.....	295
Figure 18-6: Counter Mode Diagram.....	297
Figure 18-7: Capture Mode Diagram .....	301
Figure 18-8: Compare Mode Diagram .....	303
Figure 18-9: Gated Mode Diagram .....	305
Figure 18-10: Capture/Compare Mode Diagram.....	307
Figure 19-1: One-Shot Mode Diagram.....	318
Figure 19-2: Continuous Mode Diagram.....	319
Figure 20-1: Windowed Watchdog Timer Block Diagram.....	324
Figure 20-2: WDT Early Interrupt and Reset Event Sequencing Details .....	327
Figure 20-3: WDT Late Interrupt and Reset Event Sequencing Details...	328
Figure 25-1: MAX78000 Combined Bootloader Flow .....	362
Figure 26-1: CNN Overview.....	374
Figure 26-2: CNNx16_n Processor Quadrant Block Diagram .....	375
Figure 26-3: CNN Global and Quad CNNx16n Processor Array APB Memory Map .....	376

Preliminary Draft 05/21/2021

## List of Tables

Table 2-1: Field Access Definitions .....	27
Table 2-2: System SRAM Configuration .....	29
Table 2-3: AHB Slave Base Address Map .....	31
Table 2-4: APB Peripheral Base Address Map.....	32
Table 3-1: Available System Oscillators .....	35
Table 3-2: Reset Sources and Effect on Oscillator and System Clock .....	36
Table 3-3 System RAM Retention in BACKUP Mode.....	46
Table 3-4: Wakeup Sources for Each Operating Mode in the MAX78000 .....	50
Table 3-5: Operating Mode and Clock Status .....	51
Table 3-6: Instruction Cache Controller Register Summary.....	52
Table 3-7: ICC0 Cache Information Register .....	52
Table 3-8: ICC0 Memory Size Register .....	53
Table 3-9: ICC0 Cache Control Register .....	53
Table 3-10: ICC0 Invalidate Register .....	53
Table 3-11: Miscellaneous Control Register Summary .....	54
Table 3-12: Error Correction Coding Enable Register .....	54
Table 3-13: IPO Manual Register .....	55
Table 3-14: Output Enable Register.....	55
Table 3-15: Comparator 0 Control Register .....	55
Table 3-16: Miscellaneous Control Register .....	56
Table 3-17: GPIO3 Pin Control Register .....	57
Table 3-18: SIMO Power Supply Device Pin Connectivity.....	58
Table 3-19: SIMO Controller Register Summary .....	58
Table 3-20: SIMO Buck Voltage Regulator A Control Register.....	59
Table 3-21: SIMO Buck Voltage Regulator B Control Register.....	59
Table 3-22: SIMO Buck Voltage Regulator C Control Register .....	60
Table 3-23: SIMO High Side FET Peak Current V <sub>REGO_A</sub> V <sub>REGO_B</sub> Register .....	60
Table 3-24: SIMO High Side FET Peak Current V <sub>REGO_C</sub> Register .....	61
Table 3-25: SIMO Maximum High Side FET Time On Register .....	61
Table 3-26: SIMO Buck Cycle Count V <sub>REGO_A</sub> Register.....	61
Table 3-27: SIMO Buck Cycle Count V <sub>REGO_B</sub> Register .....	61
Table 3-28: SIMO Buck Cycle Count V <sub>REGO_C</sub> Register .....	61
Table 3-29: SIMO Buck Cycle Count Alert V <sub>REGO_A</sub> Register .....	61
Table 3-30: SIMO Buck Cycle Count Alert V <sub>REGO_B</sub> Register .....	62
Table 3-31: SIMO Buck Cycle Count Alert V <sub>REGO_C</sub> Register .....	62
Table 3-32: SIMO Buck Regulator Output Ready Register.....	62
Table 3-33: SIMO Zero Cross Calibration V <sub>REGO_A</sub> Register .....	62
Table 3-34: SIMO Zero Cross Calibration V <sub>REGO_B</sub> Register .....	63
Table 3-35: SIMO Zero Cross Calibration V <sub>REGO_C</sub> Register .....	63
Table 3-36 Low-Power Control Register Summary .....	64
Table 3-37: Reset Control Register .....	64
Table 3-38: Clock Disable Register .....	65
Table 3-39: Power Sequencer Register Summary.....	65
Table 3-40: Low Power Control Register .....	66
Table 3-41: GPIO0 Low Power Wakeup Status Flags .....	67
Table 3-42: GPIO0 Low Power Wakeup Enable Registers.....	67
Table 3-43: GPIO1 Low Power Wakeup Status Flags .....	68
Table 3-44: GPIO1 Low Power Wakeup Enable Registers.....	68
Table 3-45: GPIO2 Low Power Wakeup Status Flags .....	68
Table 3-46: GPIO2 Low Power Wakeup Enable Registers.....	69

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

Table 5-1: MAX78000 GPIO Pin Count .....	103
Table 5-2: MAX78000 GPIO Pin Function Configuration .....	104
Table 5-3: MAX78000 Input Mode Configuration .....	104
Table 5-4: MAX78000 Output Mode Configuration.....	105
Table 5-5: MAX78000 GPIO0 Alternate Function Configuration Reference .....	105
Table 5-6: MAX78000 GPIO0 Output/Input Configuration Reference .....	105
Table 5-7: MAX78000 GPIO0 Interrupt Configuration Reference .....	106
Table 5-8: MAX78000 GPIO Pullup/Pulldown/Drive Strength/Voltage Configuration Reference.....	106
Table 5-9: MAX78000 GPIO Port Interrupt Vector Mapping .....	108
Table 5-10: MAX78000 GPIO Wakeup Interrupt Vector.....	108
Table 5-11: GPIO Register Summary.....	109
Table 5-12: GPIO Port n Configuration Enable Bit 0 Register .....	110
Table 5-13: GPIO Port n Configuration Enable Atomic Set Bit 0 Register .....	110
Table 5-14: GPIO Port n Configuration Enable Atomic Clear Bit 0 Register.....	110
Table 5-15: GPIO Port n Output Enable Register .....	110
Table 5-16: GPIO Port n Output Enable Atomic Set Register.....	111
Table 5-17: GPIO Port n Output Enable Atomic Clear Register .....	111
Table 5-18: GPIO Port n Output Register.....	111
Table 5-19: GPIO Port n Output Atomic Set Register .....	111
Table 5-20: GPIO Port n Output Atomic Clear Register .....	111
Table 5-21: GPIO Port n Input Register .....	112
Table 5-22: GPIO Port n Interrupt Mode Register .....	112
Table 5-23: GPIO Port n Interrupt Polarity Register .....	112
Table 5-24: GPIO Port n Input Enable Register .....	112
Table 5-25: GPIO Port n Interrupt Enable Register .....	113
Table 5-26: GPIO Port n Interrupt Enable Atomic Set Register.....	113
Table 5-27: GPIO Port n Interrupt Enable Atomic Clear Register .....	113
Table 5-28: GPIO Port n Interrupt Status Register.....	113
Table 5-29: GPIO Port n Interrupt Clear Register.....	113
Table 5-30: GPIO Port n Wakeup Enable Register .....	114
Table 5-31: GPIO Port n Wakeup Enable Atomic Set Register .....	114
Table 5-32: GPIO Port n Wakeup Enable Atomic Clear Register.....	114
Table 5-33: GPIO Port n Interrupt Dual Edge Mode Register .....	114
Table 5-34: GPIO Port n Pad Configuration 1 Register .....	114
Table 5-35: GPIO Port n Pad Configuration 2 Register .....	114
Table 5-36: GPIO Port n Configuration Enable Bit 1 Register .....	115
Table 5-37: GPIO Port n Configuration Enable Atomic Set Bit 1 Register .....	115
Table 5-38: GPIO Port n Configuration Enable Atomic Clear Bit 1 Register .....	115
Table 5-39: GPIO Port n Configuration Enable Bit 2 Register .....	115
Table 5-40: GPIO Port n Configuration Enable Atomic Set Bit 2 Register .....	116
Table 5-41: GPIO Port n Configuration Enable Atomic Clear Bit 2 Register .....	116
Table 5-42: GPIO Port n Hysteresis Enable Register .....	116
Table 5-43: GPIO Port n Output Drive Strength Bit 0 Register .....	116
Table 5-44: GPIO Port n Output Drive Strength Bit 0 Register .....	116
Table 5-45: GPIO Port n Output Drive Strength Bit 1 Register .....	116
Table 5-46: GPIO Port n Pulldown/Pullup Strength Select Register .....	116
Table 5-47: GPIO Port n Voltage Select Register .....	117
Table 6-1: MAX78000 Internal Flash Memory Organization .....	118
Table 6-2: Valid Addresses Flash Writes .....	119
Table 6-3: Flash Controller Register Summary .....	120
Table 6-4: Flash Controller Address Pointer Register .....	121
Table 6-5: Flash Controller Clock Divisor Register .....	121
Table 6-6: Flash Controller Control Register .....	121

Table 6-7: Flash Controller Interrupt Register .....	122
Table 6-8: Flash Controller Data 0 Register .....	123
Table 6-9: Flash Controller Data Register 1 .....	123
Table 6-10: Flash Controller Data Register 2 .....	123
Table 6-11: Flash Controller Data Register 3 .....	124
Table 6-12: Flash Controller Access Control Register .....	124
Table 6-13: Flash Write/Lock 0 Register .....	124
Table 6-14: Flash Write/Lock 1 Register .....	124
Table 6-15: Flash Read Lock 0 Register .....	125
Table 6-16: Flash Read Lock 1 Register .....	125
Table 7-1: MAX78000 DAP Instances.....	126
Table 8-1: MAX78000 Semaphore Instances.....	127
Table 8-2: Semaphore Register Summary .....	128
Table 8-3: Semaphore 0 Register.....	128
Table 8-4: Semaphore 1 Register .....	128
Table 8-5: Semaphore 2 Register.....	129
Table 8-6: Semaphore 3 Register.....	129
Table 8-7: Semaphore 4 Register.....	129
Table 8-8: Semaphore 5 Register.....	129
Table 8-9: Semaphore 6 Register.....	129
Table 8-10: Semaphore 7 Register.....	130
Table 8-11: Semaphore Interrupt 0 Register .....	130
Table 8-12: Semaphore Mailbox 0 Register .....	130
Table 8-13: Semaphore Interrupt 1 Register .....	131
Table 8-14: Semaphore Mailbox 1 Register .....	131
Table 8-15: Semaphore Status Register.....	131
Table 9-1: MAX78000 DMA and Channel Instances .....	133
Table 9-2: DMA Source and Destination by Peripheral .....	134
Table 9-3: Data Movement from Source to DMA FIFO.....	135
Table 9-4: Data Movement from the DMA FIFO to Destination .....	136
Table 9-5: DMA Channel Timeout Configuration.....	140
Table 9-6: DMA Register Summary.....	141
Table 9-7: DMA Interrupt Enable Register.....	141
Table 9-8: DMA Interrupt Flag Register .....	141
Table 9-9: Standard DMA Channel 0 to Channel 3 Register Summary .....	142
Table 9-10: DMA Channel Registers Summary .....	142
Table 9-11: DMA_CH n Control Register.....	142
Table 9-12: DMA Status Register .....	144
Table 9-13: DMA Channel n Source Register .....	145
Table 9-14: DMA Channel n Destination Register .....	146
Table 9-15: DMA Channel n Count Register .....	146
Table 9-16: DMA Channel n Source Reload Register .....	146
Table 9-17: DMA Channel n Destination Reload Register .....	146
Table 9-18: DMA Channel n Count Reload Register .....	147
Table 10-1: MAX78000 ADC Input Pins for the 81-CTBGA Package.....	148
Table 10-2: MAX78000 ADC Clock Frequency and ADC Conversion Time with the System Clock set to the IPO .....	150
Table 10-3: ADC Data Register Alignment Options.....	152
Table 10-4: Input and Reference Scale Support by ADC Input Channel .....	153
Table 10-5: ADC Registers Summary.....	156
Table 10-6: ADC Control Register .....	156
Table 10-7: ADC Status Register .....	158
Table 10-8: ADC Data Register .....	158
Table 10-9: ADC Interrupt Control Register .....	158

Table 10-10: ADC Limit 0 to 3 Registers .....	159
Table 10-11: Low-Power Comparator Registers Summary .....	160
Table 10-12: Low-Power Comparator n Registers .....	160
Table 11-1: MAX78000 UART/LPUART Instances .....	163
Table 11-2: MAX78000 Interrupt Events .....	165
Table 11-3: Frame Error Detection for Standard UARTs and LPUART .....	166
Table 11-4: Frame Error Detection for LPUARTs with <code>UARTn_CTRL.fdm = 1</code> and <code>UARTn_CTRL.dpfe_en = 1</code> .....	166
Table 11-5: MAX78000 Wakeup Events .....	167
Table 11-6: LPUART Low Baud Rate Generation Examples ( <code>UARTn_CTRL.fdm = 1</code> ) .....	170
Table 11-7: UART/LPUART Register Summary .....	173
Table 11-8: UART Control Register .....	173
Table 11-9: UART Status Register .....	175
Table 11-10: UART Interrupt Enable Register .....	175
Table 11-11: UART Interrupt Flag Register .....	176
Table 11-12: UART Clock Divisor Register .....	176
Table 11-13: UART Oversampling Control Register .....	177
Table 11-14: UART Transmit FIFO Register .....	177
Table 11-15: UART Pin Control Register .....	177
Table 11-16: UART Data Register .....	178
Table 11-17: UART DMA Register .....	178
Table 11-18: UART Wakeup Enable .....	178
Table 11-19: UART Wakeup Flag Register .....	179
Table 12-1: MAX78000 SPI Instances .....	181
Table 12-2: MAX78000 SPI Peripheral Pins .....	182
Table 12-3: Four-Wire Format Signals .....	182
Table 12-4: Three-Wire Format Signals .....	183
Table 12-5: SPI Modes Clock Phase and Polarity Operation .....	187
Table 12-6: SPI Register Summary .....	188
Table 12-7: SPI FIFO32 Register .....	189
Table 12-8: SPI 16-bit FIFO Register .....	189
Table 12-9: SPI 8-bit FIFO Register .....	189
Table 12-10: SPI Control 0 Register .....	189
Table 12-11: SPI Control 1 Register .....	191
Table 12-12: SPI Control 2 Register .....	191
Table 12-13: SPI Slave Select Timing Register .....	192
Table 12-14: SPI Master Clock Configuration Registers .....	193
Table 12-15: SPI DMA Control Registers .....	194
Table 12-16: SPI Interrupt Status Flags Registers .....	195
Table 12-17: SPI Interrupt Enable Registers .....	196
Table 12-18: SPI Wakeup Status Flags Registers .....	197
Table 12-19: SPI Wakeup Enable Registers .....	197
Table 12-20: SPI Slave Select Timing Registers .....	198
Table 13-1: MAX78000 I <sup>2</sup> C Peripheral Pins .....	199
Table 13-2: I <sup>2</sup> C Bus Terminology .....	200
Table 13-3: Calculated I <sup>2</sup> C Bus Clock Frequencies .....	203
Table 13-4: I <sup>2</sup> C Slave Address Format .....	204
Table 13-5: I <sup>2</sup> C Register Summary .....	217
Table 13-6: I <sup>2</sup> C Control Register .....	217
Table 13-7: I <sup>2</sup> C Status Register .....	219
Table 13-8: I <sup>2</sup> C Interrupt Flag 0 Register .....	219
Table 13-9: I <sup>2</sup> C Interrupt Enable 0 Register .....	222
Table 13-10: I <sup>2</sup> C Interrupt Flag 1 Register .....	223
Table 13-11: I <sup>2</sup> C Interrupt Enable 1 Register .....	224

Table 13-12: I <sup>2</sup> C FIFO Length Register .....	224
Table 13-13: I <sup>2</sup> C Receive Control 0 Register .....	224
Table 13-14: I <sup>2</sup> C Receive Control 1 Register .....	225
Table 13-15: I <sup>2</sup> C Transmit Control 0 Register .....	226
Table 13-16: I <sup>2</sup> C Transmit Control 1 Register .....	227
Table 13-17: I <sup>2</sup> C Data Register .....	228
Table 13-18: I <sup>2</sup> C Master Control Register .....	228
Table 13-19: I <sup>2</sup> C SCL Low Control Register .....	229
Table 13-20: I <sup>2</sup> C SCL High Control Register .....	229
Table 13-21: I <sup>2</sup> C Hs-Mode Clock Control Register .....	229
Table 13-22: I <sup>2</sup> C Timeout Register .....	230
Table 13-23: I <sup>2</sup> C DMA Register .....	230
Table 13-24: I <sup>2</sup> C Slave Address Register .....	230
Table 14-1: MAX78000 I <sup>2</sup> S Instances .....	232
Table 14-2: MAX78000 I <sup>2</sup> S Pin Mapping .....	233
Table 14-3: I <sup>2</sup> S Mode Configuration .....	234
Table 14-4: Data Ordering for Byte Data Size (Stereo Mode) .....	240
Table 14-5: Data Ordering for Half-Word Data Size (Stereo Mode) .....	240
Table 14-6: Data Ordering for Word Data Size (Stereo Mode) .....	240
Table 14-7: Configuration for Typical Audio Width and Samples per WS Clock Cycle .....	241
Table 14-8: I <sup>2</sup> S Interrupt Events .....	241
Table 14-9: I <sup>2</sup> S Register Summary .....	243
Table 14-10: I <sup>2</sup> S Control 0 Register .....	243
Table 14-11: I <sup>2</sup> S Master Mode Configuration Register .....	245
Table 14-12: I <sup>2</sup> S DMA Control Register .....	246
Table 14-13: I <sup>2</sup> S FIFO Register .....	247
Table 14-14: I <sup>2</sup> S Interrupt Flag Register .....	247
Table 14-15: I <sup>2</sup> S Interrupt Enable Register .....	247
Table 15-1: MAX78000 CAMERAIF Instances .....	249
Table 15-2: MAX78000 CAMERAIF Signals .....	249
Table 15-3: Parallel Camera Interface Register Summary .....	253
Table 15-4: CAMERAIF Version Register .....	253
Table 15-5: CAMERAIF FIFO Size Register .....	254
Table 15-6: CAMERAIF Configuration Register .....	254
Table 15-7: CAMERAIF Interrupt Enable Register .....	255
Table 15-8: CAMERAIF Status Flags Register .....	256
Table 15-9: CAMERAIF Timing Codes Register .....	257
Table 15-10: CAMERAIF FIFO Data Register .....	257
Table 16-1: MAX78000 1-Wire Master Peripheral Pins .....	259
Table 16-2: 1-Wire ROM Commands .....	263
Table 16-3: 1-Wire Slave Device ROM ID Field .....	264
Table 16-4: OWM Register Summary .....	268
Table 16-5: OWM Configuration Register .....	268
Table 16-6: OWM Clock Divisor Register .....	270
Table 16-7: OWM Control Status Register .....	270
Table 16-8: OWM Data Buffer Register .....	271
Table 16-9: OWM Interrupt Flag Register .....	271
Table 16-10: OWM Interrupt Enable Register .....	272
Table 17-1: RTC Seconds, Sub-Seconds, Time-of-Day Alarm and Sub-Seconds Alarm Register Details .....	274
Table 17-2: RTC Register Access .....	274
Table 17-3: MAX78000 RTC Square Wave Output Configuration .....	277
Table 17-4: RTC Register Summary .....	279
Table 17-5: RTC Seconds Counter Register .....	279

Table 17-6: RTC Sub-Second Counter Register (12-bit) .....	280
Table 17-7: RTC Time-of-Day Alarm Register.....	280
Table 17-8: RTC Sub-Second Alarm Register.....	280
Table 17-9: RTC Control Register .....	280
Table 17-10: RTC 32KHz Oscillator Digital Trim Register .....	282
Table 17-11: RTC 32KHz Oscillator Control Register .....	283
Table 18-1: MAX78000 TMR/LPTMR Instances .....	285
Table 18-2: MAX78000 TMR/LPTMR Instances Capture Events .....	285
Table 18-3: TimerA/TimerB 32-Bit Field Allocations.....	286
Table 18-4: MAX78000 Wakeup Events.....	289
Table 18-5: MAX78000 Operating Mode Signals for Timer 0 and Timer 1 .....	290
Table 18-6: MAX78000 Operating Mode Signals for Timer 2 and Timer 3 .....	291
Table 18-7: MAX78000 Operating Mode Signals for Low-Power Timer 0 and Low-Power Timer 1 .....	291
Table 18-8: Timer Register Summary.....	308
Table 18-9: Timer Count Register .....	309
Table 18-10: Timer Compare Register .....	309
Table 18-11: Timer PWM Register .....	309
Table 18-12: Timer Interrupt Register .....	309
Table 18-13: Timer Control 0 Register .....	310
Table 18-14: Timer Non-Overlapping Compare Register.....	313
Table 18-15: Timer Control 1 Register .....	313
Table 18-16: Timer Wakeup Status Register.....	315
Table 19-1: MAX78000 WUT Clock Period.....	317
Table 19-2: Wakeup Timer Register Summary .....	320
Table 19-3: Wakeup Timer Count Register .....	321
Table 19-4: Wakeup Timer Compare Register .....	321
Table 19-5: Wakeup Timer PWM Register.....	321
Table 19-6: Wakeup Timer Interrupt Register .....	321
Table 19-7: Wakeup Timer Control Register .....	321
Table 19-8: Wakeup Timer Non-Overlapping Compare Register .....	322
Table 20-1: MAX78000 WDT Instances Summary .....	324
Table 20-2: MAX78000 WDT Event Summary .....	326
Table 20-3: WDT Register Summary .....	330
Table 20-4: WDT Control Register .....	330
Table 20-5: WDT Reset Register .....	333
Table 20-6: WDT Clock Source Select Register .....	333
Table 20-7: WDT Count Register.....	333
Table 21-1: Pulse Train Engine Register Summary .....	337
Table 21-2: Pulse Train Engine Global Enable/Disable Register .....	338
Table 21-3: Pulse Train Engine Resync Register.....	338
Table 21-4: Pulse Train Engine Stopped Interrupt Flag Register .....	339
Table 21-5: Pulse Train Engine Interrupt Enable Register .....	340
Table 21-6: Pulse Train Engine Safe Enable Register .....	341
Table 21-7: Pulse Train Engine Safe Disable Register .....	341
Table 21-8: Pulse Train Engine Configuration Register .....	342
Table 21-9: Pulse Train Mode Bit Pattern Register.....	342
Table 21-10: Pulse Train n Loop Configuration Register.....	343
Table 21-11: Pulse Train n Automatic Restart Configuration Register .....	343
Table 22-1: MAX78000 CRC Instances .....	345
Table 22-2: Organization of Calculated Result in the CRC_VAL.value field .....	346
Table 22-3: Common CRC Polynomials.....	346
Table 22-4: CRC Register Summary.....	348
Table 22-5: CRC Control Register .....	348

Table 22-6: CRC Data Input 32 Register .....	348
Table 22-7: CRC Polynomial Register .....	349
Table 22-8: CRC Value Register.....	349
Table 23-1: MAX78000 AES Instances .....	350
Table 23-2: Interrupt Events .....	352
Table 23-3: AES Register Summary .....	353
Table 23-4: AES Control Register .....	353
Table 23-5: AES Status Register .....	354
Table 23-6: AES Interrupt Flag Register .....	354
Table 23-7: AES Interrupt Enable Register .....	355
Table 23-8: AES FIFO Register .....	355
Table 24-1: TRNG Register Summary .....	356
Table 24-2: TRNG Control Register .....	356
Table 24-3: TRNG Status Register .....	356
Table 24-4: TRNG Data Register .....	357
Table 25-1:MAX78000 Bootloader Instances .....	358
Table 25-2: The Bootloader Operating States and Prompts .....	359
Table 25-3: PERMLOCK Command Summary .....	359
Table 25-4: CHALLENGE Command Summary .....	363
Table 25-5: MAX78000 General Command Summary .....	364
Table 25-6: L - Load .....	364
Table 25-7: P – Page Erase .....	364
Table 25-8: V – Verify .....	365
Table 25-9: LOCK – Lock Device .....	365
Table 25-10: PLOCK – Permanent Lock .....	365
Table 25-11: UNLOCK – Unlock Device .....	366
Table 25-12: H – Check Device .....	366
Table 25-13: I – Get ID .....	366
Table 25-14: S – Status .....	367
Table 25-15: Q – Quit .....	367
Table 25-16: MAX78000 Secure Command Summary .....	368
Table 25-17: LK – Load Application Key .....	368
Table 25-18: LK – Load Challenge Key .....	368
Table 25-19: VK – Verify Application Key .....	368
Table 25-20: VC – Verify Challenge Key .....	369
Table 25-21: AK – Activate Application Key .....	369
Table 25-22: AC – Activate Challenge Key .....	369
Table 25-23: WL – Write Code Length .....	370
Table 25-24: MAX78000 Challenge/Response Command Summary .....	370
Table 25-25: GC – Get Challenge .....	370
Table 25-26: SR – Send Response .....	370
Table 26-1: CNN Memory Configuration (APB Accessible) for the Quad CNNx16n .....	377
Table 26-2: CNNx16 Processor Array 0 TRAM Mapping Details (APB Accessible) .....	377
Table 26-3: CNNx16 Processor Array 1 TRAM Mapping Details (APB Accessible) .....	378
able 26-4: CNNx16 Processor Array 2 TRAM Mapping Details (APB Accessible) .....	378
Table 26-5: CNNx16 Processor Array 3 TRAM Mapping Details (APB Accessible) .....	379
Table 26-6: CNNx16 Processor Array 0 MRAM Mapping Details (APB Accessible) .....	379
Table 26-7: CNNx16 Processor Array 1 MRAM Mapping Details (APB Accessible) .....	380
Table 26-8: CNNx16 Processor Array 2 MRAM Mapping Details (APB Accessible) .....	380
Table 26-9: CNNx16 Processor Array3 MRAM Mapping Details (APB Accessible) .....	381
Table 26-10: Global CNN Register Summary .....	381
Table 26-11: CNN FIFO Control Register .....	381
Table 26-12:CNN FIFO Status Register .....	383

Table 26-13: CNN FIFO 0 Write Register .....	384
Table 26-14: CNN FIFO 1 Write Register .....	384
Table 26-15: CNN FIFO 2 Write Register .....	384
Table 26-16: CNN FIFO 3 Write Register .....	384
Table 26-17: CNN Always On Domain Control Register .....	384
Table 26-18: CNNx16_n Instances and Base Offset Address .....	386
Table 26-19: CNNx16_n Processor Array Registers .....	386
Table 26-20: CNNx16_n Control Register .....	387
Table 26-21: CNNx16_n SRAM Control Register .....	390
Table 26-22: CNNx16_n Layer Count Maximum Register .....	392
Table 26-23: CNNx16_n SRAM Test Register .....	392
Table 26-24: CNNx16_n_Ly Row Count Register .....	396
Table 26-25: CNNx16_n_Ly Column Count Register .....	397
Table 26-26: CNNx16_n_Ly One Dimensional Control Register .....	397
Table 26-27: CNNx16_n_Ly Pool Row Count Register .....	399
Table 26-28: CNNx16_n_Ly Pool Column Count Register .....	399
Table 26-29: CNNx16_n_Ly Stride Count Register .....	399
Table 26-30: CNNx16_n_Ly Write Pointer Base Address Register .....	399
Table 26-31: CNNx16_n_Ly Write Pointer Timeslot Offset Register .....	400
Table 26-32: CNNx16_n_Ly Write Pointer Mask Offset Register .....	400
Table 26-33: CNNx16_n_Ly Write Pointer Multi-Pass Channel Offset Register .....	400
Table 26-34: CNNx16_n_Ly Read Pointer Base Address Register .....	401
Table 26-35: CNNx16_n_Ly Layer Control 0 Register .....	401
Table 26-36: CNNx16_n_Ly Layer Mask Count Register .....	403
Table 26-37: CNNx16_n_Ly TRAM Pointer Register .....	404
Table 26-38: CNNx16_n_Ly Enable Register .....	404
Table 26-39: CNNx16_n_Ly Post Processing Register .....	404
Table 26-40: CNNx16_n_Ly Layer Control 1 Register .....	406
Table 26-41: CNNx16_n_Muxselator Data Register .....	407
Table 26-42: CNNx16_n_Sz Stream Control 0 Register .....	407
Table 26-43: CNNx16_n_Sz Stream Control 1 Register .....	407
Table 26-44: CNNx16_n_Sz Stream Frame Buffer Size Register .....	408
Table 26-45: CNNx16_n Input FIFO Frame Size Register .....	409
Table 27-1: Revision History .....	410

## 1. Overview

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

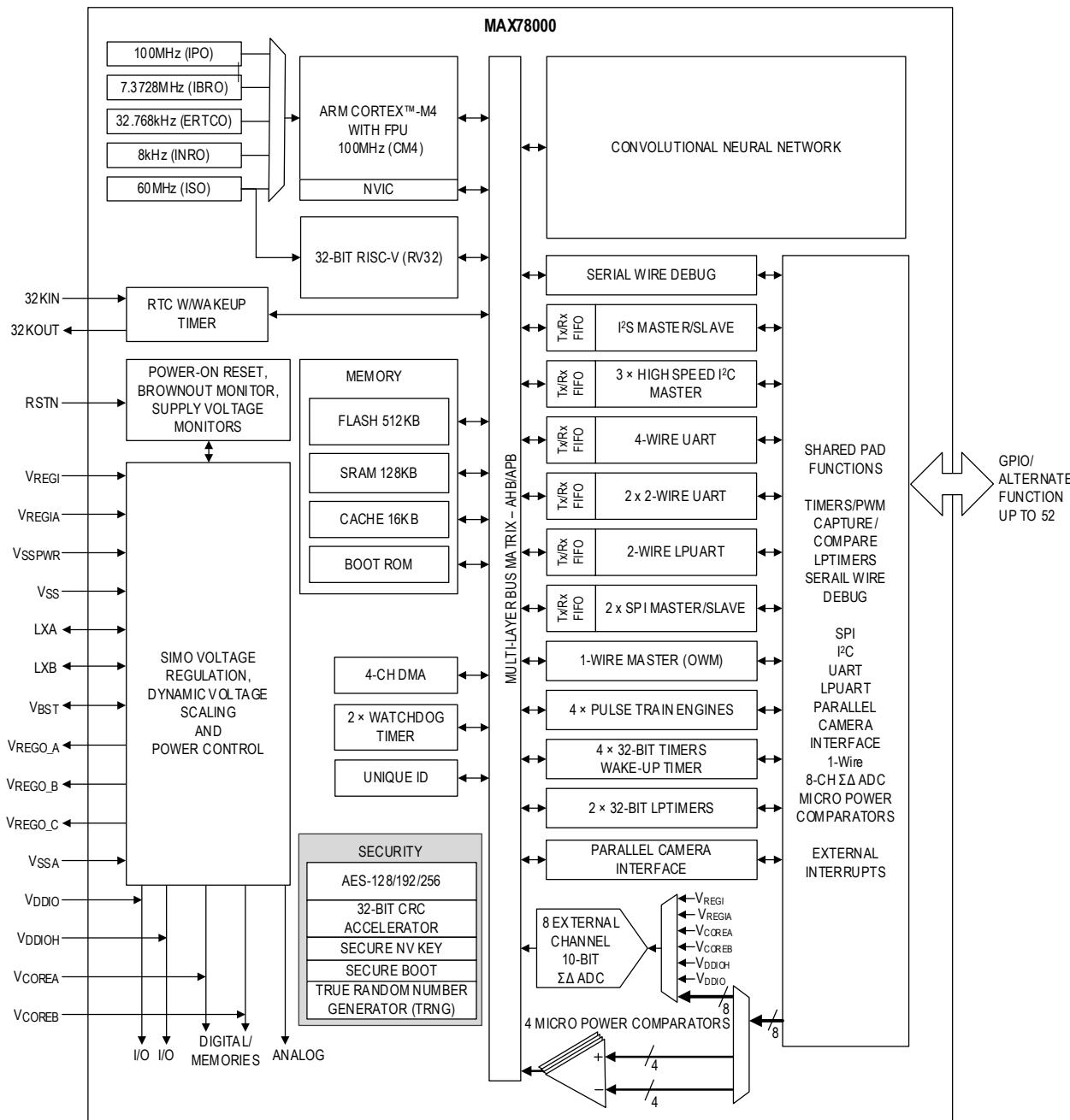
The MAX78000 is an advanced system-on-chip featuring an Arm® Cortex®-M4 with FPU CPU for efficient computation of complex functions and algorithms with integrated power management. It also includes 442KB weight CNN accelerator. The devices offer large on-chip memory with 512KB flash and up to 128KB SRAM. Multiple high-speed and low-power communications interfaces are supported, including high-speed SPI, I<sup>2</sup>C serial interface, and LPUART. Additional low-power peripherals include flexible LPTMRs and analog comparators.

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

The high-level block diagram for the MAX78000 is shown in *Figure 1-1*.

## 1.1 Block Diagram

Figure 1-1: MAX78000 Block Diagram



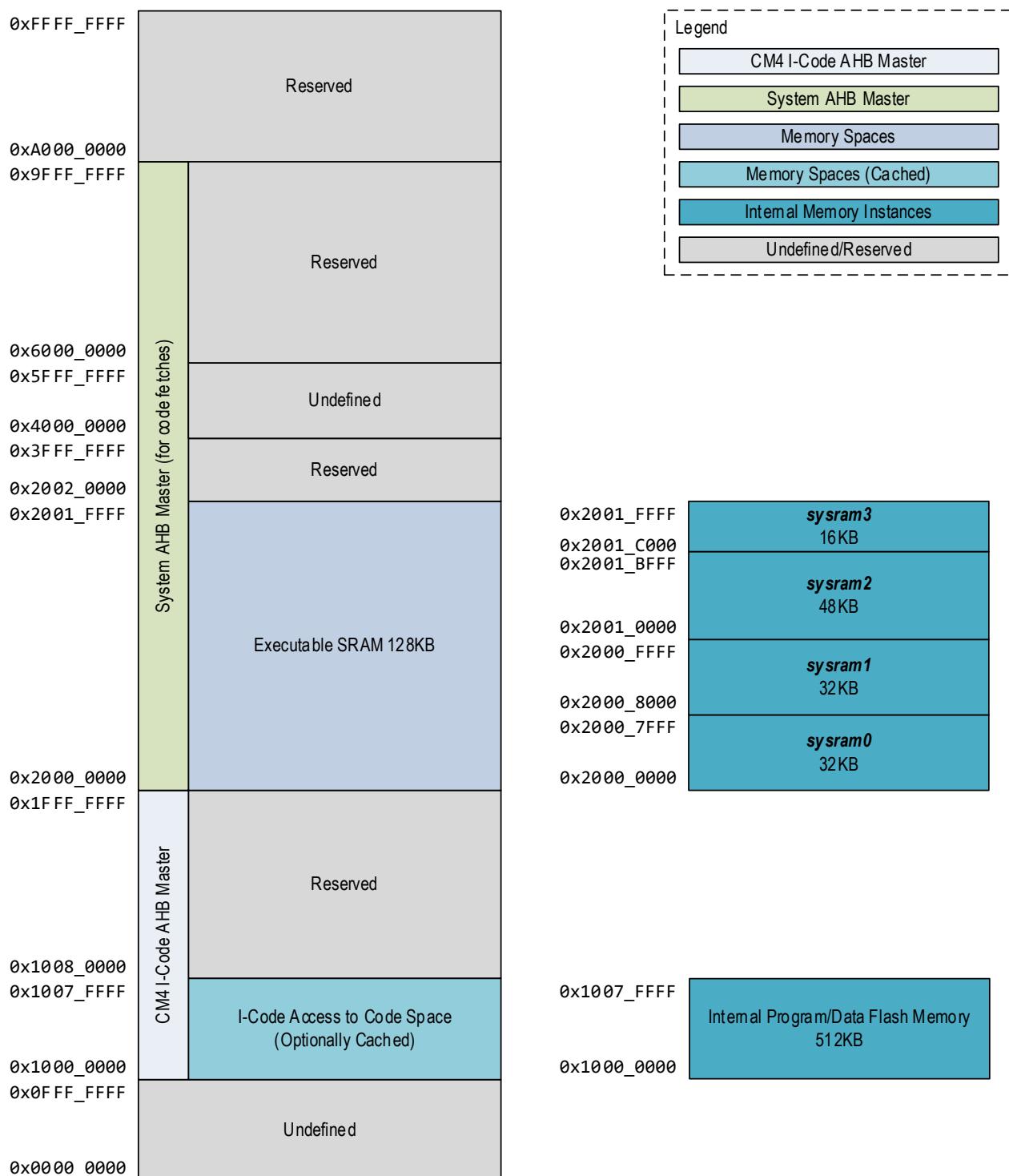
## 2. Memory, Register Mapping, and Access

### 2.1 Memory, Register Mapping, and Access Overview

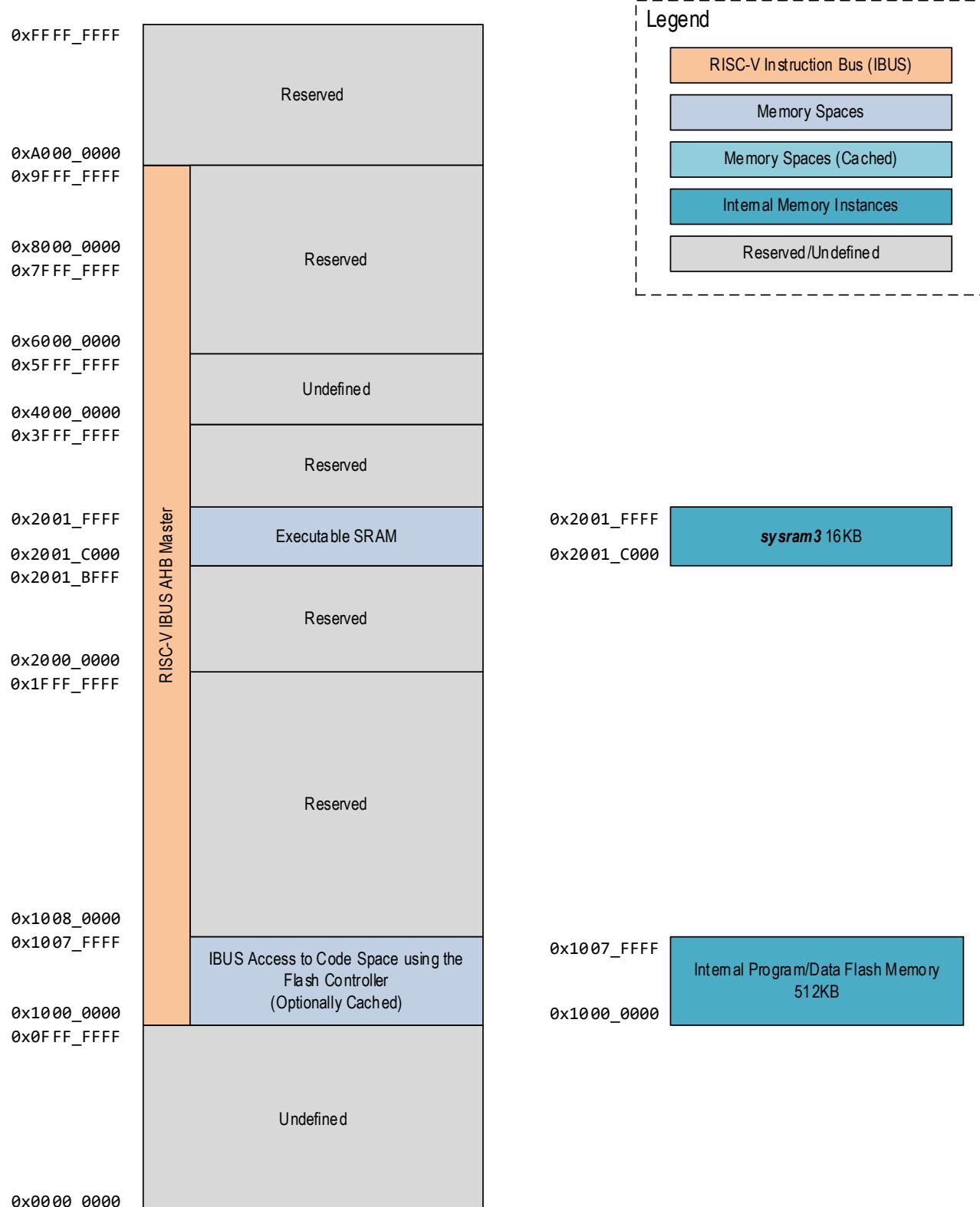
The Arm Cortex-M4 architecture defines a standard memory space for unified code and data access. This memory space is addressed in units of single bytes but is most typically accessed in 32-bit (4 byte) units. It may also be accessed, depending on the implementation, in 8-bit (1 byte) or 16-bit (2 byte)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

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

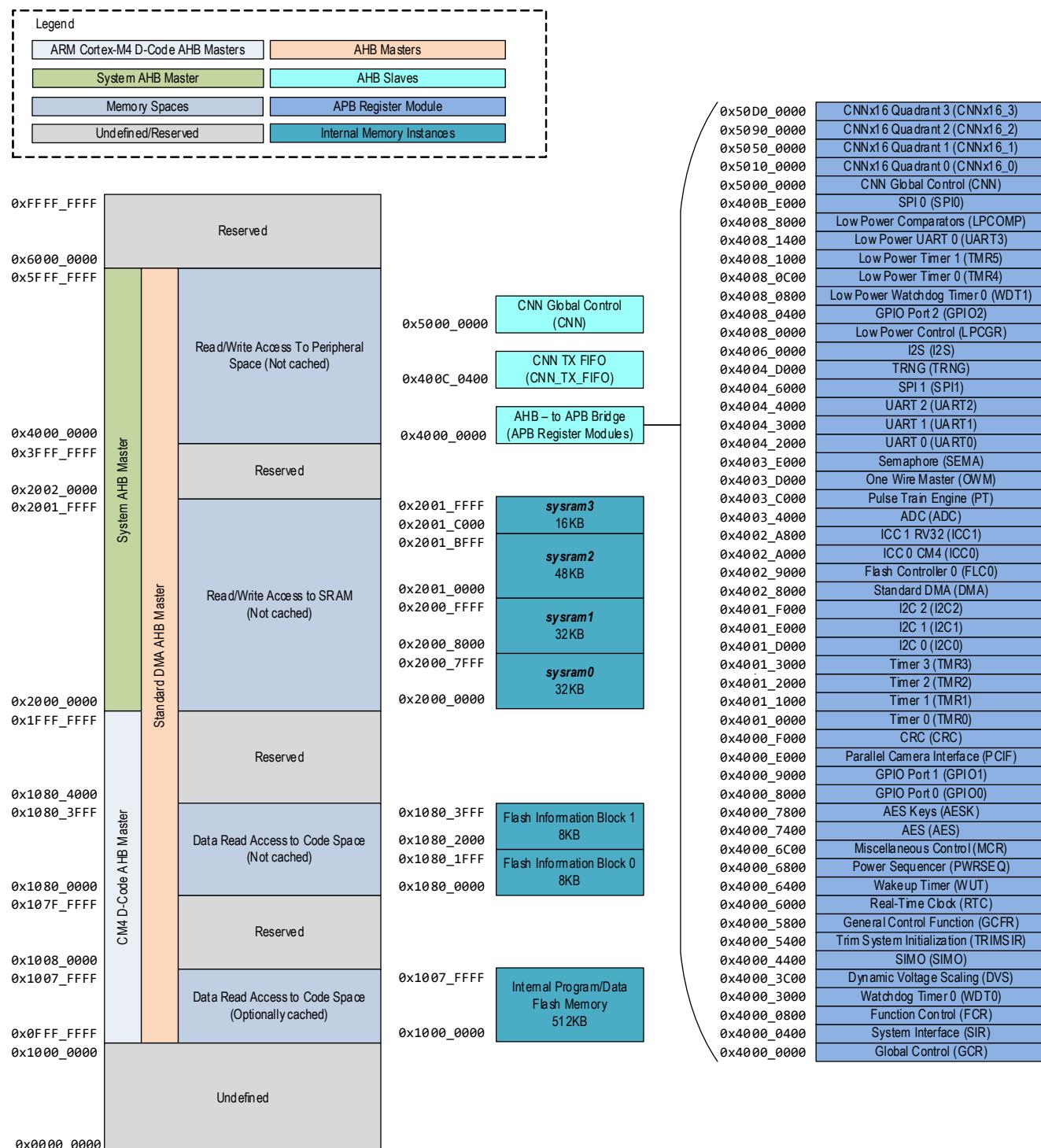
*Figure 2-1: CM4 Code Memory Mapping*



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

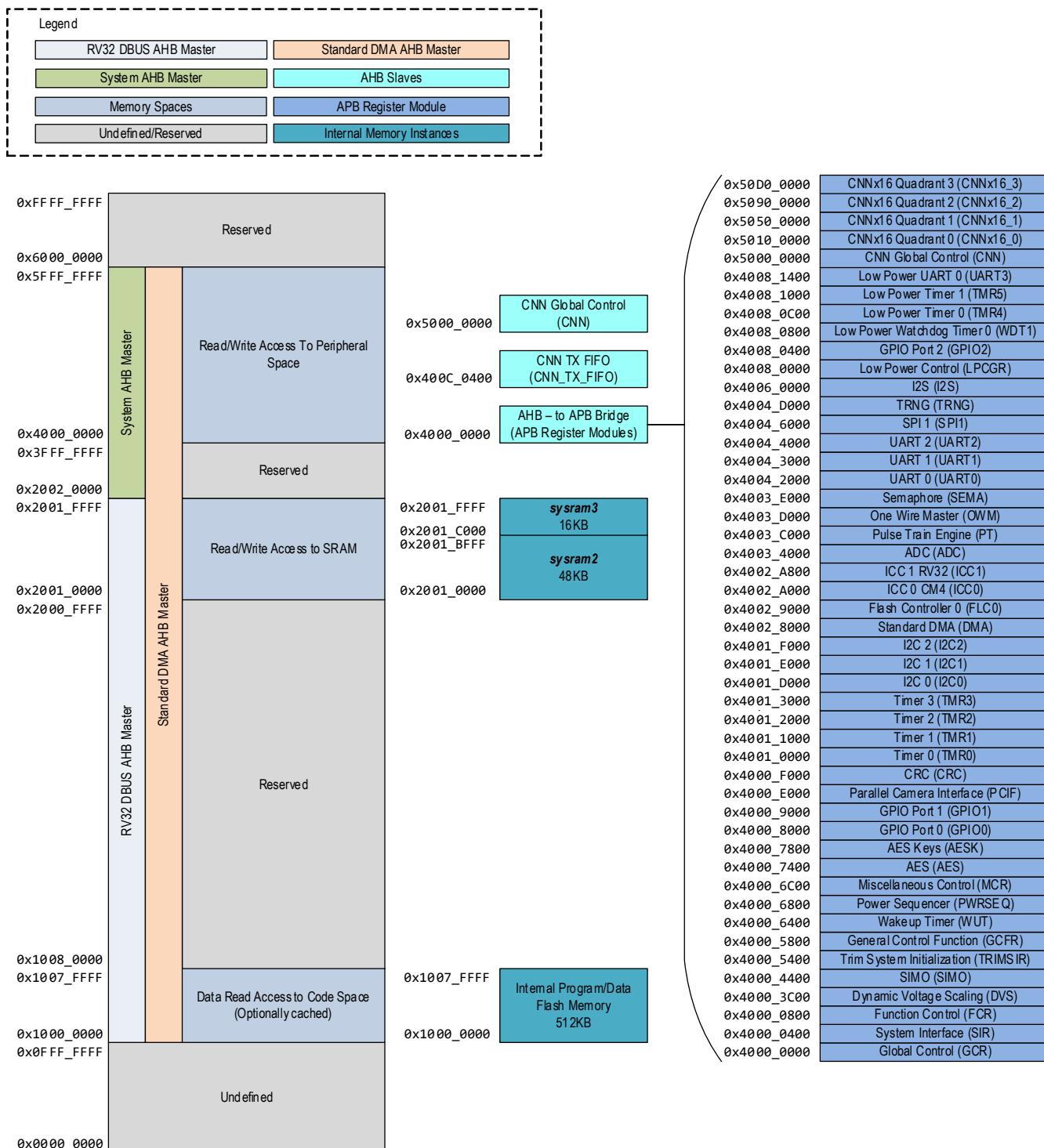


*Figure 2-3: CM4 Peripheral and Data Memory Mapping*



**Preliminary Draft 05/21/2021**

*Figure 2-4: RV32 Peripheral and Data Memory Mapping*



## 2.2 Field Access Definitions

All fields accessible by user software have distinct access capabilities. Each register table contained in this user guide has an access type defined for each field. The definition of each field access type is presented in *Table 2-1*.

*Table 2-1: Field Access Definitions*

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

## 2.3 Standard Memory Regions

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

### 2.3.1 Code Space

The code space area of memory is designed to contain the primary memory used for code execution by the device. This memory area is defined from byte address range 0x0000 0000 to 0x1FF FFFF (0.5GB maximum). Two different standard core bus masters are used by the Cortex-M4 core and Arm debugger to access this memory area. The I-Code AHB bus master is used for instruction decode fetching from code memory, while the D-Code AHB bus master is used for data

fetches from code memory. This is arranged so that data fetches avoid interfering with instruction execution. Additionally, the RV32 uses the D-BUS to access code memory in this area and the I-Bus to access data fetches from the code memory.

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

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

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

### **2.3.2 Internal Cache Memory**

The MAX78000 includes a dedicated unified internal cache controller with 16,384 Bytes of cache memory for the CM4 core (ICC0). Optionally, *sysram3* can be used as a unified internal cache controller for the RV32 (ICC1).

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. Refer to the section [Unified Internal Cache Controller](#) for detailed instructions on enabling the unified internal cache controllers.

### **2.3.3 Information Block Flash Memory**

The information block is a separate area of the internal flash memory and is 16,384 Bytes. The information block is used to store trim settings (option configuration and analog trim) as well as other nonvolatile device-specific information. The information block also contains the device's unique serial number (USN). The USN is a 104-bit field. USN bits 0 thru 7 contain the die revision.

*Figure 2-5: Unique Serial Number Format*

	Address	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
Address	0x10800000	USN bits 16 - 0																													
	0x10800004	x	USN bits 47-17																												
	0x10800008	USN bits 64 - 48																													
	0x1080000C	x	USN bits 95 - 65																												
	0x10800010	x	x	x	x	x	x	x	x	USN bits 103 - 96																					

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

1. Write 0x3A7F 5CA3 to [\*FLCn\\_ACNTL\*](#).
2. Write 0xA1E3 4F20 to [\*FLCn\\_ACNTL\*](#).
3. Write 0x9608 B2C1 to [\*FLCn\\_ACNTL\*](#).
4. The information block is now read-only accessible.

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

### 2.3.4 SRAM Space

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

The MAX78000 CM4's data memory mapping is illustrated in [Figure 2-1](#). The MAX78000 RV32's data memory mapping is illustrated in [Figure 2-4](#).

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

The SRAM memory area contains the main system SRAM. The size of the internal general-purpose data SRAM is 128KB. The SRAM is divided into four blocks and consists of the contiguous address range from 0x2000 0000 to 0x2001 FFFF. The SRAM area on the MAX78000 can be used for data storage and code execution by the CM4. The RV32 is limited to use of *sysram2* and *sysram3* for code and data storage in SRAM.

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

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

*Table 2-2: System SRAM Configuration*

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

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

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

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

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

### 2.3.5 Peripheral Space

The peripheral space area of memory is intended for mapping of control registers, internal buffers/working space, and other features needed for the firmware control of non-core peripherals. It is defined from byte address range 0x4000 0000

to 0x5FFF FFFF (0.5GB maximum). On the MAX78000, all device-specific module registers are mapped to this memory area, as well as any local memory buffers or FIFOs which are required by modules.

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

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

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

### 2.3.6 AES Key and Working Space Memory

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

### 2.3.7 External RAM Space

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

*The MAX78000 does not implement this memory area.*

### 2.3.8 External Device Space

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

*The MAX78000 does not implement this memory area.*

### 2.3.9 System Area (Private Peripheral Bus)

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

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

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

### 2.3.10 System Area (Vendor Defined)

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

*The MAX78000 does not implement this memory region.*

## 2.4 AHB Interfaces

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

### 2.4.1 Arm Core AHB Interfaces

#### 2.4.1.1 I-Code

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

#### 2.4.1.2 D-Code

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

#### 2.4.1.3 System

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

## 2.4.2 AHB Slaves

### 2.4.2.1 Standard DMA

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

### 2.4.2.2 CNN and CNN TX FIFO

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

### 2.4.2.3 SPI0

The SPI0 AHB Slave has access to all non-core memory areas accessible by the system bus. SPI0 does not have access to the internal flash memory or information blocks.

## 2.4.3 AHB Slave Base Address Map

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

*Table 2-3: AHB Slave Base Address Map*

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

## 2.5 Peripheral Register Map

### 2.5.1 APB Peripheral Base Address Map

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

*Table 2-4: APB Peripheral Base Address Map*

Peripheral Register Name	Register Prefix	APB Base Address	APB End Address
Global Control	GCR_	0x4000 0000	0x4000 03FF
System Interface	SI_	0x4000 0400	0x4000 07FF
Function Control	FCR_	0x4000 0800	0x4000 0BFF
Watchdog Timer 0	WDT0_	0x4000 3000	0x4000 33FF
Dynamic Voltage Scaling Controller	DVS_	0x4000 3C00	0x4000 3C3F
Single Input Multiple Output	SIMO_	0x4000 4400	0x4000 47FF
Trim System Initialization	TRIMSIR_	0x4000 5400	0x4000 57FF
General Control Function	GCFR_	0x4000 5800	0x4000 5BFF
Real time Clock	RTC_	0x4000 6000	0x4000 63FF
Wakeup Timer	WUT_	0x4000 6400	0x4000 67FF
Power Sequencer	PWRSEQ_	0x4000 6800	0x4000 6BFF
Miscellaneous Control	MCR_	0x4000 6C00	0x4000 6FFF
AES	AES_	0x4000 7400	0x4000 77FF
AES Key	AESK_	0x4000 7800	0x4000 7BFF
GPIO Port 0	GPIO0_	0x4000 8000	0x4000 8FFF
GPIO Port 1	GPIO1_	0x4000 9000	0x4000 9FFF
Parallel Camera Interface	PCIF_	0x4000 E000	0x4000 EFFF
CRC	CRC_	0x4000 F000	0x4000 FFFF
Timer 0	TMR0_	0x4001 0000	0x4001 OFFF
Timer 1	TMR1_	0x4001 1000	0x4001 1FFF
Timer 2	TMR2_	0x4001 2000	0x4001 2FFF
Timer 3	TMR3_	0x4001 3000	0x4001 3FFF
I <sup>2</sup> C 0	I2C0_	0x4001 D000	0x4001 DFFF
I <sup>2</sup> C 1	I2C1_	0x4001 E000	0x4001 EFFF
I <sup>2</sup> C 2	I2C2_	0x4001 F000	0x4001 FFFF
Standard DMA	DMA_	0x4002 8000	0x4002 8FFF
Flash Controller 0	FLCO_	0x4002 9000	0x4002 93FF
Instruction-Cache Controller 0 (CM4)	ICCO_	0x4002 A000	0x4002 A7FF
Instruction Cache Controller 1 (RV32)	ICC1_	0x4002 A800	0x4002 AFFF
ADC	ADC_	0x4003 4000	0x4003 4FFF
Pulse Train Engine	PT_	0x4003 C000	0x4003 C09F
One Wire Master	OWM0_	0x4003 D000	0x4003 DFFF
Semaphore	SEMA_	0x4003 E000	0x4003 EFFF
UART 0	UART0_	0x4004 2000	0x4004 2FFF
UART 1	UART1_	0x4004 3000	0x4004 3FFF
UART 2	UART2_	0x4004 4000	0x4004 4FFF

Peripheral Register Name	Register Prefix	APB Base Address	APB End Address
SPI1	SPI1_	0x4004 6000	0x4004 7FFF
TRNG	TRNG_	0x4004 D000	0x4004 DFFF
I <sup>2</sup> S	I2S_	0x4006 0000	0x4006 OFFF
Low Power General Control	LPGCR_	0x4008 0000	0x4008 03FF
GPIO Port 2	GPIO2_	0x4008 0400	0x4008 05FF
Low Power Watchdog Timer 0 (WDT1)	WDT1_	0x4008 0800	0x4008 0BFF
Low Power Timer 4	TMR4_	0x4008 0C00	0x4008 OFFF
Low Power Timer 5	TMR5_	0x4008 1000	0x4008 13FF
Low Power UART 0 (UART3)	UART3_	0x4008 1400	0x4008 17FF
Low Power Comparator	LPCOMP_	0x4008 8000	0x4008 83FF
CNN Global Control	CNN_	0x5000 0000	0x500F FFFF
CNNx16 Quadrant 0	CNNx16_0_	0x5010 0000	0x504F FFFF
CNNx16 Quadrant 1	CNNx16_1_	0x5050 0000	0x508F FFFF
CNNx16 Quadrant 2	CNNx16_2_	0x5090 0000	0x50CF FFFF
CNNx16 Quadrant 3	CNNx16_3_	0x50D0 0000	0x510F FFFF

## 2.6 Error Correction Coding (ECC) Module

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

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

If only one bit is determined to be corrupt, reads will contain the “corrected” value. Reading memory does not correct the 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 only occurs during a “read” operation, it is recommended that the application periodically “reads” critical memory so that errors can be identified and corrected.

### 2.6.1 SRAM

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

### 2.6.2 Limitations

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

## 3. System, Power, Clocks, Reset

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

### 3.1 Oscillator Sources

#### 3.1.1 100MHz Internal Primary Oscillator (IPO)

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

The IPO can optionally be powered down in *LPM* by setting the *GCR\_PM.ipo\_pd* field to 1.

The IPO can be selected as the *SYS\_OSC*. To use this oscillator as the *SYS\_OSC*, the following steps must be followed:

1. Enable the IPO by setting *GCR\_CLKCTRL.ipo\_en* to 1.
2. Wait until the *GCR\_CLKCTRL.ipo\_rdy* field reads 1 indicating the IPO is operating.
3. Set *GCR\_CLKCTRL.sysclk\_sel* to 4.
4. Wait until the *GCR\_CLKCTRL.sysclk\_rdy* field reads 1. The IPO is now operating as the *SYS\_OSC*.

#### 3.1.2 60MHz Internal Secondary Oscillator (ISO)

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

The following steps show how to enable the ISO and select it as the *SYS\_OSC*.

1. Enable the ISO by setting *GCR\_CLKCTRL.iso\_en* to 1.
2. Wait until the *GCR\_CLKCTRL.iso\_rdy* field reads 1 indicating the ISO is operating.
3. Set *GCR\_CLKCTRL.sysclk\_sel* to 0.
4. Wait until the *GCR\_CLKCTRL.sysclk\_rdy* field reads 1. The ISO is now operating as the *SYS\_OSC*.

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

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

The frequency of this oscillator is configurable to 8kHz, 16kHz or 30kHz. Use the *TRIMSR\_INRO.inro\_sel* field to select the desired frequency. On a POR or System Reset, the frequency defaults to 30kHz.

The following steps show how to set the INRO as the *SYS\_OSC*.

1. Verify the *GCR\_CLKCTRL.inro\_rdy* field reads 1.
2. Set *GCR\_CLKCTRL.sysclk\_sel* to 3.
3. Wait until the *GCR\_CLKCTRL.sysclk\_rdy* field reads 1. The INRO is now operating as the *SYS\_OSC*.

### 3.1.4 7.3728MHz Internal Baud Rate Oscillator (IBRO)

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

The following steps show how to enable the IBRO and select it as the SYS\_OSC.

1. Wait until the [\*GCR\\_CLKCTRL.ibro\\_rdy\*](#) field reads 1 indicating the IBRO is operating.
2. Set [\*GCR\\_CLKCTRL.sysclk\\_sel\*](#) to 5.
3. Wait until the [\*GCR\\_CLKCTRL.sysclk\\_rdy\*](#) field reads 1. The IBRO is now operating as the SYS\_OSC.

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

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

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

The following steps show how to enable the ERTCO and select it as the SYS\_OSC.

1. Enable the ERTCO by setting [\*GCR\\_CLKCTRL.ertco\\_en\*](#) to 1.
2. Wait until the [\*GCR\\_CLKCTRL.ertco\\_rdy\*](#) field reads 1 indicating the ERTCO is operating.
3. Set [\*GCR\\_CLKCTRL.sysclk\\_sel\*](#) to 6.
4. Wait until the [\*GCR\\_CLKCTRL.sysclk\\_rdy\*](#) field reads 1. The ERTCO is now operating as the SYS\_OSC.

## 3.2 System Oscillator (SYS\_OSC)

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

*Table 3-1: Available System Oscillators*

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

### 3.2.1 System Oscillator Selection

Set the system oscillator using the [\*GCR\\_CLKCTRL.sysclk\\_sel\*](#) field. Prior to selecting an oscillator as the system oscillator, the oscillator source must first be enabled and ready. Refer to each individual Oscillator Source's detailed description for the required steps to enable the oscillator and select it as the system oscillator.

When the [GCR\\_CLKCTRL.sysclk\\_sel](#) is modified, hardware clears the [GCR\\_CLKCTRL.sysclk\\_rdy](#) field and there is a delay until the switchover is complete. When the switchover to the selected SYS\_OSC is complete, the [GCR\\_CLKCTRL.sysclk\\_rdy](#) field is set to 1 by hardware. Application firmware must verify the switchover is complete before continuing operation.

### 3.2.2 System Clock (SYS\_CLK)

The selected SYS\_OSC is the input to the system oscillator divider to generate the System Clock (SYS\_CLK). The system clock divider divides the selected SYS\_OSC by the [GCR\\_CLKCTRL.sysclk\\_div](#) field as shown in [Equation 3-1](#).

*Equation 3-1: System Clock Scaling*

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

[GCR\\_CLKCTRL.sysclk\\_div](#) is selectable from 0 to 7, resulting in divisors of 1, 2, 4, 8, 16, 32, 64 or 128.

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

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

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

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

- Power-On Reset
- System Reset

Oscillator settings are *not* reset during:

- Soft Reset
- Peripheral Reset

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

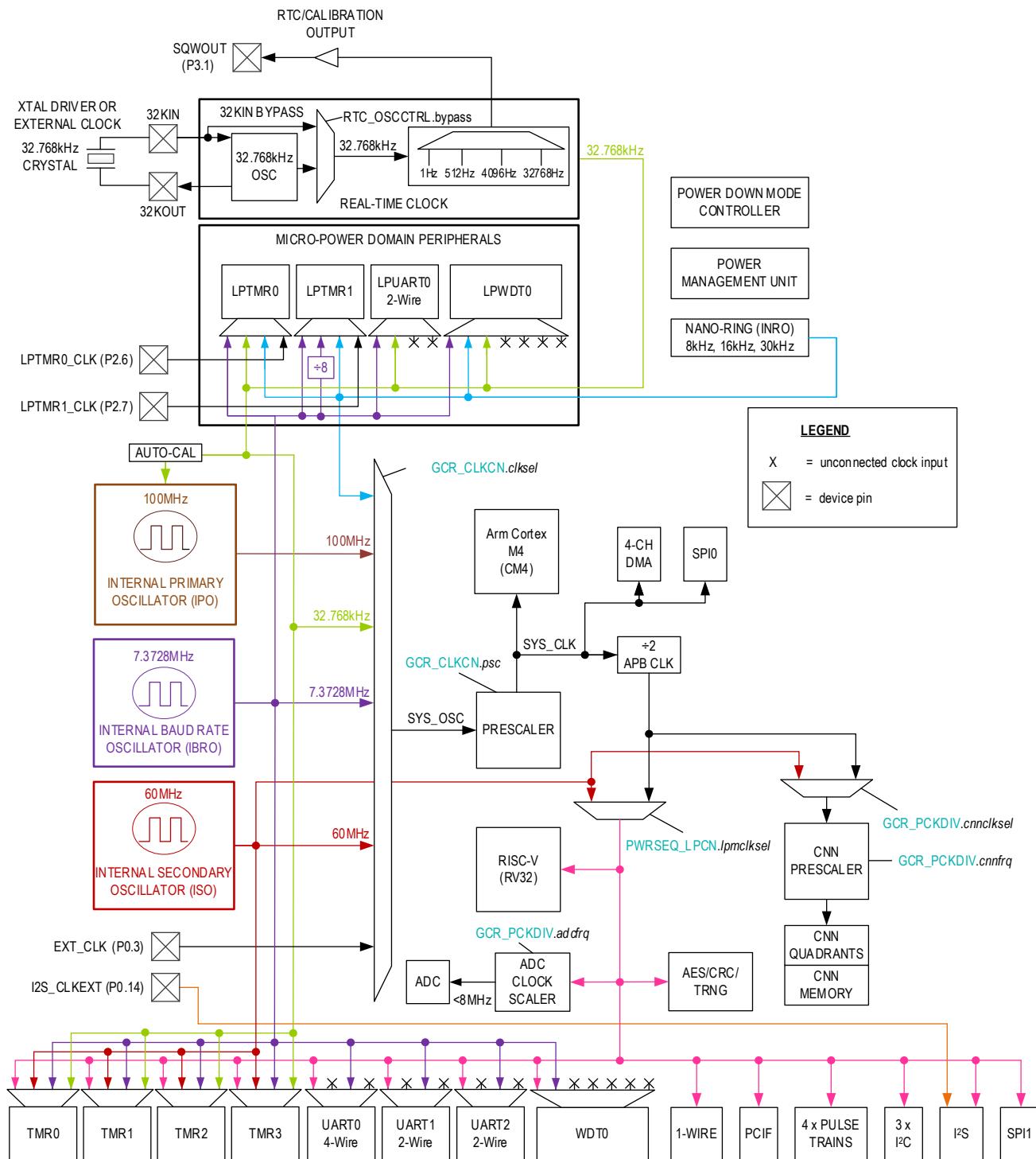
*Note: A Watchdog Timer Reset performs a System Reset.*

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

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

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

*Figure 3-1: MAX78000 Clock Block Diagram*



**Preliminary Draft 05/21/2021**

### 3.3 Operating Modes

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

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

#### 3.3.1 ACTIVE Mode

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

Each of the peripherals can be individually enabled during active mode or powered down. The CNN and each of the four CNNx16\_n Processor Arrays and their associated memories can be powered down or set to active mode. See section CNN Configuration for details on enabling the CNNx16\_n Processor Arrays and their associated RAM.

#### 3.3.2 Low-power Modes

##### 3.3.2.1 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 are individually configurable for power down
- Standard DMA is available for use
- All peripherals are on unless explicitly disabled prior to entering SLEEP

###### 3.3.2.1.1 Entering SLEEP

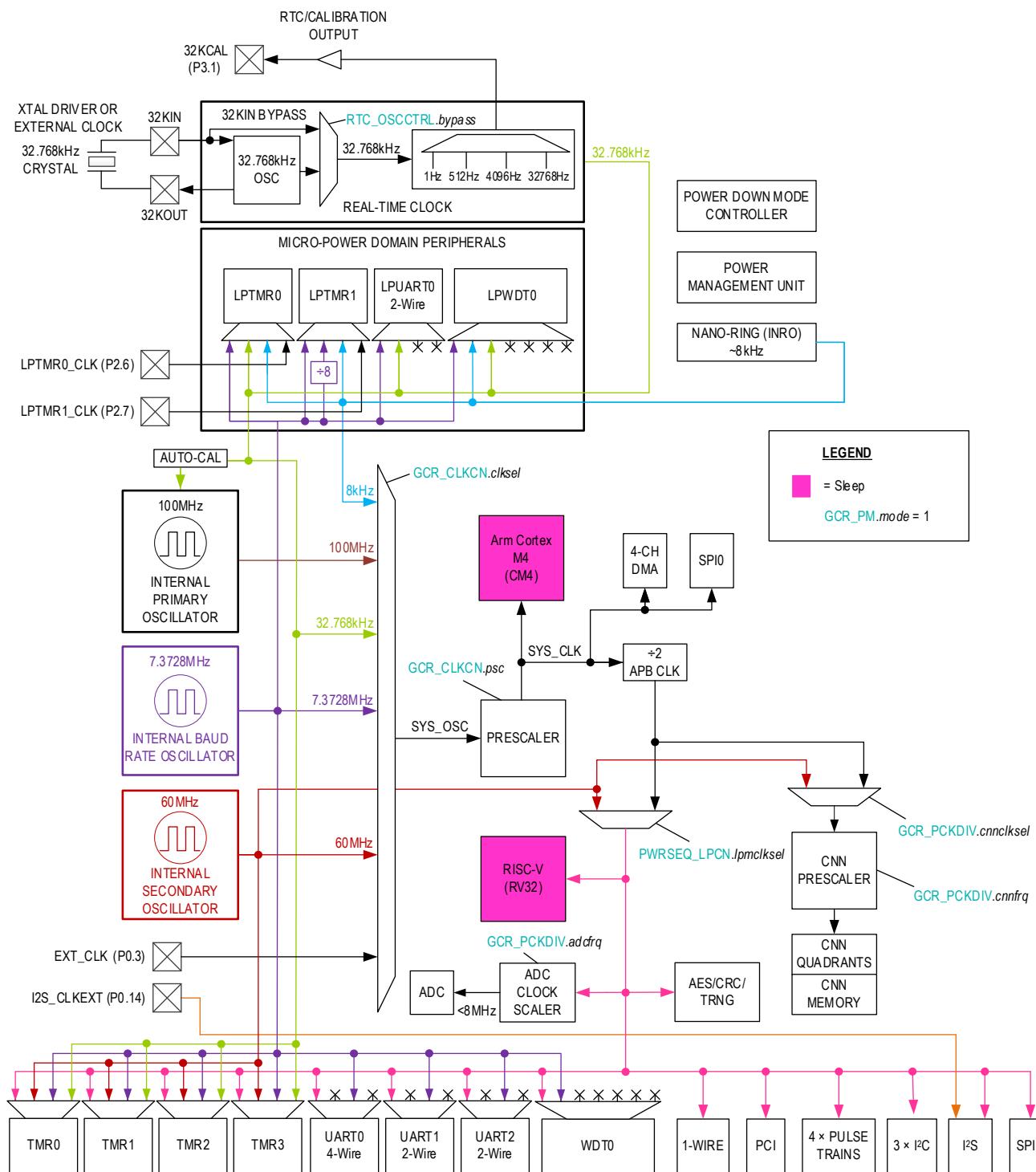
Entering SLEEP requires both the CM4 and RV32 to cooperate to enter SLEEP. Synchronization is necessary for deterministic entry into SLEEP. Two methods are described below allowing either core to request entry into SLEEP. Both methods use the Semaphore peripheral interrupt to communicate between the 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 a 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 3-2: SLEEP Mode Clock Control*



### 3.3.2.2 LPM

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

- The CM4, *sysram0*, and *sysram1* are in state retention
- The CNN quadrants and memory are active and configurable.
- The RV32 can access the SPI, all UARTS, all Timers, I<sup>2</sup>C, 1-Wire, Timers, Pulse Train Engine, I<sup>2</sup>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\_GPO* and *PWRSEQ\_GP1* registers retain state
- Choose the system PCLK or ISO as the clock source for the RV32 and all Peripherals
  - ◆ *PWRSEQ\_LPCN.lpmclksel* defaults to use ISO during *LPM*, setting this field to 1 uses the PCLK
- The following oscillators are powered down:
  - ◆ IPO
- The following oscillators are enabled:
  - ◆ IBRO
  - ◆ ERTCO
  - ◆ INRO
  - ◆ ISO

#### 3.3.2.2.1 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*.

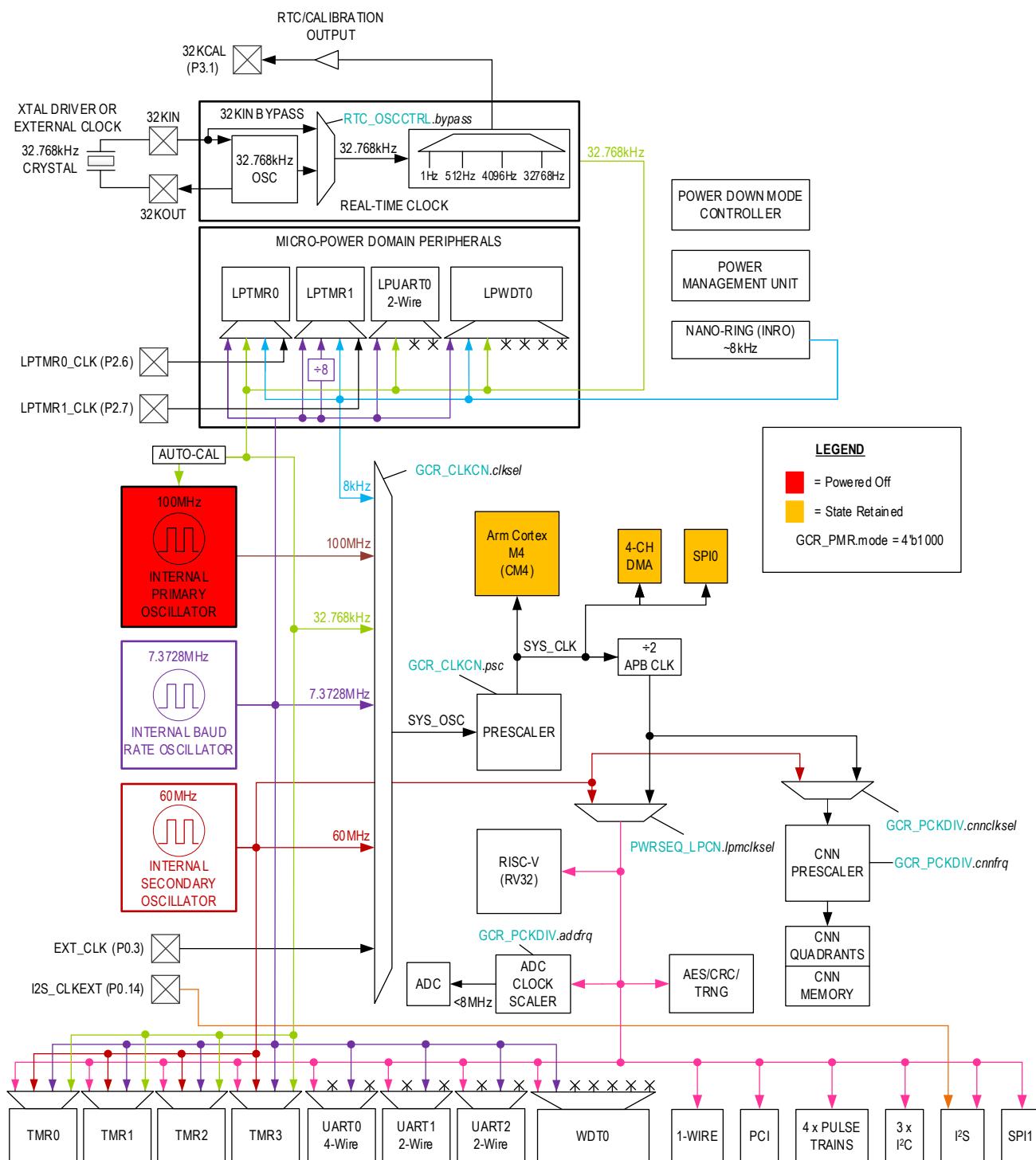
When the CM4 puts itself into *deep sleep*, the device will automatically enter *LPM* and hardware sets the *GCR\_PM.mode* to *LPM*. To place the CM4 in *LPM* mode in firmware, perform the following instructions.

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

If the RV32 requests the CM4 to enter *LPM* mode 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 3-3: Low Power Mode (LPM) Clock and State Retention Diagram*



Preliminary Draft 05/21/2021

### 3.3.2.3 UPM

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

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

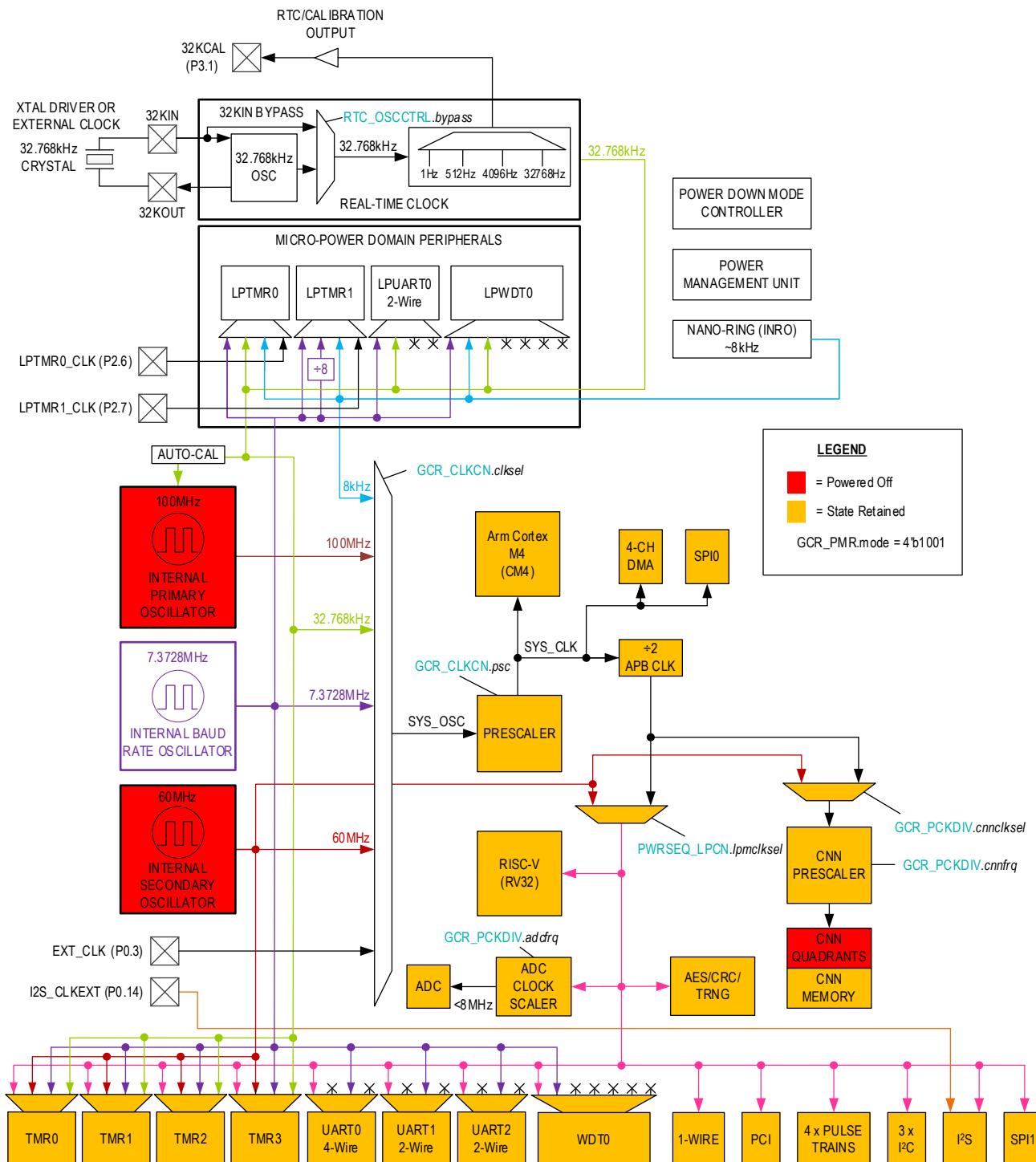
#### 3.3.2.3.1 Entering UPM

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 into *UPM*.

Alternatively, the CM4 can initiate the request to enter *UPM* by sending the request to the RV32 using *Multiprocessor Communications*. The RV32 confirms the request 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 3-4: Micro Power Mode (UPM) Clock and State Retention Block Diagram*



Preliminary Draft 05/21/2021

### 3.3.2.4 STANDBY

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

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

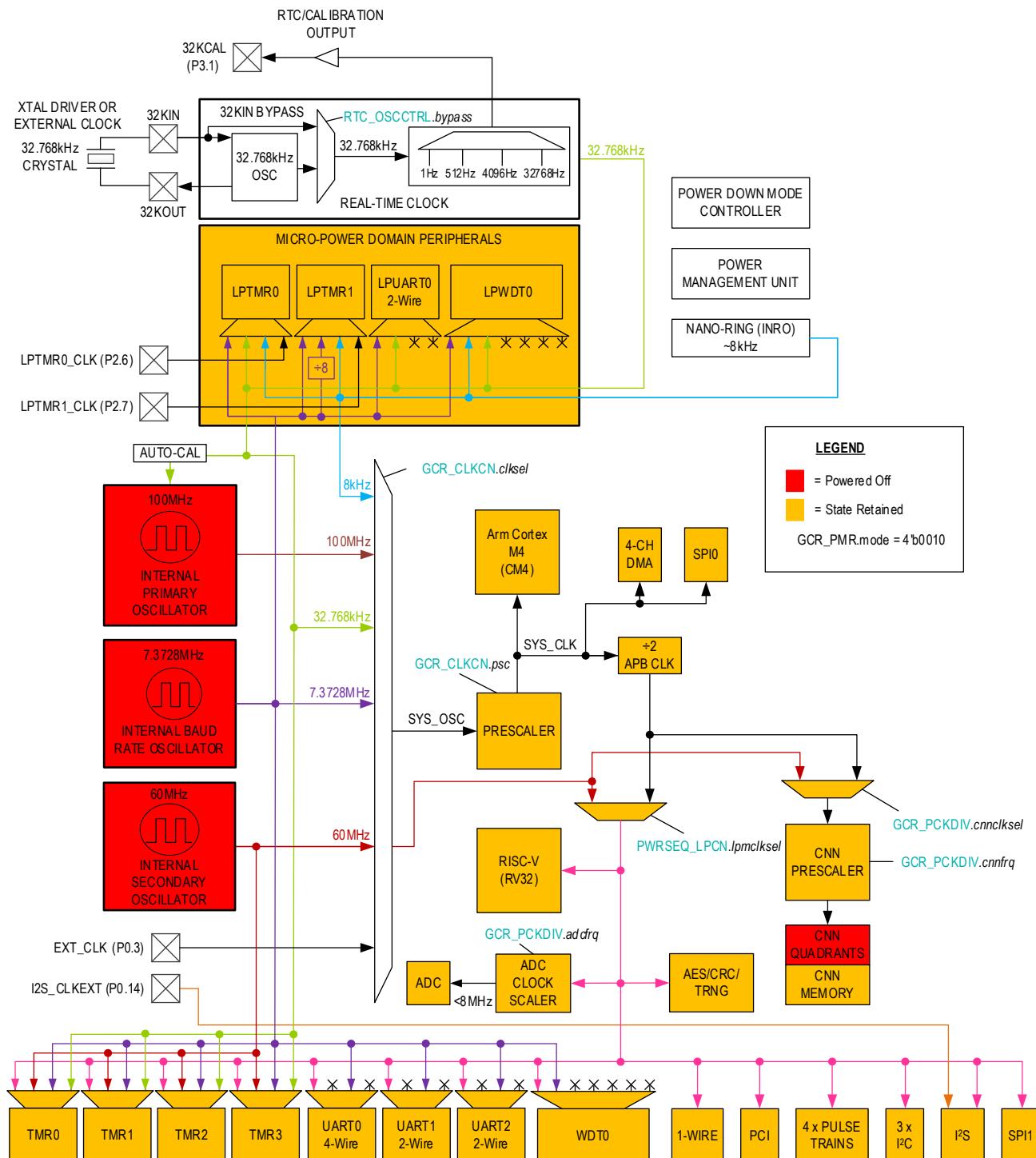
#### 3.3.2.4.1 Entering STANDBY

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



### 3.3.2.5 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 to be state retained as shown in [Table 3-3](#).
- User configurable CNN memory retention
- All peripherals are powered off
- The following oscillators are powered down:
  - ◆ IPO
  - ◆ ISO
  - ◆ IBRO
  - ◆ INRO
- The following oscillators are enabled:
  - ◆ ERTCO (The RTC peripheral can be turned off, but not the oscillator)

*Table 3-3 System RAM Retention in BACKUP Mode*

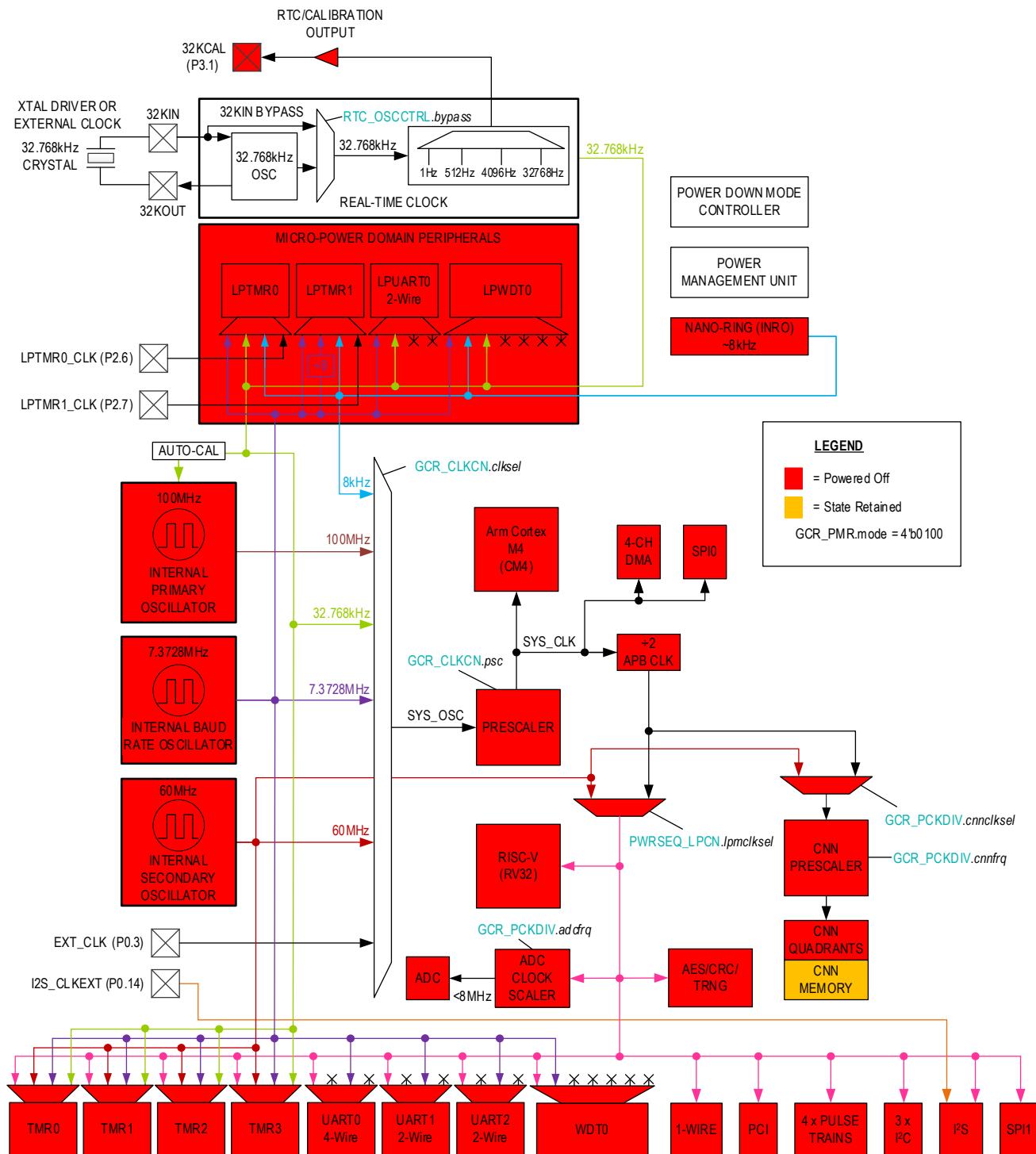
RAM Block #	Size	State Retention Control
<i>sysram0</i>	<i>32KB + ECC if enabled</i>	<i>PWRSEQ_LPCN.ramret0</i>
<i>sysram1</i>	<i>32KB</i>	<i>PWRSEQ_LPCN.ramret1</i>
<i>sysram2</i>	<i>48KB</i>	<i>PWRSEQ_LPCN.ramret2</i>
<i>sysram3</i>	<i>16KB</i>	<i>PWRSEQ_LPCN.ramret3</i>

### 3.3.2.5.1 Entering BACKUP

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

Either core can set *GCR\_PM.mode* to *BACKUP* and the device immediately enters *BACKUP*.

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



Preliminary Draft 05/21/2021

### 3.3.2.6 PDM

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

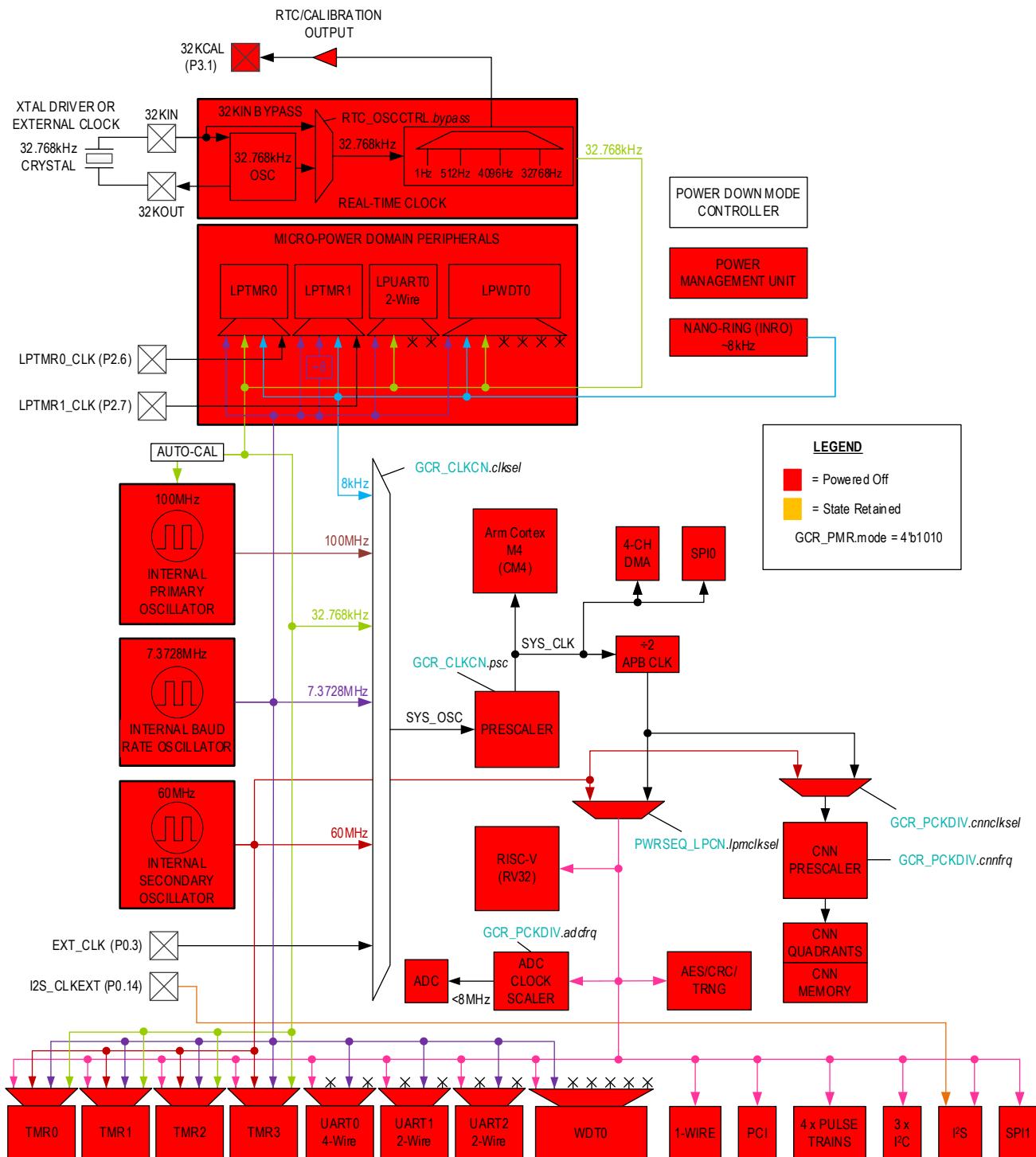
- The CM4 and RV32 are powered off
- All peripherals and all RAMs are powered down
- All oscillators are powered down
- There is no data retention in this mode, but values in the flash are preserved
- V<sub>REGI</sub> POR voltage monitor is operational.
- Exit from PDM is possible through an external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

#### 3.3.2.6.1 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 3-7: Power Down Mode (PDM) Clock and State Retention Block Diagram*



Preliminary Draft 05/21/2021

### 3.4 Operating Modes Wakeup Sources

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

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

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

Operating Mode	Any Peripheral Interrupts	External Reset	RV32	CNN	CNN FIFO	SPI1	SPI0	I2S	I2C2	I2C1	I2C0	LPUART0 (UART3)	UART2	UART1	UART0	LPTMR1 (TMR5)	LPTMR0 (TMR4)	TMR3	TMR2	TMR1	TMR0	LPWDTO (WDT1)	WDT0	LPCOMP3	LPCMOP2	LPCMOP1	COMP0	RTC	WUT	GPIO3	GPIO2	GPIO1	GPIO0
<i>SLEEP</i>	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
<i>LPM</i>		✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
<i>UPM</i>		✓										✓			✓	✓					✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
<i>STANDBY</i>		✓																								✓	✓	✓	✓	✓	✓		
<i>BACKUP</i>		✓																								✓	✓	✓	✓	✓	✓		
<i>PDM</i>		✓																															

### 3.5 Device Resets

Four device resets are available:

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

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

Contents of the Always-On Domain (AoD) are reset only on power-cycling V<sub>COREA</sub>, V<sub>COREB</sub>, V<sub>DDA</sub>, V<sub>DDIOH</sub> or V<sub>REGI</sub>.

Each of the on-chip peripherals can also be reset to their POR default state using the two reset registers [\*GCR\\_RST0\*](#) and [\*GCR\\_RST1\*](#).

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

*Table 3-5: Operating Mode and Clock Status*

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

### 3.5.1 Peripheral Reset

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

To start a Peripheral Reset, set [\*GCR\\_RST0.periph\*](#) = 1. The reset will be completed immediately upon setting [\*GCR\\_RST0\*](#) = 1.

### 3.5.2 Soft Reset

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

To start a Soft Reset, set [\*GCR\\_RST0.soft\*](#) = 1. The reset will be completed immediately upon setting [\*GCR\\_RST0.soft\*](#) = 1.

### 3.5.3 System Reset

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

A watchdog timer reset event initiates a System Reset. To start a System Reset from firmware, set [\*GCR\\_RST0.sys\*](#) = 1.

### 3.5.4 Power-On Reset

A POR resets everything in the device to its default state. A POR results from  $V_{COREA}$ ,  $V_{COREB}$ ,  $V_{DDA}$  or  $V_{REGI}$  falling below their reset voltage level. Refer to the [\*MAX78000 data sheet\*](#) for details of the reset voltage levels.

## 3.6 Unified Internal Cache Controllers

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

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

### 3.6.1 Enabling the Internal Cache Controllers

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

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

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

Perform the following steps to enable the ICCn:

1. Set the [\*ICCn\\_CTRL.en\*](#) to 0, ensuring the cache is invalidated when enabled.
2. Set [\*ICCn\\_CTRL.en\*](#) to 1.
3. Read [\*ICCn\\_CTRL.rdy\*](#) until it returns 1.
4. Zeroize the ICC instance by setting [\*GCR\\_MEMZ.icc0\*](#) or [\*GCR\\_MEMZ.icc1\*](#) to 1.

### 3.6.2 Disabling the ICC

Disable an ICC instance by setting *ICCn\_CTRL.en* to 0.

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

### 3.6.3 Invalidating the ICC Cache and Tag RAM

The System Configuration Register (*GCR\_SYSCTRL*) includes a field for flushing the ICC0 cache. Setting *GCR\_SYSCTRL.icc0\_flush* to 1 flushes the ICC0 16KB cache and the tag RAM.

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

### 3.6.4 Flushing the ICC

Flush ICC0 using the System Configuration Register (*GCR\_SYSCTRL*). Set *GCR\_SYSCTRL.icc0\_flush* to 1 to immediately flush the contents of the 16KB cache and tag RAM.

Flush ICC1 using the RV32 Control Register (*FCR\_URVCTRL*). Set *FCR\_URVCTRL.icc1\_flush* to 1 to immediately flush the contents of the 16KB cache and tag RAM.

### 3.6.5 Internal Cache Control Registers (ICC)

See *Table 2-4* for the base address of this peripheral/module. See *Table 2-1* 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 3-6: Instruction Cache Controller Register Summary*

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

### 3.6.6 ICC0 Register Details

*Table 3-7: ICC0 Cache Information Register*

ICC0 Cache Information			<i>ICCn_INFO</i>	[0x0000]
Bits	Field	Access	Reset	Description
31:16	-	RO	0	<b>Reserved</b>
15:10	id	R	-	<b>Cache ID</b> Returns the ID for this cache instance.
9:6	partnum	R	-	<b>Cache Part Number</b> Returns the part number indicator for this cache instance.
5:0	relnum	R	-	<b>Cache Release Number</b> Returns the release number for this cache instance.

Table 3-8: ICC0 Memory Size Register

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

Table 3-9: ICC0 Cache Control Register

ICCO Cache Control				ICCn_CTRL	[0x0100]
Bits	Field	Access	Reset	Description	
31:17	-	R/W	-	<b>Reserved</b>	
16	rdy	R	-	<b>Ready</b> This field is cleared by hardware anytime the cache as a whole is invalidated (including a POR). Hardware automatically sets this field to 1 when the invalidate operation is complete and the cache is ready. 0: Cache invalidation in process. 1: Cache is ready. <i>Note: While this field reads 0, the cache is bypassed and reads come directly from the line fill buffer.</i>	
15:1	-	R/W	-	<b>Reserved</b>	
0	en	R/W	0	<b>Cache Enable</b> Set this field to 1 to enable the cache. Setting this field to 0 automatically invalidates the cache contents and reads are handled by the line fill buffer. 0: Disable 1: Enable	

Table 3-10: ICC0 Invalidate Register

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

## 3.7 RAM Memory Management

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

### 3.7.1 On-Chip Cache Management

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

### 3.7.2 RAM Zeroization

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

The following SRAM memories can be zeroized:

- Each of the System RAMs can be individually zeroized by setting the respective [GCR\\_MEMZ](#) bit:
  - ◆ [GCR\\_MEMZ.ram0](#)
  - ◆ [GCR\\_MEMZ.ram0ecc](#)
  - ◆ [GCR\\_MEMZ.ram1](#)
  - ◆ [GCR\\_MEMZ.ram2](#)
  - ◆ [GCR\\_MEMZ.ram3](#)
- ICC0 16KB Cache
  - ◆ [GCR\\_MEMZ.icc0](#)
- ICC1 16KB Cache, if enabled
  - ◆ [GCR\\_MEMZ.icc1](#)
- Each of the CNNx16n Processor Arrays support zeroizing the tornado RAM, mask RAM, bias RAM, and data SRAM:
  - ◆ [CNNx16\\_n\\_TEST.tramz](#) set to 1 to zero, read [CNNx16\\_n\\_TEST.tallzdone](#) until 1 for completion
  - ◆ [CNNx16\\_n\\_TEST.mramz](#) set to 1 to zero, read [CNNx16\\_n\\_TEST.mallzdone](#) until 1 for completion
  - ◆ [CNNx16\\_n\\_TEST.bramz](#) set to 1 to zero, read [CNNx16\\_n\\_TEST.ballzdone](#) until 1 for completion
  - ◆ [CNNx16\\_n\\_TEST.sramz](#) set to 1 to zero, read [CNNx16\\_n\\_TEST.sallzdone](#) until 1 for completion

## 3.8 Miscellaneous Control Registers (MCR)

See [Table 2-4](#) for the base address of this peripheral/module. See [Table 2-1](#) 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 3-11: Miscellaneous Control Register Summary*

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

### 3.8.1 Miscellaneous Control Register Details

*Table 3-12: Error Correction Coding Enable Register*

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

Table 3-13: IPO Manual Register

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

Table 3-14: Output Enable Register

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

Table 3-15: Comparator 0 Control Register

Comparator 0 Control			MCR_CMPO_CTRL		[0x000C]
Bits	Name	Access	Reset	Description	
31:16	-	RO	0	<b>Reserved</b>	
15	if	R/W1C	0	<b>Comparator 0 Interrupt Flag</b> This field is set to 1 by hardware when the comparator output changes to the active state as set using the <a href="#">MCR_CMPO_CTRL.pol</a> field. Write 1 to clear this flag. 0: No interrupt 1: Interrupt occurred	

Comparator 0 Control				MCR_CMPO_CTRL	[0x000C]
Bits	Name	Access	Reset	Description	
14	out	RO	*	<b>Comparator 0 Output</b> This field is the comparator output state. 0: Output low 1: Output high	
13:7	-	RO	0	<b>Reserved</b>	
6	int_en	R/W	0	<b>Comparator 0 Interrupt Enable</b> Set this field to 1 to enable the IRQ for comparator 0. 0: Interrupt disabled 1: Interrupt enabled	
5	pol	R/W	0	<b>Comparator 0 Interrupt Polarity Select</b> 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	<b>Reserved</b>	
0	en	R/W	0	<b>Comparator 0 Enable</b> Set this field to 1 to enable the comparator 0: Comparator disabled 1: Comparator enable	

Table 3-16: Miscellaneous Control Register

Miscellaneous Control				MCR_CTRL	[0x0010]
Bits	Name	Access	Reset	Description	
31:10	-	RO	0	<b>Reserved</b>	
9	simo_rstd	R/W	0	<b>SIMO System Reset Disable</b> If this field is set, the SIMO is only reset by a Power-On Reset. When this bit is set, the VSET* stay's unchanged when exiting all low power modes. 0: The SIMO is reset by all system resets. 1: The SIMO is only reset by a Power-On Reset.	
8	simo_clkscl_en	R/W	0	<b>SIMO Clock Scaling Enable</b> Set this field to 1 to enable dynamic clock scaling to the SIMO based on load current. When enabled, the SIMO clock slows down in low power modes, reducing current consumption. 0: SIMO clock scaling disabled 1: SIMO clock scaling enabled	
7:4	-	DNM	0x01	<b>Reserved</b>	
3	ertco_en	R/W	0	<b>ERTCO Enable</b> Set this field to 1 to enable the ERTCO. 0: ERTCO disabled 1: ERTCO enabled	
2	inro_en	R/W	0	<b>INRO Enable</b> Set this field to 1 to enable the INRO. 0: INRO disabled 1: INRO enabled	

Miscellaneous Control			MCR_CTRL		[0x0010]
Bits	Name	Access	Reset	Description	
1:0	-	RO	0	Reserved	

Table 3-17: 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	<b>GPIO3 Pin 1 Input Status</b> Read this field to determine the input status of P3.1. 0: Input Low 1: Input High	
6	p31_pe	R/W	0	<b>GPIO3 Pin 1 Pull-up Enable</b> Set this bit to 1 to enable the pull-up resistor for P3.1 0: Pull-up Disabled 1: Pull-up Enabled	
5	p31_oe	R/W	0	<b>GPIO3 Pin 1 Output Enable</b> 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	<b>GPIO3 Pin 1 Data Output</b> 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	<b>GPIO3 Pin 0 Input Status</b> Read this field to determine the input status of P3.0. 0: Input Low 1: Input High	
2	p30_pe	R/W	0	<b>GPIO3 Pin 0 Pull-up Enable</b> Set this bit to 1 to enable the pull-up resistor for P3.0 0: Pull-up Disabled 1: Pull-up Enabled	
1	p30_oe	R/W	0	<b>GPIO3 Pin 0 Output Enable</b> 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	<b>GPIO3 Pin 0 Data Output</b> If p30_oe is set to 1, this field is used to control the output state of P3.0. 0: Output low if p30_oe is 1 1: Output high if p30_oe is 1.	

### 3.9 Single Inductor Multiple Output Power Supply (SIMO)

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

### 3.9.1 Power Supply Monitor

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

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

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

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

*Table 3-18: SIMO Power Supply Device Pin Connectivity*

SIMO Supply Output Pin	Connection	Device Power Supply Input Pin	Supply Monitor Reset Action
$V_{REGO\_A}$	→	$V_{DDA}$	POR
$V_{REGO\_B}$	→	$V_{COREB}$	POR
$V_{REGO\_C}$	→	$V_{COREA}$	POR
-	-	$V_{REGI}$	POR
-	-	$V_{DDIO}$ Power On	GPIO pad held in reset until the voltage rises above threshold
-	-	$V_{DDIOH}$ Power On	GPIO pad held in reset until the voltage rises above threshold
-	-	$V_{DDIO}$	GPIO pad logic enters POR
-	-	$V_{DDIOH}$	GPIO pad logic enters POR

### 3.9.2 Single Inductor Multiple Output Registers (SIMO)

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

*Table 3-19: SIMO Controller Register Summary*

Offset	Register	Access	Name
[0x0004]	<a href="#"><i>SIMO_VREGO_A</i></a>	R/W	Buck Voltage Regulator A Control Register
[0x0008]	<a href="#"><i>SIMO_VREGO_B</i></a>	R/W	Buck Voltage Regulator B Control Register
[0x000C]	<a href="#"><i>SIMO_VREGO_C</i></a>	R/W	Buck Voltage Regulator C Control Register
[0x0014]	<a href="#"><i>SIMO_IPKA</i></a>	RO	Reserved. Do not modify this register.
[0x0018]	<a href="#"><i>SIMO_IPKB</i></a>	RO	Reserved. Do not modify this register.
[0x001C]	<a href="#"><i>SIMO_MAXTON</i></a>	RO	Reserved. Do not modify this register.
[0x0020]	<a href="#"><i>SIMO_ILOAD_A</i></a>	RO	Reserved. Do not modify this register.
[0x0024]	<a href="#"><i>SIMO_ILOAD_B</i></a>	RO	Reserved. Do not modify this register.
[0x0028]	<a href="#"><i>SIMO_ILOAD_C</i></a>	RO	Reserved. Do not modify this register.
[0x0030]	<a href="#"><i>SIMO_BUCK_ALERT_THR_A</i></a>	RO	Reserved. Do not modify this register.
[0x0034]	<a href="#"><i>SIMO_BUCK_ALERT_THR_B</i></a>	RO	Reserved. Do not modify this register.
[0x0038]	<a href="#"><i>SIMO_BUCK_ALERT_THR_C</i></a>	RO	Reserved. Do not modify this register.
[0x0040]	<a href="#"><i>SIMO_BUCK_OUT_READY</i></a>	RO	Buck Regulator Output Ready Register
[0x0044]	<a href="#"><i>SIMO_ZERO_CROSS_CAL_A</i></a>	RO	Reserved. Do not modify this register.

Offset	Register	Access	Name
[0x0048]	<a href="#">SIMO_ZERO_CROSS_CAL_B</a>	RO	<i>Reserved. Do not modify this register.</i>
[0x004C]	<a href="#">SIMO_ZERO_CROSS_CAL_C</a>	RO	<i>Reserved. Do not modify this register.</i>

### 3.9.3 Single Inductor Multiple Output (SIMO) Registers Details

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

SIMO Buck Voltage Regulator A Control			SIMO_VREGO_A		[0x0004]
Bits	Field	Access	Reset	Description	
31:8	-	RO	-	<b>Reserved</b>	
7	rangea	R/W	1	<b>Regulator Output A Range</b> Selects the Regulator output range for V <sub>REGO_A</sub> . 0: 0.5V to 1.77 1: 0.6V to 1.87V	
6:0	vseta	R/W	0x78h	<b>Regulator Output A Voltage</b> Each bit increment in this field represents 10mV allowing output voltage settings from the minimum to the maximum of the <a href="#">SIMO_VREGO_A.rangea</a> selected. $SIMO\_VREGO\_A.rangea = 1: Output\ Voltage = 0.6V + (10mV \times vseta)$ $SIMO\_VREGO\_A.rangea = 0: Output\ Voltage = 0.5V + (10mV \times vseta)$ Default: 0x78 = <a href="#">SIMO_VREGO_A.rangea</a> = 0, Output Voltage = 1.7V; <a href="#">SIMO_VREGO_A.rangea</a> = 1, Output Voltage = 1.8V <i>Warning: When this regulator is connected as shown in <a href="#">Table 3-18: SIMO Power Supply Device Pin Connectivity</a>, the following apply:</i> <ol style="list-style-type: none"> <li>1. The maximum setting for this regulator must be followed for V<sub>DDA</sub> as indicated in the device data sheet.</li> <li>2. Setting the regulator to a voltage below the Power-Fail Reset Voltage for V<sub>DDA</sub> will initiate the Power Monitor Reset Action.</li> </ol>	

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

SIMO Buck Voltage Regulator B Control			SIMO_VREGO_B		[0x0008]
Bits	Field	Access	Reset	Description	
31:8	-	RO	-	<b>Reserved</b>	
7	rangeb	R/W	1	<b>Regulator Output B Range</b> Selects the Regulator output range for V <sub>REGO_B</sub> . 0: 0.5V to 1.77 1: 0.6V to 1.87V	

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

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

SIMO Buck Voltage Regulator C Control			SIMO_VREGO_C		[0x000C]
Bits	Field	Access	Reset	Description	
31:8	-	RO	-	<b>Reserved</b>	
7	rangec	R/W	1	<p><b>Regulator Output Range</b>            Selects the Regulator output range for <math>V_{REGO\_C}</math>.</p> <p>0: 0.5V to 1.77            1: 0.6V to 1.87V</p>	
6:0	vsetc	R/W	0x32h	<p><b>Regulator Output Voltage</b>            Each increment in the register represents 10mV.</p> <p><i>SIMO_VREGO_C.rangec</i> = 1; <i>Output Voltage</i> = <math>0.6V + (10mV \times vsetc)</math>  <i>SIMO_VREGO_C.rangec</i> = 0; <i>Output Voltage</i> = <math>0.5V + (10mV \times vsetc)</math></p> <p>Setting this field to 0x7F results in the maximum output voltage per the <i>SIMO_VREGO_C.rangec</i> selected (1.77V or 1.87V)</p> <p>Default: 0x32 = <i>SIMO_VREGO_C.rangec</i> = 0, <i>Output Voltage</i> = 1.0V;  <i>SIMO_VREGO_C.rangec</i> = 1, <i>Output Voltage</i> = 1.1V</p> <p><b>Warning:</b> When this regulator is connected as shown in Table 3-18: SIMO Power Supply Device Pin Connectivity, the following apply:</p> <ol style="list-style-type: none"> <li>1. The maximum setting for this regulator must be followed for <math>V_{COREA}</math> as indicated in the device data sheet.</li> <li>2. Setting the regulator to a voltage below the Power-Fail Reset Voltage for <math>V_{COREA}</math> will initiate the Power Monitor Reset Action.</li> </ol>	

Table 3-23: 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	-	<b>Reserved</b>	

SIMO High Side FET Peak Current V <sub>REGO_A</sub> V <sub>REGO_B</sub>				SIMO_IPKA	[0x0014]
Bits	Field	Access	Reset	Description	
7:4	ipksetb	RO	8	Reserved	
3:0	ipkseta	RO	8	Reserved	

Table 3-24: SIMO High Side FET Peak Current V<sub>REGO\_C</sub> Register

SIMO High Side FET Peak Current V <sub>REGO_C</sub> V <sub>REGO_D</sub>				SIMO_IPKB	[0x0018]
Bits	Field	Access	Reset	Description	
31:4	-	RO	-	Reserved	
3:0	ipksetc	RO	8	Reserved	

Table 3-25: 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	0x8h	Reserved	

Table 3-26: SIMO Buck Cycle Count V<sub>REGO\_A</sub> Register

SIMO Buck Cycle Count V <sub>REGO_A</sub>				SIMO_ILOAD_A	[0x0020]
Bits	Field	Access	Reset	Description	
31:8	-	RO	0	Reserved	
7:0	iloada	RO	0	Reserved	

Table 3-27: SIMO Buck Cycle Count V<sub>REGO\_B</sub> Register

SIMO Buck Cycle Count V <sub>REGO_B</sub>				SIMO_ILOAD_B	[0x0024]
Bits	Field	Access	Reset	Description	
31:8	-	RO	0	Reserved	
7:0	iloadb	RO	0	Reserved	

Table 3-28: SIMO Buck Cycle Count V<sub>REGO\_C</sub> Register

SIMO Buck Cycle Count V <sub>REGO_C</sub>				SIMO_ILOAD_C	[0x0028]
Bits	Field	Access	Reset	Description	
31:8	-	RO	0	Reserved	
7:0	iloadc	RO	0	Reserved	

Table 3-29: SIMO Buck Cycle Count Alert V<sub>REGO\_A</sub> Register

SIMO Buck Cycle Count Alert V <sub>REGO_A</sub>				SIMO_BUCK_ALERT_THR_A	[0x0030]
Bits	Field	Access	Reset	Description	
31:8	-	RO	0	Reserved	
7:0	buckthra	RO	0	Reserved	

Table 3-30: SIMO Buck Cycle Count Alert V<sub>REGO\_B</sub> 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 3-31: SIMO Buck Cycle Count Alert V<sub>REGO\_C</sub> 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 3-32: SIMO Buck Regulator Output Ready Register

SIMO Buck Regulator Output Ready				SIMO_BUCK_OUT_READY	[0x0040]
Bits	Field	Access	Reset	Description	
31:4	-	RO	0	Reserved	
3	buckoutrdya	RO	0	<b>V<sub>REGO_A</sub> Output Ready</b> When <a href="#">SIMO_VREGO_A.vseta</a> changes, this bit will be set when the output voltage has reached its regulated value. It will not be cleared if the output voltage drops below its set value. 0: Not ready 1: Ready	
2	buckoutrdyb	RO	0	<b>V<sub>REGO_B</sub> Output Ready</b> When <a href="#">SIMO_VREGO_B.vsetb</a> changes, this bit will be set when the output voltage has reached its regulated value. It will not be cleared if the output voltage drops below its set value. 0: Not ready 1: Ready	
1	buckoutrdyc	R/W	0	<b>V<sub>REGO_C</sub> Output Ready</b> When <a href="#">SIMO_VREGO_C.vsetc</a> changes, this bit will be set when the output voltage has reached its regulated value. It will not be cleared if the output voltage drops below its set value. 0: Not ready 1: Ready	
0	-	RO	0	Reserved	

 Table 3-33: SIMO Zero Cross Calibration V<sub>REGO\_A</sub> Register

SIMO Zero Cross Calibration V <sub>REGO_A</sub>				SIMO_ZERO_CROSS_CAL_A	[0x0044]
Bits	Field	Access	Reset	Description	
31:5	-	RO	0	Reserved	
4:0	zxcla	RO	0	Reserved	

Table 3-34: SIMO Zero Cross Calibration V<sub>REGO\_B</sub> Register

SIMO Zero Cross Calibration V <sub>REGO_B</sub>			SIMO_ZERO_CROSS_CAL_B		[0x0048]
Bits	Field	Access	Reset	Description	
31:5	-	RO	0	Reserved	
4:0	zxclb	RO	0	Reserved	

Table 3-35: SIMO Zero Cross Calibration V<sub>REGO\_C</sub> Register

SIMO Zero Cross Calibration V <sub>REGO_C</sub>			SIMO_ZERO_CROSS_CAL_C		[0x004C]
Bits	Field	Access	Reset	Description	
31:5	-	RO	0	Reserved	
4:0	zxclc	RO	0	Reserved	
4:0	zxclid	RO	0	Reserved	

### 3.10 Low-Power General Control Registers (LPGCR)

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

- LPUART0 (UART3)
- LPTMRO (TMR4)
- LPTMR1 (TMR5)
- LPWDTO (WDT1)
- LPCOMP1, LPCOMP2 and LPCOMP3
- GPIO2

See [Table 2-4](#) for the base address of this peripheral/module. See [Table 2-1](#) 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 3-36 Low-Power Control Register Summary*

Offset	Register	Name
[0x0004]	<a href="#">LPGCR_RST</a>	<i>Reset Control Register</i>
[0x0008]	<a href="#">LPGCR_PCLKDIS</a>	<i>Clock Control Register</i>

#### 3.10.1 Low-Power General Control Registers Details

*Table 3-37: Reset Control Register*

Low-Power Reset Control			LPGCR_RST		[0x0004]
Bits	Field	Access	Reset	Description	
31:7	-	RO	0	<b>Reserved</b>	
6	lpcomp	W1O	0	<b>Low Power Comparators Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete. See <a href="#">Device Resets</a> for additional information.	
5	-	RO	0	<b>Reserved</b>	
4	uart3	W1O	0	<b>UART3 (LPUART0) Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete. See <a href="#">Device Resets</a> for additional information.	
3	tmr5	W1O	0	<b>TMR5 (LPTMR1) Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete. See <a href="#">Device Resets</a> for additional information.	
2	tmr4	W1O	0	<b>TMR4 (LPTMR0) Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete. See <a href="#">Device Resets</a> for additional information.	
1	wdt1	W1O	0	<b>WDT1 (LPWDTO) Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete. See <a href="#">Device Resets</a> for additional information.	
0	gpio2	W1O	0	<b>GPIO2 Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete. See <a href="#">Device Resets</a> for additional information.	

**Table 3-38: Clock Disable Register**

Clock Disable			LPGCR_PCLKDIS		[0x008]
Bits	Field	Access	Reset	Description	
31:7	-	RO	0	<b>Reserved</b>	
6	lpcomp	R/W	0	<b>Low Power Comparators Clock Disable</b> Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. <i>Note: This field disables clocks to LPCOMP1, LPCOMP2 and LPCOMP3.</i> 0: Enabled 1: Disabled	
5	-	RO	0	<b>Reserved</b>	
4	uart3	R/W	0	<b>UART3 (LPUART0) Clock Disable</b> 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	<b>TMR5 (LPTMR1) Clock Disable</b> 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	<b>TMR4 (LPTMR0) Clock Disable</b> 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	<b>WDT1 (LPWDT0) Clock Disable</b> Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Enabled 1: Disabled	
0	gpio2	R/W	0	<b>GPIO2 Clock Disable</b> Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Enabled 1: Disabled	

### 3.11 Power Sequencer Registers (PWRSEQ)

See [Table 2-4](#) for the base address of this peripheral/module. See [Table 2-1](#) 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 3-39: Power Sequencer Register Summary**

Offset	Register	Name
[0x0000]	<a href="#">PWRSEQ_LPCN</a>	Low Power Control Register
[0x0004]	<a href="#">PWRSEQ_LPWKSTO</a>	Low Power GPIO0 Wakeup Status Flags
[0x0008]	<a href="#">PWRSEQ_LPWKENO</a>	Low Power GPIO0 Wakeup Enable Register

Offset	Register	Name
[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_GPO	General Purpose Register 0
[0x004C]	PWRSEQ_GP1	General Purpose Register 1

### 3.11.1 Power Sequencer Register Details

Table 3-40: Low Power Control Register

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

Low Power Control				PWRSEQ_LPCN	[0x0000]
Bits	Field	Access	Reset	Description	
3	ramret3	R/W	0	<b>System RAM 3 (<i>sysram3</i>) Data Retention Enable for BACKUP Mode</b> Set this field to 1 to enable Data Retention for System RAM 3. See <a href="#">SRAM Space</a> for System RAM configuration. 0: Disable data retention for <i>sysram3</i> address space in BACKUP. 1: Enable data retention for <i>sysram3</i> address space in BACKUP.	
2	ramret2	R/W	0	<b>System RAM 2 Data Retention Enable for BACKUP Mode</b> Set this field to 1 to enable Data Retention for System RAM 2. See <a href="#">SRAM Space</a> for System RAM configuration. 0: Disable data retention for <i>sysram2</i> address space in BACKUP. 1: Enable data retention for <i>sysram2</i> address space in BACKUP.	
1	ramret1	R/W	0	<b>System RAM 1 (<i>sysram1</i>) Data Retention Enable for BACKUP Mode</b> Set this field to 1 to enable Data Retention for System RAM 1. See <a href="#">SRAM Space</a> for System RAM configuration. 0: Disable data retention for <i>sysram1</i> address space in BACKUP mode. 1: Enable data retention for <i>sysram1</i> address space in BACKUP mode.	
0	ramret0	R/W	0	<b>System RAM 0 (<i>sysram0</i>) Data Retention Enable for BACKUP Mode</b> Set this field to 1 to enable Data Retention for System RAM 1. See <a href="#">SRAM Space</a> for System RAM configuration. 0: Disable data retention for <i>sysram0</i> address space in BACKUP mode. 1: Enable data retention for <i>sysram0</i> address space in BACKUP mode.	

Table 3-41: GPIO0 Low Power Wakeup Status Flags

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

Table 3-42: GPIO0 Low Power Wakeup Enable Registers

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

Table 3-43: GPIO1 Low Power Wakeup Status Flags

GPIO1 Low Power Wakeup Status Flags			PWRSEQ_LPWKST1		[0x000C]
Bits	Field	Access	Reset	Description	
31:0	-	RO	0	<b>Reserved</b> Bits corresponding to unimplemented GPIO are ignored.	
9:0	st	R/W1C	0	<b>GPIO1 Pin Wakeup Status Flag</b> 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 <a href="#">PWRSEQ_LPWKEN1</a> . <i>Note: Clear this register before entering any low power mode.</i>	

Table 3-44: GPIO1 Low Power Wakeup Enable Registers

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

Table 3-45: GPIO2 Low Power Wakeup Status Flags

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

Table 3-46: GPIO2 Low Power Wakeup Enable Registers

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

Table 3-47: GPIO3 Low Power Wakeup Status Flags

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

Table 3-48: GPIO3 Low Power Wakeup Enable Registers

GPIO3 Low Power Wakeup Enable			PWRSEQ_LPWKEN3		[0x0020]
Bits	Field	Access	Reset	Description	
31:2		RO	0	<b>Reserved</b>	
1:0	en	R/W	0	<b>GPIO3 Pin Wakeup Interrupt Enable</b> Setting a GPIO3 pin's bit in this register will cause an interrupt to be generated that wake up the device from any low power mode to ACTIVE mode. A wakeup event sets the corresponding GPIO3's bit in the <a href="#">PWRSEQ_LPWKST3</a> register, enabling determination of which GPIO3 pin triggered the wakeup event. Bits corresponding to unimplemented GPIO are ignored.  <i>Note: To enable the MAX78000 to wake up from a low power mode on a GPIO pin transition, first set the "GPIO Wakeup Enable" register bit <a href="#">GCR_PM gpio_we = 1</a>.</i>	

Table 3-49: Low Power Peripheral Wakeup Status Flags

Low Power Peripheral Wakeup Status Flags			PWRSEQ_LPPWST		[0x0030]
Bits	Field	Access	Reset	Description	
31:18		RO	0	<b>Reserved</b>	

Low Power Peripheral Wakeup Status Flags			PWRSEQ_LPPWST		[0x0030]
Bits	Field	Access	Reset	Description	
17	reset	R/W1C	0	<b>Reset Detected Wakeup Flag</b> This field is set when an external reset caused the wakeup event.	
16	backup	R/W1C	0	<b>BACKUP Mode Wakeup Flag</b> This field is set when the device wakes up from BACKUP mode.	
15:5	-	RO	0	<b>Reserved</b>	
4	comp0	R/W1C	0	<b>Comparator 0 Wakeup Flag</b> This field is set if the wakeup event was the result of a comparator 0 trigger event.	
3:0	-	RO	0	<b>Reserved</b>	

Table 3-50: Low Power Peripheral Wakeup Enable Registers

Low Power Peripheral Wakeup Enable			PWRSEQ_LPPWEN		[0x0034]
Bits	Field	Access	Reset	Description	
31:27		RO	0	<b>Reserved</b>	
26	lpcomp	R/W	0	<b>Low Power Comparator IRQ Wakeup Enable</b> Set this field to 1 to enable wakeup events from the LPCOMPn IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ.	
25	spi1	R/W	0	<b>SPI1 IRQ Wakeup Enable</b> Set this field to 1 to enable wakeup events from the SPI1 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ.	
24	i2s	R/W	0	<b>I2S IRQ Wakeup Enable</b> Set this field to 1 to enable wakeup events from the I2S IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ.	
23	i2c2	R/W	0	<b>I2C2 IRQ Wakeup Enable</b> Set this field to 1 to enable wakeup events from the I2C2 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ.	
22	i2c1	R/W	0	<b>I2C1 IRQ Wakeup Enable</b> Set this field to 1 to enable wakeup events from the I2C1 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ.	
21	i2c0	R/W	0	<b>I2C0 IRQ Wakeup Enable</b> Set this field to 1 to enable wakeup events from the I2C0 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ.	
20	uart3	R/W	0	<b>LPUART0 (UART3) IRQ Wakeup Enable</b> Set this field to 1 to enable wakeup events from LPUART0 (UART3) IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ.	

Low Power Peripheral Wakeup Enable				PWRSEQ_LPPWEN	[0x0034]
Bits	Field	Access	Reset	Description	
19	uart2	R/W	0	<b>UART2 IRQ Wakeup Enable</b> Set this field to 1 to enable wakeup events from the UART2 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ.	
18	uart1	R/W	0	<b>UART1 IRQ Wakeup Enable</b> Set this field to 1 to enable wakeup events from the UART1 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ.	
17	uart0	R/W	0	<b>UART0 IRQ Wakeup Enable</b> Set this field to 1 to enable wakeup events from the UART0 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ.	
16	tmr5	R/W	0	<b>LPTMR1 (TMR5) IRQ Wakeup Enable</b> Set this field to 1 to enable wakeup events from the LPTMR1 (TMR5) IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ.	
15	tmr4	R/W	0	<b>LPTMR0 (TMR4) IRQ Wakeup Enable</b> Set this field to 1 to enable wakeup events from the LPTMR0 (TMR4) IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ.	
14	tmr3	R/W	0	<b>TMR3 IRQ Wakeup Enable</b> Set this field to 1 to enable wakeup events from the TMR3 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ.	
13	tmr2	R/W	0	<b>TMR2 IRQ Wakeup Enable</b> Set this field to 1 to enable wakeup events from the TMR2 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ.	
12	tmr1	R/W	0	<b>TMR1 IRQ Wakeup Enable</b> Set this field to 1 to enable wakeup events from the TMR1 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ.	
11	tmr0	R/W	0	<b>TMRO IRQ Wakeup Enable</b> Set this field to 1 to enable wakeup events from the TMRO IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ.	
10	cpu1	R/W	0	<b>CPU1 (RV32) IRQ Wakeup Enable</b> Set this field to 1 to enable wakeup events from the RV32 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ.	
9	wdt1	R/W	0	<b>WDT1 (LPWDT0) IRQ Wakeup Enable</b> Set this field to 1 to enable wakeup events from the WDT1 (LPWDT0) IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ.	

Low Power Peripheral Wakeup Enable			PWRSEQ_LPPWEN		[0x0034]
Bits	Field	Access	Reset	Description	
8	wdt0	R/W	0	<b>WDT0 IRQ Wakeup Enable</b> Set this field to 1 to enable wakeup events from the WDT0 IRQ. 0: Disable wakeup on IRQ. 1: Enable wakeup on IRQ.	
7:5	-	RO	0	<b>Reserved</b>	
4	comp0	R/W	0	<b>Comparator 0 Wakeup Enable</b> Set this field to 1 to enable wakeup events from Comparator 0. Comparator 0 can wake the device up from <i>SLEEP</i> , <i>LPM</i> , <i>UPM</i> , <i>STANDBY</i> , and <i>BACKUP</i> . 0: Disable wakeup on IRQ 1: Enable wakeup on IRQ	
3:0	-	RO	0	<b>Reserved</b>	

Table 3-51: Low Power General Purpose 0 Register

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

Table 3-52: Low Power General Purpose 1 Register

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

### 3.12 Trim System Initialization Registers (TRIMSIR)

See [Table 2-4](#) for the base address of this peripheral/module. See [Table 2-1](#) 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 Power-On-Reset. System Reset, Soft Reset and Peripheral Reset do not affect the TRIMSIR register values.*

Table 3-53: Trim System Initialization Register Summary

Offset	Register Name	Description
[0x0008]	<a href="#">TRIMSIR_RTC</a>	RTC Trim System Initialization Register
[0x0034]	<a href="#">TRIMSIR_SIMO</a>	System Initialization Register
[0x003C]	<a href="#">TRIMSIR_IPOLO</a>	System initialization Function Status Register
[0x0040]	<a href="#">TRIMSIR_CTRL</a>	Control Trim System Initialization Register
[0x0044]	<a href="#">TRIMSIR_INRO</a>	INRO Trim System Initialization Register

### 3.12.1 TRIM System Initialization Register Details

*Table 3-54: RTC Trim System Initialization Register*

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

*Table 3-55: SIMO Trim System Initialization Register*

SIMO System Initialization			TRIMSIR_SIMO		[0x0034]
Bits	Name	Access	Reset	Description	
31:3	-	RO	0	<b>Reserved</b>	
2:0	clkdiv	R/W	1	<b>SIMO Clock Divide</b> This field selects the SIMO clock divisor. The SIMO uses the INRO as its input clock. 0: $\frac{INRO}{1}$ 1: $\frac{INRO}{16}$ 2: Reserved for Future Use 3: $\frac{INRO}{32}$ 4: Reserved for Future Use 5: $\frac{INRO}{64}$ 6: Reserved for Future Use 7: $\frac{INRO}{128}$	

*Table 3-56: IPO Low Trim System Initialization Register*

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

**Table 3-57: Control Trim System Initialization Register**

Control System Initialization			TRIMSir_CTRL		[0x0040]
Bits	Name	Access	Reset	Description	
31:29	inro_trim	R/W	See Description	<b>INRO Clock Trim</b> This field contains the trim for the INRO when set to 8KHz.	
28:26	-	RO	0	<b>Reserved</b>	
25:24	inro_sel	R/W	2	<b>INRO Clock Select</b> Selects the INRO frequency. 0: 8KHz 1: 16KHz 2: 30KHz (Power-On Reset default) 3: Reserved for Future Use	
23:15	ipo_limithi	R/W	0x1FF	<b>IPO High Trim Limit</b> This field contains the high limit for the IPO.	
14:8	vdda_limithi	R/W	0x78	<b>V<sub>DDA</sub> High Trim Limit</b> High trim limit for V <sub>DDA</sub> .	
7	-	RO	0	<b>Reserved</b>	
6:0	vdda_limitlo	R/W	0x64	<b>V<sub>DDA</sub> Low Trim Limit</b> Low trim limit for V <sub>DDA</sub>	

**Table 3-58: INRO Trim System Initialization Register**

INRO System Initialization			TRIMSir_INRO		[0x0044]
Bits	Name	Access	Reset	Description	
31:8	-	RO	0	<b>Reserved</b>	
7:6	lpclkSEL	R/W	2	<b>INRO Low Power Mode Clock Select</b> Selects the INRO clock frequency for LPM operation. 0: 8KHz 1: 16KHz 2: 30KHz (POR default) 3: Reserved for Future Use	
5:3	trim30k	R/W	0	<b>INRO 30KHz Trim</b> This field contains the trim for INRO when set to 30KHz.	
2:0	trim16k	R/W	0	<b>INRO 16KHz Trim</b> This field contains the trim for INRO when set to 16KHz.	

### 3.13 Global Control Registers (GCR)

See [Table 2-4](#) for the base address of this peripheral/module. See [Table 2-1](#) 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 General Control Registers are only reset on a System Reset or Power-On Reset. A Soft Reset or Peripheral Reset does not affect these registers.*

Table 3-59: Global Control Register Summary

Offset	Register	Description
[0x0000]	<a href="#">GCR_SYSCTRL</a>	System Control Register
[0x0004]	<a href="#">GCR_RST0</a>	Reset Register 0
[0x0008]	<a href="#">GCR_CLKCTRL</a>	Clock Control Register
[0x000C]	<a href="#">GCR_PM</a>	Power Management Register
[0x0018]	<a href="#">GCR_PCLKDIV</a>	Peripheral Clocks Divisor
[0x0024]	<a href="#">GCR_PCLKDIS0</a>	Peripheral Clocks Disable 0
[0x0028]	<a href="#">GCR_MEMCTRL</a>	Memory Clock Control
[0x002C]	<a href="#">GCR_MEMZ</a>	Memory Zeroize Register
[0x0040]	<a href="#">GCR_SYST</a>	System Status Flags
[0x0044]	<a href="#">GCR_RST1</a>	Reset Register 1
[0x0048]	<a href="#">GCR_PCLKDIS1</a>	Peripheral Clocks Disable 1
[0x004C]	<a href="#">GCR_EVENTEN</a>	Event Enable Register
[0x0050]	<a href="#">GCR_REVISION</a>	Revision Register
[0x0054]	<a href="#">GCR_SYSIE</a>	System Status Interrupt Enable
[0x0064]	<a href="#">GCR_ECCERR</a>	Error Correction Coding Error Register
[0x0068]	<a href="#">GCR_ECCCED</a>	Error Correction Coding Correctable Error Detected
[0x006C]	<a href="#">GCR_ECCIE</a>	Error Correction Coding Interrupt Enable Register
[0x0070]	<a href="#">GCR_ECCADDR</a>	Error Correction Coding Error Address Register
[0x0080]	<a href="#">GCR_GPRO</a>	General Purpose Register 0

### 3.13.1 Global Control Register Details (GCR)

Table 3-60: System Control Register

System Control			<a href="#">GCR_SYSCTRL</a>		[0x0000]
Bits	Field	Access	Reset	Description	
31:18	-	RO	0	Reserved	
17:16	ovr	R/W	0b10	<b>Operating Voltage Range</b> Set this field to match the V <sub>COREA</sub> voltage to enable the on-chip RAM to operate at the optimal timing range. 0b00: 0.9V ± 10% 0b01: 1.0V ± 10% 0b10: 1.1V ± 10% 0b11: Reserved for Future Use	
15	chkres	R	0	<b>ROM Checksum Calculation Pass/Fail</b> This is the result after setting bit <a href="#">GCR_SYSCTRL.cchk</a> . This bit is only valid after the ROM checksum is complete and <a href="#">GCR_SYSCTRL.cchk</a> is cleared. 0: Pass 1: Fail	

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

Table 3-61: Reset Register 0

Reset 0				GCR_RST0	[0x0004]
Bits	Field	Access	Reset	Description	
31	sys	R/W	0	<b>System Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete. See <a href="#">System Reset</a> for additional information.	

Reset 0			GCR_RST0		[0x0004]
Bits	Field	Access	Reset	Description	
30	periph	R/W	0	<b>Peripheral Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete. <i>Note: Watchdog Timers, GPIO Ports, the AoD, RAM Retention and the General Control Registers (GCR) are unaffected. See <a href="#">Peripheral Reset</a> for additional information.</i>	
29	soft	R/W	0	<b>Soft Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete. See <a href="#">Soft Reset</a> for additional information.	
28	uart2	R/W	0	<b>UART2 Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete.	
27	-	R/W	0	<b>Reserved</b>	
26	adc	R/W	0	<b>ADC Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete.	
25	cnn	R/W	0	<b>CNN Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete.	
24	trng	R/W	0	<b>TRNG Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete.	
23:18	-	R/W	0	<b>Reserved</b>	
17	rtc	R/W	0	<b>RTC Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete.	
16	i2c0	R/W	0	<b>I2C0 Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete.	
15:14	-	RO	0	<b>Reserved</b>	
13	spi1	R/W	0	<b>SPI1 Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete.	
12	uart1	R/W	0	<b>UART1 Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete.	
11	uart0	R/W	0	<b>UART0 Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete.	
10:9	-	R/W	0	<b>Reserved</b>	
8	tmr3	R/W	0	<b>TMR3 Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete.	
7	tmr2	R/W	0	<b>TMR2 Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete.	
6	tmr1	R/W	0	<b>TMR1 Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete.	
5	tmr0	R/W	0	<b>TMR0 Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete.	
4	-	RO	-	<b>Reserved</b>	

Reset 0				GCR_RST0	[0x0004]
Bits	Field	Access	Reset	Description	
3	gpio1	R/W	0	<b>GPIO1 Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete.	
2	gpio0	R/W	0	<b>GPIO0 Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete.	
1	wdt0	R/W	0	<b>Watchdog Timer 0 Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete.	
0	dma	R/W	0	<b>DMA Access Block Reset</b> Write 1 to reset. This field is cleared by hardware when the reset is complete.	

Table 3-62: Clock Control Register

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

Clock Control			GCR_CLKCTRL		[0x0008]
Bits	Field	Access	Reset	Description	
17	ertco_en	R/W	0	<b>32.768kHz ERTCO Enable</b> 0: Disabled if the <a href="#">RTC_CTRL.en</a> field is also set to 0. 1: Enabled and ready when <a href="#">GCR_CLKCTRL.ertco_rdy</a> = 1, regardless of the state of the <a href="#">RTC_CTRL.en</a> field.	
16:14	-	RO	0	<b>Reserved</b>	
13	sysclk_rdy	R	0	<b>SYS_OSC Select Ready</b> When SYS_OSC is changed by modifying <a href="#">GCR_CLKCTRL.sysclk_sel</a> , there is a delay until the switchover is complete. This bit is cleared until the switchover is complete. 0: Switch to new clock source not yet complete. 1: SYS_OSC is clock source selected in <a href="#">GCR_CLKCTRL.sysclk_sel</a> .	
12	-	RO	0	<b>Reserved</b>	
11:9	sysclk_sel	R/W	0	<b>System Clock Source Select</b> Selects the system oscillator (SYS_OSC) used as the system clock (SYS_CLK) source. Modifying this field immediately clears <a href="#">GCR_CLKCTRL.sysclk_rdy</a> . 0: ISO (POR and System Reset default) 1: Reserved 2: Reserved 3: INRO 4: IPO 5: IBRO 6: ERTCO 7: External Clock, EXT_CLK, P0.3, AF1	
8:6	sysclk_div	R/W	0	<b>System Clock Prescaler</b> 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	<b>Reserved</b>	

Table 3-63: Power Management Register

Power Management			GCR_PM		0x000C
Bits	Field	Access	Reset	Description	
31:18	-	RO	0	<b>Reserved</b>	
17	ibro_pd	R/W	1	<b>IBRO Power Down LPM</b> 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	<b>IPO Power Down LPM</b> 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	

Power Management			GCR_PM		0x000C
Bits	Field	Access	Reset	Description	
15	iso_pd	R/W	1	<b>ISO Power Down LPM</b> Set this field to 1 to power down the ISO when entering <i>LPM</i> . 0: ISO is powered on during <i>LPM</i> 1: ISO is powered off during <i>LPM</i>	
14:10	-	DNM	0b11100	<b>Reserved</b>	
9	aincomp_we	R/W	0	<b>Analog Input Comparator Wakeup Enable</b> This bit enables the Analog Input Comparator IRQ to wake the device from <i>SLEEP</i> , <i>LPM</i> , or <i>BACKUP</i> modes.	
8	-	RO	0	<b>Reserved</b>	
7	wut_we	R/W	0	<b>Wakeup Timer Enable</b> Set this field to 1 to enable the Wakeup Timer as a wakeup source. The Wakeup Timer will wake the device from <i>SLEEP</i> , <i>LPM</i> , or <i>BACKUP</i> modes. 0: Wakeup source disabled 1: Wakeup source enabled.	
6	-	RO	0	<b>Reserved</b>	
5	rtc_we	R/W	0	<b>RTC Alarm Wakeup Enable</b> Set this field to 1 to enable an RTC alarm to wake the device. The RTC alarm will wake the device from <i>SLEEP</i> , <i>LPM</i> , or <i>BACKUP</i> modes. 0: Wakeup source disabled 1: Wakeup source enabled	
4	gpio_we	R/W	0	<b>GPIO Wakeup Enable</b> Set this field to 1 to enable all GPIO pins as potential wakeup sources. Any GPIO configured for wakeup will wake the device from <i>SLEEP</i> , <i>LPM</i> , or <i>BACKUP</i> modes. 0: Wakeup source disabled 1: Wakeup source enabled	
3:0	mode	R/W	0	<b>Operating Mode</b> This field controls the operating mode of the device. 0: <i>ACTIVE</i> 1: <i>SLEEP</i> 2: <i>STANDBY</i> 3: Reserved 4: <i>BACKUP</i> 5-7: Reserved 8: <i>LPM</i> ( <i>CM4 deep sleep</i> ) 9: <i>UPM</i> 10: <i>PDM</i> 11-15: Reserved	

Table 3-64: Peripheral Clock Divisor Register

Peripheral Clocks Divisor			GCR_PCLKDIV		[0x0018]
Bits	Field	Access	Reset	Description	
31:18	-	RO	-	<b>Reserved</b>	
17	cnnclksel	R/W	0	<b>CNN Peripheral Clock Select</b> Set this field to select the clock source for the CNN peripheral clock, $f_{CNN\_clock}$ . 0: PCLK 1: ISO	

Peripheral Clocks Divisor				GCR_PCLKDIV	[0x0018]
Bits	Field	Access	Reset	Description	
16:14	cnnclkdiv	R/W		<b>CNN Peripheral Clock Frequency Divider</b> This field is used as a divider of the CNN Peripheral Clock. The CNN Peripheral Clock, $f_{CNN\_Clock}$ , is selected using the field <a href="#">GCR_PCLKDIV.cnnclksel</a> . 0: $\frac{CNN\_Clock}{2}$ 1: $\frac{CNN\_Clock}{4}$ 2: $\frac{CNN\_Clock}{8}$ 3: $\frac{CNN\_Clock}{16}$ 4-7: $\frac{CNN\_Clock}{1}$	
13:10	adcfreq	R/W	0	<b>ADC Peripheral Clock Frequency Select</b> Configures the frequency of the ADC peripheral clock from the PCLK. 0: Reserved 1: Reserved 2 – 15: $f_{adc\_clk} = f_{PCLK}/adcfreq$	
9:0	-	RO	-	<b>Reserved</b>	

Table 3-65: Peripheral Clock Disable Register 0

Peripheral Clocks Disable 0				GCR_PCLKDIS0	[0x0024]
Bits	Field	Access	Reset	Description	
31:30	-	R/W	1	<b>Reserved</b>	
29	pt	R/W	1	<b>Pulse Train Clock Disable</b> 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	<b>I2C1 Clock Disable</b> 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	<b>Reserved</b>	
25	cnn	R/W	1	<b>CNN Clock Disable</b> 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	<b>Reserved</b>	
23	adc	R/W	1	<b>ADC Clock Disable</b> 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	
22:19	-	RO	1	<b>Reserved</b>	
18	tmr3	R/W	1	<b>TMR3 Clock Disable</b> Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Clock enabled 1: Clock disabled	
17	tmr2	R/W	1	<b>TMR2 Clock Disable</b> 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	<b>TMR1 Clock Disable</b> 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	<b>TMRO Clock Disable</b> 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	<b>Reserved</b>	
13	i2c0	R/W	1	<b>I2C0 Clock Disable</b> 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	<b>Reserved</b>	
10	uart1	R/W	1	<b>UART1 Clock Disable</b> 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	<b>UART0 Clock Disable</b> 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	<b>Reserved</b>	
6	spi1	R/W	1	<b>SPI1 Clock Disable</b> 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	
5	dma	R/W	1	<b>DMA Clock Disable</b> Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Clock enabled 1: Clock disabled	
4:2	-	RO	0b11	<b>Reserved</b>	
1	gpio1	R/W	1	<b>GPIO1 Port and Pad Logic Clock Disable</b> 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	<b>GPIO0 Port and Pad Logic Clock Disable</b> Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained. 0: Clock enabled 1: Clock disabled	

Table 3-66: Memory Clock Control Register

Memory Clock Control			GCR_MEMCTRL		[0x0028]
Bits	Field	Access	Reset	Description	
31:17	-	RO	0	<b>Reserved</b>	
16	sysram0ecc	R/W	0	<b>sysram0 ECC Enable</b> Set this field to 1 to enable ECC for sysram0. 0: sysram0 Active, ECC disabled. 1: sysram0 Active, ECC enabled.	
15:3	-	RO	0	<b>Reserved</b>	
2:0	fws	R/W	5	<b>Program Flash Wait States</b> Number of wait-state cycles per Flash code read access. 0: Invalid 1 – 7: Number of Flash code access wait states <i>Note: For the IPO and ISO clocks the minimum wait state is 2.</i> <i>Note: For all other clock sources the minimum wait state is 1.</i>	

Table 3-67: Memory Zeroize Control Register

Memory Zeroize			GCR_MEMZ		[0x002C]
Bits	Field	Access	Reset	Description	
31:7	-	RO	-	<b>Reserved</b>	
6	icc1	R/W1O	0	<b>ICC1 Zeroization</b> 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	
5	icc0	R/W1O	0	<b>ICCO Zeroization</b> Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Operation complete. 1: Operation in progress.	
4	sysram0ecc	R/W1O	0	<b>sysram0 ECC Zeroization</b> Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Operation complete. 1: Operation in progress.	
3	ram3	R/W1O	0	<b>sysram3 Zeroization</b> Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Operation complete. 1: Operation in progress.	
2	ram2	R/W1O	0	<b>sysram2 Zeroization</b> Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Operation complete. 1: Operation in progress.	
1	ram1	R/W1O	0	<b>sysram1 Zeroization</b> Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Operation complete. 1: Operation in progress.	
0	ram0	R/W1O	0	<b>sysram0 Zeroization</b> Write 1 to initiate the operation. This field is automatically cleared by hardware on completion. 0: Operation complete. 1: Operation in progress.	

Table 3-68: System Status Flag Register

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

*Table 3-69: Reset Register 1*

Reset 1			GCR_RST1		[0x0044]
Bits	Field	Access	Reset	Description	
31	cpu1	RO	0	<b>CPU1 (RV32) Reset</b> Write 1 to initiate the reset operation.  0: Operation complete 1: Operation in progress	
30:26	-	RO	0	<b>Reserved</b>	
25	simo	R/W	0	<b>Single Inductor Multiple Output Block Reset</b> Write 1 to initiate the reset operation.  0: Operation complete 1: Operation in progress	
24	dvs	R/W	0	<b>Dynamic Voltage Scaling Controller Reset</b> Write 1 to initiate the operation.  0: Operation complete. 1: Operation in progress.	
23:21	-	RO	0	<b>Reserved</b>	
20	i2c2	R/W	0	<b>I<sup>2</sup>C2 Reset</b> Write 1 to initiate the operation.  0: Operation complete. 1: Operation in progress.	
19	i2s	R/W	0	<b>Audio Interface Reset</b> Write 1 to initiate the operation.  0: Operation complete. 1: Operation in progress.	
18:17	-	R/W	0	<b>Reserved</b>	
16	smphr	R/W	0	<b>Semaphore Block Reset</b> Write 1 to initiate the operation.  0: Operation complete. 1: Operation in progress.	
15:12	-	R/W	-	<b>Reserved</b>	
11	spi0	R/W	0	<b>SPI0 Reset</b> Write 1 to initiate the operation.  0: Operation complete. 1: Operation in progress.	
10	aes	R/W	0	<b>AES Block Reset</b> Write 1 to initiate the operation.  0: Operation complete. 1: Operation in progress.	
9	crc	R/W	0	<b>CRC Reset</b> Write 1 to initiate the operation.  0: Operation complete. 1: Operation in progress.	
8	-	R/W	0	<b>Reserved</b>	

Reset 1			GCR_RST1		[0x0044]
Bits	Field	Access	Reset	Description	
7	owm	R/W	0	<b>1-Wire Reset</b> Write 1 to initiate the operation. 0: Operation complete. 1: Operation in progress.	
6:2	-	RO	0	<b>Reserved</b>	
1	pt	R/W	0	<b>Pulse Train Reset</b> Write 1 to initiate the operation. 0: Operation complete. 1: Operation in progress.	
0	i2c1	R/W	0	<b>I<sup>2</sup>C1 Reset</b> Write 1 to initiate the operation. 0: Operation complete. 1: Operation in progress.	

Table 3-70: Peripheral Clock Disable Register 1

Peripheral Clock Disable 1			GCR_PCLKDIS1		[0x0048]
Bits	Field	Access	Reset	Description	
31	cpu1	R/W	1	<b>CPU1 (RV32 Clock Disable</b> Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked. 0: Enabled 1: Disabled	
30:28	-	R/W	1	<b>Reserved</b>	
27	wdt0	R/W	1	<b>Watchdog Timer 0 Disable</b> 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	<b>Reserved</b>	
24	i2c2	R/W	1	<b>I<sup>2</sup>C2 Clock Disable</b> 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	<b>I<sup>2</sup>S Audio Interface Clock Disable</b> Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked. 0: Enabled 1: Disabled	
22:17	-	R/W	1	<b>Reserved</b>	

Peripheral Clock Disable 1				GCR_PCLKDIS1	[0x0048]
Bits	Field	Access	Reset	Description	
16	spi0	R/W	1	<b>SPI0 Clock Disable</b> 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	<b>AES Block Clock Disable</b> 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	<b>CRC Clock Disable</b> 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	<b>1-Wire Clock Disable</b> Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked. 0: Enabled. 1: Disabled.	
12:10	-	R/W1	1	<b>Reserved</b>	
9	smphr	R/W	1	<b>Semaphore Block Clock Disable</b> Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked. 0: Enabled. 1: Disabled.	
8:3	-	R/W1	1	<b>Reserved</b>	
2	trng	R/W	1	<b>TRNG Clock Disable</b> 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	<b>UART2 Clock Disable</b> Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked. 0: Enabled. 1: Disabled.	
0	-	R/W1	1	<b>Reserved</b>	

Table 3-71: Event Enable Register

Event Enable				GCR_EVENTEN	[0x004C]
Bits	Field	Access	Reset	Description	
31:3	-	RO	0	<b>Reserved</b>	

Event Enable				GCR_EVENTEN	[0x004C]
Bits	Field	Access	Reset	Description	
2	tx	R/W	0	<b>CPU0 (CM4) TXEV Event Enable</b> When this bit is set, the TXEV event will wake the CM4 from a low power mode entered with a WFE instruction. 0: Disabled. 1: Enabled.	
1	-	RO	0	<b>Reserved</b>	
0	dma	R/W	0	<b>CPU0 (CM4) DMA CTZ Wake-Up Enable</b> Enables a DMA CTZ event to generate an RXEV interrupt to wake the CM4 from a low power mode entered with a WFE instruction. 0: Disabled. 1: Enabled.	

Table 3-72: Revision Register

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

Table 3-73: System Status Interrupt Enable Register

System Status Interrupt Enable				GCR_SYSIE	[0x0054]
Bits	Field	Access	Reset	Description	
31:1	-	RO	-	<b>Reserved</b>	
0	iceunlock	R/W	0	<b>Arm ICE Unlocked Interrupt Enable</b> Set this field to generate an interrupt if the <a href="#">GCR_SYSST.iceclock</a> is set. 0: Interrupt disabled 1: Interrupt enabled	

Table 3-74: Error Correction Coding Error Register

Error Correction Coding Error				GCR_ECCERR	[0x0064]
Bits	Field	Access	Reset	Description	
31:1	-	RO	0	<b>Reserved</b>	
0	ram0	R/W1C	0	<b>sysram0 ECC Error</b> This flag is set if an ECC error occurs in sysram0. Write to 1 to clear the flag. 0: No error 1: Error	

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

Error Correction Coding Correctable Error Detected				GCR_ECCCED	[0x0068]
Bits	Field	Access	Reset	Description	
31:1	-	RO	0	<b>Reserved</b>	

Error Correction Coding Correctable Error Detected					GCR_ECCCED	[0x0068]
Bits	Field	Access	Reset	Description		
0	ram0	R/W1C	0	<b>sysram0 Correctable ECC Error Detected</b> 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 <a href="#">GCR_ECCERR.ram0</a> is set to 1. 1: Correctable error detected.		

Table 3-76: Error Correction Coding Interrupt Enable Register

Error Correction Coding Interrupt Enable					GCR_ECCIE	[0x006C]
Bits	Field	Access	Reset	Description		
31:1	-	RO	0	<b>Reserved</b>		
0	ram0	R/W	0	<b>sysram0 ECC Error Interrupt Enable</b> Set this field to 1 to generate an interrupt if an ECC error condition occurs for sysram0. 0: Interrupt disabled 1: Interrupt enabled		

Table 3-77: Error Correction Coding Error Address Register

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

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

Table 3-78: General Purpose 0 Register

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

### 3.14 Error Correction Coding (ECC)

See [Table 2-4](#) for the base address of this peripheral/module. See [Table 2-1](#) 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 3-79: Error Correction Coding Enable Register Summary

Offset	Register Name	Description
[0x0000]	<a href="#">ECC_EN</a>	Error Correction Coding Enable

#### 3.14.1 Error Correction Coding Enable Register Details

Table 3-80: Error Correction Coding Enable Register

Error Correction Coding Enable			ECC_EN		[0x0000]
Bits	Name	Access	Reset	Description	
31:9	-	RO	0	<b>Reserved</b>	
8	ram0	R/W	0	<b>System RAM ECC Enable</b> Set this field to 1 to enable ECC for <i>sysram0</i> . 0: Disabled 1: Enabled	
7:0	-	RO	0	<b>Reserved</b>	

### 3.15 System Initialization Registers (SIR)

See *Table 2-4* for the base address of this peripheral/module. See *Table 2-1* 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 3-81: System Initialization Register Summary*

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

#### 3.15.1 System Initialization Register Details

*Table 3-82: System Initialization Status Register*

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

*Table 3-83: System Initialization Address Error Register*

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

*Table 3-84: System Initialization Function Status Register*

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

*Table 3-85: System Initialization Security Function Status Register*

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

### 3.16 Function Control Registers (FCR)

See *Table 2-4* for the base address of this peripheral/module. See *Table 2-1* 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 3-86: Function Control Register Summary*

Offset	Register	Description
[0x0000]	<a href="#">FCR_FCTRL0</a>	<i>Function Control 0 Register (I<sup>2</sup>C Glitch Filter Control)</i>
[0x0004]	<a href="#">FCR_AUTOCAL0</a>	<i>IPO Automatic Calibration 0 Register</i>
[0x0008]	<a href="#">FCR_AUTOCAL1</a>	<i>IPO Automatic Calibration 1 Register</i>
[0x000C]	<a href="#">FCR_AUTOCAL2</a>	<i>IPO Automatic Calibration 2 Register</i>

Offset	Register	Description
[0x0010]	<a href="#">FCR_UrvBootAddr</a>	RV32 Boot Address Register
[0x0014]	<a href="#">FCR_UrvCtrl</a>	RV32 Control Register

### 3.16.1 Function Control Register Details

Table 3-87: Function Control 0 Register

Function Control 0			FCR_FCTRL0		[0x0000]
Bits	Field	Access	Reset	Description	
31:26	-	RO	0	Reserved	
25	i2c2_scl_filter_en	R/W	0	<b>I<sup>2</sup>C2 SCL Glitch Filter Enable</b> 0: Disabled 1: Enabled	
24	i2c2_sda_filter_en	R/W	0	<b>I<sup>2</sup>C2 SDA Glitch Filter Enable</b> 0: Disabled 1: Enabled	
23	i2c1_scl_filter_en	R/W	0	<b>I<sup>2</sup>C1 SCL Glitch Filter Enable</b> 0: Disabled 1: Enabled	
22	i2c1_sda_filter_en	R/W	0	<b>I<sup>2</sup>C1 SDA Glitch Filter Enable</b> 0: Disabled 1: Enabled	
21	i2c0_scl_filter_en	R/W	0	<b>I<sup>2</sup>C0 SCL Glitch Filter Enable</b> 0: Disabled 1: Enabled	
20	i2c0_sda_filter_en	R/W	0	<b>I<sup>2</sup>C0 SDA Glitch Filter Enable</b> 0: Disabled 1: Enabled	
19:0	-	RO	0	Reserved	

Table 3-88: IPO Automatic Calibration 0 Register

IPO Automatic Calibration 0			FCR_AUTOCAL0		[0x0004]
Bits	Field	Access	Reset	Description	
31:23	trim	RO	0	<b>IPO Trim Value</b> Initial factory trim value for the IPO.	
22:20	-	RO		Reserved	
19:8	gain	R/W	0	<b>IPO Trim Adaptation Gain</b>	
7:5	-	RO	0	Reserved	
4	atomic	R/W1	0	<b>IPO Trim Atomic Start</b> Set this bit to start an atomic automatic calibration of the IPO. The calibration will run for <a href="#">FCR_runtime</a> milliseconds. This bit is automatically cleared by hardware when the calibration is complete.	
3	invert	R/W	0	<b>IPO Trim Step Invert</b> 0: IPO trim step is not inverted 1: IPO trim step is inverted	

IPO Automatic Calibration 0			FCR_AUTOCAL0		[0x0004]
Bits	Field	Access	Reset	Description	
2	load	R/*	0	<b>IPO Initial Trim Load</b> Set this bit to load the initial trim value for the IPO from <a href="#">FCR_.initial</a> . This bit will be cleared by hardware once the load is complete.	
1	en	R/W	0	<b>IPO Automatic Calibration Continuous Mode Enable</b> 0: Disabled 1: Enabled	
0	sel	R/W	0	<b>IPO Trim Select</b> 0: Use default trim 1: Use automatic calibration trim values	

Table 3-89: IPO Automatic Calibration 1 Register

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

Table 3-90: IPO Automatic Calibration 2 Register

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

Table 3-91: RV32 Boot Address Register

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

Table 3-92: RV32 Control Register

RV32 Boot Address			FCR_URVCTRL		[0x0014]
Bits	Field	Access	Reset	Description	
31:2	-	RO	0	<b>Reserved</b>	

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

### 3.17 General Control Function Registers (GCFR)

See [Table 2-4](#) for the base address of this peripheral/module. See [Table 2-1](#) 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 3-93: General Control Function Register Summary*

Offset	Register	Description
[0x0000]	<a href="#">GCFR_REG0</a>	General Control Function Register 0
[0x0004]	<a href="#">GCFR_REG1</a>	General Control Function Register 1
[0x0008]	<a href="#">GCFR_REG2</a>	General Control Function Register 2
[0x000C]	<a href="#">GCFR_REG3</a>	General Control Function Register 3

#### 3.17.1 General Control Function Register Details

*Table 3-94: General Control Function Register 0*

General Control Function 0			GCFR_REG0		[0x0000]
Bits	Field	Access	Reset	Description	
31:4	-	RO	0	<b>Reserved</b>	
3	cnnx16_3_pwr_en	R/W	0	<b>CNNx16_3 Power Domain Enable</b> 0: Disabled 1: Enabled	
2	cnnx16_2_pwr_en	R/W	0	<b>CNNx16_2 Power Domain Enable</b> 0: Disabled 1: Enabled	
1	cnnx16_1_pwr_en	R/W	0	<b>CNNx16_1 Power Domain Enable</b> 0: Disabled 1: Enabled	

General Control Function 0			GCFR_REG0		[0x0000]
Bits	Field	Access	Reset	Description	
0	cnnx16_0_pwr_en	R/W	0	<b>CNNx16_0 Power Domain Enable</b> 0: Disabled 1: Enabled	

Table 3-95: General Control Function Register 1

General Control Function Register 1			GCFR_REG1		[0x0004]
Bits	Field	Access	Reset	Description	
31:4	-	RO	0	Reserved	
3	cnnx16_3_ram_en	R/W	0	<b>CNNx16_3 RAM Power Enable</b> 0: Disabled 1: Enabled	
2	cnnx16_2_ram_en	R/W	0	<b>CNNx16_2 RAM Power Enable</b> 0: Disabled 1: Enabled	
1	cnnx16_1_ram_en	R/W	0	<b>CNNx16_1 RAM Power Enable</b> 0: Disabled 1: Enabled	
0	cnnx16_0_ram_en	R/W	0	<b>CNNx16_0 RAM Power Enable</b> 0: Disabled 1: Enabled	

Table 3-96: General Control Function Register 2

General Control Function Register 2			GCFR_REG2		[0x0008]
Bits	Field	Access	Reset	Description	
31:4	-	RO	0	Reserved	
3	cnnx16_3_iso	R/W	0	<b>CNNx16_3 Power Domain Isolation</b> 0: Disabled 1: Enabled	
2	cnnx16_2_iso	R/W	0	<b>CNNx16_2 Power Domain Isolation</b> 0: Disabled 1: Enabled	
1	cnnx16_1_iso	R/W	0	<b>CNNx16_1 Power Domain Isolation</b> 0: Disabled 1: Enabled	
0	cnnx16_0_iso	R/W	0	<b>CNNx16_0 Power Domain Isolation</b> 0: Disabled 1: Enabled	

Table 3-97: General Control Function Register 3

General Control Function Register 3			GCFR_REG3		[0x000C]
Bits	Field	Access	Reset	Description	
31:4	-	RO	0	Reserved	

General Control Function Register 3			GCFR_REG3		[0x000C]
Bits	Field	Access	Reset	Description	
3	cnnx16_3_RST	R/W	0	<b>CNNx16+3 Power Domain Reset</b> Write this field to 1 to initiate a power domain reset for the CNNx16_3. 0: Reset not in process 1: Reset	
2	cnnx16_2_RST	R/W	0	<b>CNNx16_2 Power Domain Reset</b> Write this field to 1 to initiate a power domain reset for the CNNx16_2. 0: Normal operation 1: Reset	
1	cnnx16_1_RST	R/W	0	<b>CNNx16_1 Power Domain Reset</b> Write this field to 1 to initiate a power domain reset for the CNNx16_1. 0: Normal operation 1: Reset	
0	cnnx16_0_RST	R/W	0	<b>CNNx16_0 Power Domain Reset</b> Write this field to 1 to initiate a power domain reset for the CNNx16_0. 0: Normal operation 1: Reset	

Preliminary Draft 05/21/2021

## 4. Interrupts and Exceptions

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

### 4.1 CM4 Interrupt and Exception Features

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

### 4.2 CM4 Interrupt Vector Table

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

*Table 4-1: MAX78000 CM4 Interrupt Vector Table*

Exception (Interrupt) Number	Offset	Name	Description
1	[0x0004]	Reset_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]	WDTO_IRQn	Windowed Watchdog Timer 0 Interrupt
18	[0x0048]	-	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 <sup>2</sup> 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	[0x90]	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 <sup>2</sup> 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 <sup>2</sup> C Port 2 Interrupt
79	[0x013C]	RISCV_IRQn	CPU1 (RV32) Interrupt
80:82	[0x0140]:[0x0148]	-	Reserved
83	[0x014C]	OWM_IRQn	1-Wire Master Interrupt
84:97	[0x0150]:[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
108:112	[0x01B0]:[0x01C0]	-	Reserved
113	[0x01C4]	AES_IRQn	AES Interrupt
114	[0x01C8]	-	Reserved

Exception (Interrupt) Number	Offset	Name	Description
115	[0x01CC]	I2S_IRQn	I2S Interrupt
116	[0x01D0]	CNN_FIFO_IRQn	CNN Transmit FIFO Interrupt
117	[0x01D4]	CNN_IRQn	CNN Interrupt
118	[0x01D8]	-	Reserved
119	[0x01DC]	LPCOMP_IRQn	Low Power Comparator Interrupt

## 4.3 RV32 Interrupt Vector Table

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

*Table 4-2: MAX78000 RV32 Interrupt Vector Table*

Exception (Interrupt) Number	Name	Description
4	PF_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 <sup>2</sup> C Port 0 Interrupt
15	UART0_IRQn	UART Port 0 Interrupt
16	-	Reserved
17	I2C1_IRQn	I <sup>2</sup> C Port 1 Interrupt
18	UART1_IRQn	UART Port 1 Interrupt
19	UART2_IRQn	UART Port 2 Interrupt
20	I2C2_IRQn	I <sup>2</sup> 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
31	DMA3_IRQn	DMA3 Interrupt
32:45	-	Reserved
46	AES_IRQn	AES Interrupt
47	TRNG_IRQn	TRNG Interrupt

Exception (Interrupt) Number	Name	Description
48	WDT1_IRQn	Watchdog Timer 1 (LPWDT0) Interrupt
49	DVS_IRQn	Digital Voltage Scaling Interrupt
50	SIMO_IRQn	Single Input Multiple Output Interrupt
51	-	Reserved
52	PT_IRQn	Pulse Train Interrupt
53	ADC_IRQn	ADC Interrupt
54	OWM_IRQn	1-Wire Master Interrupt
55	I2S_IRQn	I2S Interrupt
56	CNN_FIFO_IRQn	CNN TX FIFO Interrupt
57	CNN_IRQn	CNN Interrupt
58	-	Reserved
59	PCIF_IRQn	Parallel Camera Interface Interrupt

Preliminary Draft 05/21/2021

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

General-purpose I/O (GPIO) pins can be individually configured to operate in a digital I/O mode or in an alternate function (AF) mode, which maps a signal associated with an enabled peripheral to that GPIO. Each GPIO supports 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 is still readable 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.

The GPIO are divided logically 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 varies 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\\_VSEL](#) are device-dependent in their usage. GPIO3 is controlled differently and has different features than the other GPIO ports in the MAX78000. See [MCR\\_GPIO3\\_CTRL](#) for details on using GPIO3.*

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 the [GPIOn\\_OUT](#) register or an enabled peripheral.
- Input data can be read from the [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.

### 5.1 Instances

[Table 5-1](#) 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 5-1: MAX78000 GPIO Pin Count*

Package	GPIO	PINS
81-CTBGA	GPIO0[30:0]	31
	GPIO1[9:0]	10
	GPIO2[7:0]	8
	GPIO3[1:0] <sup>†</sup>	2

*Note:* Refer to [Power Sequencer Registers \(PWRSEQ\)](#)for details on using GPIO3.

*Note:* Refer to the *MAX78000 device data sheet* for descriptions of each GPIO port pin's alternate functions.

## 5.2 Configuration

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

### 5.2.1 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. After a POR, the SWD is enabled by default 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.en[pin]* = 1
  - ◆ *GPIOn\_EN1.en[pin]* = 0
  - ◆ *GPIOn\_EN2.en[pin]* = 0
- Pullup/pulldown disabled, I/O in Hi-Z mode
  - ◆ *GPIOn\_PADCTRL0.mode[pin]* = 0
  - ◆ *GPIOn\_PADCTRL1.mode[pin]*
- Output mode disabled
  - ◆ *GPIOn\_OUTEN.en[pin]* = 0
- Interrupt disabled
  - ◆ *GPIOn\_INTEN.en[pin]* = 0

### 5.2.2 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.config[28]* = 0
  - b. *GPIOn\_EN1.config[28]* = 0
  - c. *GPIOn\_EN2.config[28]* = 0
2. Set device pin P0.29 for AF1 mode:
  - a. *GPIOn\_EN0.config[29]* = 0
  - b. *GPIOn\_EN1.config[29]* = 0
  - c. *GPIOn\_EN2.config[29]* = 0

*Note:* To use the SWD pins in I/O mode, set the desired GPIO pins for SWD AF and disable the SWD (*GCR\_.swd\_dis* = 1).

### 5.2.3 Pin Function Configuration

*Table 5-2* depicts the bit settings for the *GPIOOn\_EN0*, *GPIOOn\_EN1*, and *GPIOOn\_EN2* registers to configure a GPIO port pin's function. Each of the bits within these registers represents the configuration of a single pin on the GPIO port. For example, *GPIOO\_EN0.config[25]*, *GPIOO\_EN1.config[25]*, and *GPIOO\_EN2.config[25]* all represent configuration for device pin P0.25. See *Table 5-5* for a detailed example of how each of these bits applies to each GPIO device pin.

*Table 5-2: MAX78000 GPIO Pin Function Configuration*

MODE	<i>GPIOOn_EN0.config[pin]</i>	<i>GPIOOn_EN1.config[pin]</i>	<i>GPIOOn_EN2.config[pin]</i>
AF1	0	0	0
AF2	0	1	0
I/O (transition to AF1)	1	0	0
I/O (transition to AF2)	1	1	0

### 5.2.4 Input Mode Configuration

*Table 5-3* 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, *GPIOO\_PADCTRL1.config[25]*, *GPIOO\_PADCTRL0.config[25]*, *GPIOO\_PS.pull\_sel[25]*, and *GPIOO\_VSEL.v\_sel[25]* all represent configuration for device pin P0.25. See *Table 5-8* for a detailed example of how each of these bits applies to each GPIO device pin. Refer to the device's data sheet for details of specific electrical characteristics.

*Table 5-3: MAX78000 Input Mode Configuration*

Input Mode	Mode Select			Pullup/Pulldown Strength	Power Supply
	<i>GPIOOn_PADCTRL1.config[pin]</i>	<i>GPIOOn_PADCTRL0.config[pin]</i>	<i>GPIOOn_PS.pull_sel[pin]</i>	<i>GPIOOn_VSEL.v_sel[pin]</i>	
High-impedance	0	0	N/A	N/A	N/A
Weak Pullup to V <sub>DDIO</sub> (1MΩ)	0	1	0	0	0
Strong Pullup to V <sub>DDIO</sub> (25KΩ)	0	1	1	0	0
Weak Pulldown to V <sub>DDIOH</sub> (1MΩ)	1	0	0	1	1
Strong Pulldown to V <sub>DDIOH</sub> (25KΩ)	1	0	1	1	1
Reserved	1	1	N/A	N/A	N/A

### 5.2.5 Output Mode Configuration

*Table 5-4* 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.config[25]*, *GPIO2\_DS1.config[25]*, and *GPIO2\_VSEL.v\_sel[25]* all represent configuration for GPIO port 2 pin 25 (device pin P0.25). See *Table 5-8* for a detailed

example of how each of these bits applies to each GPIO device pin. Refer to the data sheet for details of specific electrical characteristics.

*Table 5-4: MAX78000 Output Mode Configuration*

Input Mode	Drive Strength		Power Supply
	<i>GPIOOn_DS1.config[pin]</i>	<i>GPIOOn_DSO.config[pin]</i>	<i>GPIOOn_VSEL.v_sel[pin]</i>
Output Drive Strength 0, V <sub>DDIO</sub> Supply	0	0	0
Output Drive Strength 1, V <sub>DDIO</sub> Supply	0	1	0
Output Drive Strength 2, V <sub>DDIO</sub> Supply	1	0	0
Output Drive Strength 3, V <sub>DDIO</sub> Supply	1	1	0
Output Drive Strength 0, V <sub>DDIOH</sub> Supply	0	0	1
Output Drive Strength 1, V <sub>DDIOH</sub> Supply	0	1	1
Output Drive Strength 2, V <sub>DDIOH</sub> Supply	1	0	1
Output Drive Strength 3, V <sub>DDIOH</sub> Supply	1	1	1

Each GPIO port is assigned a dedicated interrupt vector, as shown in *Table 5-9*.

### 5.3 Reference Tables

The tables in this section provide example references for register bit assignment to configure a device's GPIO port 0 pins. Other GPIO port pins are configured similarly using the respective GPIO1 or GPIO2 registers.

*Table 5-5: MAX78000 GPIO0 Alternate Function Configuration Reference*

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

*Table 5-6: MAX78000 GPIO0 Output/Input Configuration Reference*

Device Pin	GPIO Output Enable	GPIO Output Write	GPIO Input Enable	GPIO Input Read
P0.0	<i>GPIO0_OUTEN.en[0]</i>	<i>GPIO0_OUT.level[0]</i>	<i>GPIO0_INEN.en[0]</i>	<i>GPIO0_IN.level[0]</i>
P0.1	<i>GPIO0_OUTEN.en[1]</i>	<i>GPIO0_OUT.level[1]</i>	<i>GPIO0_INEN.en[1]</i>	<i>GPIO0_IN.level[1]</i>
...	...	...	...	...
P0.30	<i>GPIO0_OUTEN.en[30]</i>	<i>GPIO0_OUT.level[30]</i>	<i>GPIO0_INEN.en[30]</i>	<i>GPIO0_IN.level[30]</i>
P0.31	<i>GPIO0_OUTEN.en[31]</i>	<i>GPIO0_OUT.level[31]</i>	<i>GPIO0_INEN.en[31]</i>	<i>GPIO0_IN.level[31]</i>

*Table 5-7: MAX78000 GPIO0 Interrupt Configuration Reference*

Device Pin	Enable	Status	Dual Edge	Polarity	Trigger	Wakeup
P0.0	<i>GPIO0_INTEN.en[0]</i>	<i>GPIO0_INFL.config[0]</i>	<i>GPIO0_DUALEDGE.dualedge[0]</i>	<i>GPIO0_INTPOL.pol[0]</i>	<i>GPIO0_INTMODE gpio_intmode[0]</i>	<i>GPIO0_WKEN.en[0]</i>
P0.1	<i>GPIO0_INTEN.en[1]</i>	<i>GPIO0_INFL.config[1]</i>	<i>GPIO0_DUALEDGE.config[1]</i>	<i>GPIO0_INTPOL.pol[1]</i>	<i>GPIO0_INTMODE gpio_intmode[1]</i>	<i>GPIO0_WKEN.en[1]</i>
...	...	...	...	...	...	...
P0.30	<i>GPIO0_INTEN.en[30]</i>	<i>GPIO0_INFL.int[30]</i>	<i>GPIO0_DUALEDGE gpio_dualedge[30]</i>	<i>GPIO0_INTPOL.pol[30]</i>	<i>GPIO0_INTMODE gpio_intmode[30]</i>	<i>GPIO0_WKEN.en[30]</i>
P0.31	<i>GPIO0_INTEN.en[31]</i>	<i>GPIO0_INFL.int[31]</i>	<i>GPIO0_DUALEDGE gpio_dualedge[31]</i>	<i>GPIO0_INTPOL.pol[31]</i>	<i>GPIO0_INTMODE gpio_intmode[31]</i>	<i>GPIO0_WKEN.en[31]</i>

*Table 5-8: MAX78000 GPIO Pullup/Pulldown/Drive Strength/Voltage Configuration Reference*

Device Pin	Pullup/Pulldown/Strength Select			Drive Strength		Voltage
P0.0	<i>GPIO0_PADCTRL0.config[0]</i>	<i>GPIO0_PADCTRL1.config[0]</i>	<i>GPIO0_PS.pull_sel[0]</i>	<i>GPIO0_DS0.config[0]</i>	<i>GPIO0_DS1.config[0]</i>	<i>GPIOOn_VSEL.v_sel[0]</i>
P0.1	<i>GPIO0_PADCTRL0.config[1]</i>	<i>GPIO0_PADCTRL1.config[1]</i>	<i>GPIO0_PS.pull_sel[1]</i>	<i>GPIO0_DS0.config[1]</i>	<i>GPIO0_DS1.config[1]</i>	<i>GPIOOn_VSEL.v_sel[1]</i>
...	...	...	...	...	...	...
P0.30	<i>GPIO0_PADCTRL0.config[30]</i>	<i>GPIO0_PADCTRL1.config[30]</i>	<i>GPIO0_PS.pull_sel[30]</i>	<i>GPIO0_DS0.config[30]</i>	<i>GPIO0_DS1.config[30]</i>	<i>GPIOOn_VSEL.v_sel[30]</i>
P0.31	<i>GPIO0_PADCTRL0.config[31]</i>	<i>GPIO0_PADCTRL1.config[31]</i>	<i>GPIO0_PS.pull_sel[31]</i>	<i>GPIO0_DS0.config[31]</i>	<i>GPIO0_DS1.config[31]</i>	<i>GPIOOn_VSEL.v_sel[31]</i>

## 5.4 Usage

### 5.4.1 Reset State

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

1. The GPIO configuration enable bits shown in *Table 5-2* are set to I/O (transition to AF1) mode.
2. Input mode is enabled (*GPIOOn\_INEN.en[pin]* = 1).
3. High impedance mode enabled (*GPIOOn\_PADCTRL1.config[pin]* = 0, *GPIOOn\_PADCTRL0.config[pin]* = 0), pullup and pulldown disabled.
4. Output mode disabled (*GPIOOn\_OUTEN.en[pin]* = 0).
5. Interrupt disabled (*GPIOOn\_INTEN.en[pin]* = 0).

### 5.4.2 Input Mode Configuration

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

1. Set the GPIO Configuration Enable bits shown in *Table 5-2* to any one of the I/O mode settings.
2. Configure the electrical characteristics of the pin as desired, as shown in *Table 5-3*.
3. Enable the input buffer connected to the GPIO pin by setting *GPIOOn\_INEN.en[pin]* to 1.
4. Read the input state of the pin using the *GPIOOn\_IN.level[pin]* field.

### 5.4.3 Output Mode Configuration

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

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

### 5.4.4 Alternate Function Configuration

Most GPIO support one or more alternate functions selected with the GPIO configuration enable bits shown in [Table 5-2](#).

The bits that select the AF must only be changed while the pin is in one of the I/O modes ([`GPIOn\_EN0`](#) = 1). The specific I/O mode must match the desired AF. For example, if a transition to AF1 is desired, first select the setting corresponding to I/O (transition to AF1). Then enable the desired mode by selecting the AF1 mode.

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

## 5.5 Configuring GPIO (External) Interrupts

Each GPIO pin supports external interrupt events when the GPIO is configured for I/O mode, and the input mode is enabled. If the GPIO is configured for an alternate peripheral 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 will occur 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 will always be set when the current interrupt configuration event occurs, but an interrupt will only be generated if explicitly enabled.

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

1. Disable interrupts by setting the [`GPIOn\_INTEN.en\[pin\]`](#) field to 0. This will prevent any new interrupts on the pin from triggering but will not clear previously triggered (pending) interrupts. The application can disable all interrupts for a GPIO port by writing 0 to the [`GPIOn\_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.clr\[pin\]`](#) bit.
3. Configure the pin for the desired interrupt event
4. Set [`GPIOn\_INTMODE.mode\[pin\]`](#) to select the desired interrupt.
5. For level triggered interrupts, the interrupt triggers on an input high ([`GPIOn\_INTPOL.pol\[pin\]`](#) = 0) or input low level.
6. For edge triggered interrupts, the interrupt triggers on a transition from low to high ([`GPIOn\_INTPOL.pol\[pin\]`](#) = 0) or high to low ([`GPIOn\_INTPOL.pol\[pin\]`](#) = 1).
7. Optionally set [`GPIOn\_DUALEdge.de\_en\[pin\]`](#) to 1 to trigger on both the rising and falling edges of the input signal.
  - a. Set [`GPIOn\_INTEN.en\[pin\]`](#) to 1 to enable the interrupt for the pin.

### 5.5.1 *GPIO Interrupt Handling*

Each GPIO port is assigned a dedicated interrupt vector, as shown in *Table 5-9*.

*Table 5-9: MAX78000 GPIO Port Interrupt Vector Mapping*

GPIO Interrupt Source	GPIO Interrupt Status Register	CM4 Interrupt Vector Number	RV32 Interrupt Vector Number	GPIO Interrupt Vector
GPIO0[31:0]	<i>GPIOn_INFL</i>	40	25	GPIO0_IRQn
GPIO1[9:0]	<i>GPIOn_INFL</i>	41	26	GPIO1_IRQn
GPIO2[7:0]	<i>GPIOn_INFL</i>	42	27	GPIO2_IRQn

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

1. Read the *GPIOn\_INFL* 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\_INFL* register by writing a 1 to the *GPIOn\_INFL\_CLR* bit position that triggered the interrupt; this also clears and rearms the edge detectors for edge-triggered interrupts.
4. Return from the interrupt vector handler.

### 5.5.2 *Using GPIO for Wakeup from Low-Power Modes*

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

A single wakeup interrupt vector, *GPIOAKE\_IRQn*, is assigned for all pins of all GPIO ports. When the GPIO wakeup event occurs, the application software must interrogate each *GPIOn\_INFL* register to determine which external port pin caused the wakeup event.

*Table 5-10: MAX78000 GPIO Wakeup Interrupt Vector*

GPIO Wake Interrupt Source	GPIO Wake Interrupt Status Register	CM4 Interrupt Vector Number	RV32 Interrupt Vector Number	GPIO Wakeup Interrupt Vector
GPIO0	<i>GPIO0_INFL</i>	70	6	GPIOAKE_IRQn
GPIO1	<i>GPIO1_INFL</i>	70	6	GPIOAKE_IRQn
GPIO2	<i>GPIO2_INFL</i>	70	6	GPIOAKE_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\_INFL\_CLR.clr[pin]*.
2. Activate the GPIO wakeup function by writing a logic 1 to *GPIOn\_WKEN.we[pin]*.
3. Configure the power manager to use the GPIO as a wakeup source by *GCR\_PM.gpiowken* field to 1.

## 5.6 Registers

See [Table 2-4](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own independent set of the registers shown in [Table 5-11](#). 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 2-1](#) 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 5-11: GPIO Register Summary*

Offset	Register	Description
[0x0000]	<a href="#">GPIO_n_EN0</a>	GPIO Port n Configuration Enable Bit 0 Register
[0x0004]	<a href="#">GPIO_n_EN0_SET</a>	GPIO Port n Configuration Enable Atomic Set Bit 0 Register
[0x0008]	<a href="#">GPIO_n_EN0_CLR</a>	GPIO Port n Configuration Enable Atomic Clear Bit 0 Register
[0x000C]	<a href="#">GPIO_n_OUTEN</a>	GPIO Port n Output Enable Register
[0x0010]	<a href="#">GPIO_n_OUTEN_SET</a>	GPIO Port n Output Enable Atomic Set Register
[0x0014]	<a href="#">GPIO_n_OUTEN_CLR</a>	GPIO Port n Output Enable Atomic Clear Register
[0x0018]	<a href="#">GPIO_n_OUT</a>	GPIO Port n Output Register
[0x001C]	<a href="#">GPIO_n_OUT_SET</a>	GPIO Port n Output Atomic Set Register
[0x0020]	<a href="#">GPIO_n_OUT_CLR</a>	GPIO Port n Output Atomic Clear Register
[0x0024]	<a href="#">GPIO_n_IN</a>	GPIO Port n Input Register
[0x0028]	<a href="#">GPIO_n_INTMODE</a>	GPIO Port n Interrupt Mode Register
[0x002C]	<a href="#">GPIO_n_INTPOL</a>	GPIO Port n Interrupt Polarity Register
[0x0030]	<a href="#">GPIO_n_INEN</a>	GPIO Port n Input Enable Register
[0x0034]	<a href="#">GPIO_n_INTEN</a>	GPIO Port n Interrupt Enable Register
[0x0038]	<a href="#">GPIO_n_INTEN_SET</a>	GPIO Port n Interrupt Enable Atomic Set Register
[0x003C]	<a href="#">GPIO_n_INTEN_CLR</a>	GPIO Port n Interrupt Enable Atomic Clear Register
[0x0040]	<a href="#">GPIO_n_INTFL</a>	GPIO Port n Interrupt Status Register
[0x0048]	<a href="#">GPIO_n_INTFL_CLR</a>	GPIO Port n Interrupt Clear Register
[0x004C]	<a href="#">GPIO_n_WKEN</a>	GPIO Port n Wakeup Enable Register
[0x0050]	<a href="#">GPIO_n_WKEN_SET</a>	GPIO Port n Wakeup Enable Atomic Set Register
[0x0054]	<a href="#">GPIO_n_WKEN_CLR</a>	GPIO Port n Wakeup Enable Atomic Clear Register
[0x005C]	<a href="#">GPIO_n_DUALEDGE</a>	GPIO Port n Interrupt Dual Edge Mode Register
[0x0060]	<a href="#">GPIO_n_PADCTRL0</a>	GPIO Port n Pad Configuration 1 Register
[0x0064]	<a href="#">GPIO_n_PADCTRL1</a>	GPIO Port n Pad Configuration 2 Register
[0x0068]	<a href="#">GPIO_n_EN1</a>	GPIO Port n Configuration Enable Bit 1 Register
[0x006C]	<a href="#">GPIO_n_EN1_SET</a>	GPIO Port n Configuration Enable Atomic Set Bit 1 Register
[0x0070]	<a href="#">GPIO_n_EN1_CLR</a>	GPIO Port n Configuration Enable Atomic Clear Bit 1 Register
[0x0074]	<a href="#">GPIO_n_EN2</a>	GPIO Port n Configuration Enable Bit 2 Register
[0x0078]	<a href="#">GPIO_n_EN2_SET</a>	GPIO Port n Configuration Enable Atomic Set Bit 2 Register
[0x007C]	<a href="#">GPIO_n_EN2_CLR</a>	GPIO Port n Configuration Enable Atomic Clear Bit 2 Register
[0x00A8]	<a href="#">GPIO_n_HYSEN</a>	GPIO Port n Hysteresis Enable Register
[0x00AC]	<a href="#">GPIO_n_SRSEL</a>	GPIO Port n Slew Rate Select Register
[0x00B0]	<a href="#">GPIO_n_DSO</a>	GPIO Port n Output Drive Strength Bit 0 Register
[0x00B4]	<a href="#">GPIO_n_DS11</a>	GPIO Port n Output Drive Strength Bit 1 Register

Offset	Register	Description
[0x00B8]	<a href="#">GPIOn_PS</a>	<i>GPIO Port n Pulldown/Pullup Strength Select Register</i>
[0x00C0]	<a href="#">GPIOn_VSEL</a>	<i>GPIO Port n Voltage Select Register</i>

### 5.6.1 Register Details

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

GPIO Port n Configuration Enable Bit 0			GPIOOn_EN0		[0x0000]
Bits	Field	Access	Reset	Description	
31:0	config	R/W	1	<b>GPIO Configuration Enable Bit 0</b> In conjunction with the bits in <a href="#">Table 5-2</a> , this field configures 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 <a href="#">GPIOOn_EN0_SET</a> or <a href="#">GPIOOn_EN0_CLR</a> . <i>Table 5-5</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. Bits associated with unimplemented GPIO should not be changed from their default value.</i> <i>Note: This register setting does not affect the input and interrupt functionality of the associated pin.</i>	

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

GPIO Port n Configuration Enable Atomic Set Bit 0			GPIOOn_EN0_SET		[0x0004]
Bits	Field	Access	Reset	Description	
31:0	set	R/W1	0	<b>GPIO Configuration Enable Atomic Set Bit 0</b> Writing 1 to one or more bits sets the corresponding bits in the <a href="#">GPIOOn_EN0</a> register. 0: No effect. 1: Corresponding bits in <a href="#">GPIOOn_EN0</a> register set to 1.	

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

GPIO Port n Configuration Enable Atomic Clear Bit 0			GPIOOn_EN0_CLR		[0x0008]
Bits	Field	Access	Reset	Description	
31:0	clr	R/W1	0	<b>GPIO Configuration Enable Atomic Clear Bit 0</b> Writing 1 to one or more bits clears the corresponding bits in the <a href="#">GPIOOn_EN0</a> register. 0: No effect. 1: Corresponding bits in <a href="#">GPIOOn_EN0</a> register cleared to 0.	

Table 5-15: *GPIO Port n Output Enable Register*

GPIO Port n Output Enable			GPIOOn_OUTEN		[0x000C]
Bits	Field	Access	Reset	Description	
31:0	en	R/W	0	<b>GPIO Output Enable</b> 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 <a href="#">GPIOOn_OUTEN_SET</a> or <a href="#">GPIOOn_OUTEN_CLR</a> . 0: Pin is set to input mode; output driver disabled. 1: Pin is set to output mode.	

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

GPIO Port n Output Enable Atomic Set			GPIOn_OUTEN_SET		[0x0010]
Bits	Field	Access	Reset	Description	
31:0	set	R/W1	0	<b>GPIO Output Enable Atomic Set</b> Writing 1 to one or more bits sets the corresponding bits in the <a href="#">GPIOn_OUTEN</a> register.  0: No effect. 1: Corresponding bits in <a href="#">GPIOn_OUTEN</a> set to 1.	

Table 5-17: 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	clr	R/W1	0	<b>GPIO Output Enable Atomic Clear</b> Writing 1 to one or more bits sets the corresponding bits in the <a href="#">GPIOn_OUTEN</a> register.  0: No effect. 1: Corresponding bits in <a href="#">GPIOn_OUTEN</a> cleared to 0.	

Table 5-18: GPIO Port n Output Register

GPIO Port n Output			GPIOn_OUT		[0x0018]
Bits	Field	Access	Reset	Description	
31:0	level	R/W	0	<b>GPIO Output</b> Set the corresponding output pin high or low.  0: Drive the corresponding output pin low (logic 0). 1: Drive the corresponding output pin high (logic 1).	

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

GPIO Port n Output Atomic Set			GPIOn_OUT_SET		[0x001C]
Bits	Field	Access	Reset	Description	
31:0	set	R/W1	0	<b>GPIO Output Atomic Set</b> Writing 1 to one or more bits sets the corresponding bits in the <a href="#">GPIOn_OUT</a> register.  0: No effect. 1: Corresponding bits in <a href="#">GPIOn_OUTEN</a> set to 1.	

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

GPIO Port n Output Atomic Clear			GPIOn_OUT_CLR		[0x0020]
Bits	Field	Access	Reset	Description	
31:0	clr	WO	0	<b>GPIO Output Atomic Clear</b> Writing 1 to one or more bits clears the corresponding bits in the <a href="#">GPIOn_OUT</a> register.  0: No effect. 1: Corresponding bits in <a href="#">GPIOn_OUTEN</a> cleared to 0.	

Table 5-21: GPIO Port n Input Register

GPIO Port n Input		GPIOn_IN			[0x0024]
Bits	Field	Access	Reset	Description	
31:0	level	RO	-	<b>GPIO Input</b> Returns the state of the input pin only if the corresponding bit in the <a href="#">GPIOn_INEN</a> register is set. The state is not affected by the pin's configuration as an output or alternate function. 0: Input pin low 1: Input pin high.	

Table 5-22: GPIO Port n Interrupt Mode Register

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

Table 5-23: GPIO Port n Interrupt Polarity Register

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

Table 5-24: GPIO Port n Input Enable Register

GPIO Port n Input Enable		GPIOn_INEN			[0x0030]
Bits	Field	Access	Reset	Description	
31:0	en	R/W	1	<b>GPIO Input Enable</b> Connects the corresponding input pad to the specified input pin for reading the pin state using the <a href="#">GPIOn_IN</a> register. 0: Input not connected. 1: Input pin connected to the pad for reading through <a href="#">GPIOn_IN</a> register.	

Table 5-25: GPIO Port n Interrupt Enable Register

GPIO Port n Interrupt Enable				GPIOOn_INTEN	[0x0034]
Bits	Field	Access	Reset	Description	
31:0	ie	R/W	0	<b>GPIO Interrupt Enable</b> Enable or disable the interrupt for the corresponding GPIO pin. 0: Disabled. 1: Enabled. <i>Note: Disabling a GPIO interrupt does not clear pending interrupts for the associated pin. Use the <a href="#">GPIOOn_INTFL_CLR</a> register to clear pending interrupts.</i>	

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

GPIO Port Interrupt Enable Atomic Set				GPIOOn_INTEN_SET	[0x0038]
Bits	Field	Access	Reset	Description	
31:0	set	R/W1	0	<b>GPIO Interrupt Enable Atomic Set</b> Writing 1 to one or more bits sets the corresponding bits in the <a href="#">GPIOOn_INTEN</a> register. 0: No effect. 1: Corresponding bits in <a href="#">GPIOOn_INTEN</a> register set to 1.	

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

GPIO Port Interrupt Enable Atomic Clear				GPIOOn_INTEN_CLR	[0x003C]
Bits	Field	Access	Reset	Description	
31:0	clr	R/W1	0	<b>GPIO Interrupt Enable Atomic Clear</b> Writing 1 to one or more bits clears the corresponding bits in the <a href="#">GPIOOn_INTEN</a> register. 0: No effect. 1: Corresponding bits in <a href="#">GPIOOn_INTEN</a> register cleared to 0.	

Table 5-28: GPIO Port n Interrupt Status Register

GPIO Port Interrupt Status				GPIOOn_INTFL	[0x0040]
Bits	Field	Access	Reset	Description	
31:0	if	RO	0	<b>GPIO Interrupt Status</b> An interrupt is pending for the associated GPIO pin when this bit reads 1. 0: No GPIO interrupt is pending for the associated GPIO pin. 1: A GPIO interrupt is pending for the associated GPIO pin. <i>Note: Write a 1 to the corresponding bit in the <a href="#">GPIOOn_INTFL_CLR</a> register to clear the interrupt pending status flag.</i>	

Table 5-29: GPIO Port n Interrupt Clear Register

GPIO Port Interrupt Clear				GPIOOn_INTFL_CLR	[0x0048]
Bits	Field	Access	Reset	Description	
31:0	clr	R/W1C	0	<b>GPIO Interrupt Clear</b> Write 1 to clear the associated interrupt status ( <a href="#">GPIOOn_INTFL</a> ). 0: No effect on the associated <a href="#">GPIOOn_INTFL</a> flag. 1: Clear the associated interrupt pending flag in the <a href="#">GPIOOn_INTFL</a> register.	

Table 5-30: GPIO Port n Wakeup Enable Register

GPIO Port n Wakeup Enable				GPIOOn_WKEN	[0x004C]
Bits	Field	Access	Reset	Description	
31:0	we	R/W	0	<b>GPIO Wakeup Enable</b> Enable the I/O as a wakeup source from <i>SLEEP</i> , <i>DEEPSLEEP</i> , and <i>BACKUP</i> . 0: The GPIO pin is not enabled as a wakeup source from low-power modes. 1: The GPIO pin is enabled as a wakeup source from low-power modes.	

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

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

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

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

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

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

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

GPIO Port n Pad Configuration 1				GPIOOn_PADCTRL0	[0x0060]
Bits	Field	Access	Reset	Description	
31:0	config	R/W	0	<b>GPIO Pad Configuration 1</b> Input mode configuration for the associated GPIO pin. Input mode selection and the selection of a weak or strong pullup or weak or strong pulldown resistor are described in <i>Table 5-3</i> .	

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

GPIO Port n Pad Configuration 2				GPIOOn_PADCTRL1	[0x0064]
Bits	Field	Access	Reset	Description	
31:0	config	R/W	0	<b>GPIO Pad Configuration 2</b> Input mode configuration for the associated GPIO pin. Input mode selection and the selection of a weak or strong pullup or weak or strong pulldown resistor are described in <i>Table 5-3</i> .	

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

GPIO Port n Configuration Enable Bit 1			GPIOOn_EN1		[0x0068]
Bits	Field	Access	Reset	Description	
31:0	config	R/W	0	<b>GPIO Configuration Enable Bit 1</b> In conjunction with the bits in <a href="#">Table 5-2</a> , this field configures 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 the <a href="#">GPIOOn_EN1_SET</a> or <a href="#">GPIOOn_EN1_CLR</a> registers. <i>Note: Some GPIO are not implemented in all devices. The bits associated with unimplemented GPIO should not be changed from their default value.</i> <i>Note: This register setting does not affect input and interrupt functionality of the associated pin.</i>	

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

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

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

GPIO Port n Configuration Enable Atomic Clear Bit 1			GPIOOn_EN1_CLR		[0x0070]
Bits	Field	Access	Reset	Description	
31:0	clr	R/W1	0	<b>GPIO Configuration Enable Atomic Clear Bit 1</b> Writing 1 to one or more bits clears the corresponding bits in the <a href="#">GPIOOn_EN1</a> register. 0: No effect. 1: Corresponding bits in <a href="#">GPIOOn_EN1</a> register cleared to 0.	

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

GPIO Port n Configuration Enable Bit 2			GPIOOn_EN2		[0x0074]
Bits	Field	Access	Reset	Description	
31:0	config	R/W	0	<b>GPIO Configuration Enable Bit 2</b> In conjunction with the bits in <a href="#">Table 5-2</a> , this field configures 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 <a href="#">GPIOOn_EN2_SET</a> or <a href="#">GPIOOn_EN2_CLR</a> . <i>Note: Some GPIO are not implemented in all devices. The bits associated with unimplemented GPIO should not be changed from their default value.</i> <i>Note: This register setting does not affect input and interrupt functionality of the associated pin.</i>	

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

GPIO Port n Configuration Enable Atomic Set Bit 2				GPIOOn_EN2_SET	[0x0078]
Bits	Field	Access	Reset	Description	
31:0	set	R/W1	0	<b>GPIO Alternate Function Select Atomic Set Bit 2</b> Writing 1 to one or more bits sets the corresponding bits in the <i>GPIOOn_EN2</i> register. 0: No effect. 1: Corresponding bits in <i>GPIOOn_EN2</i> register set to 1.	

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

GPIO Port n Configuration Enable Atomic Clear Bit 2				GPIOOn_EN2_CLR	[0x007C]
Bits	Field	Access	Reset	Description	
31:0	clr	R/W1	0	<b>GPIO Alternate Function Select Atomic Clear Bit 2</b> Writing 1 to one or more bits clears the corresponding bits in the <i>GPIOOn_EN2</i> register. 0: No effect. 1: Corresponding bits in <i>GPIOOn_EN2</i> register cleared to 0.	

Table 5-42: GPIO Port n Hysteresis Enable Register

GPIO Port n Hysteresis Enable				GPIOOn_HYSEN	[0x00A8]
Bits	Field	Access	Reset	Description	
31:0	en	RO	0	Reserved	

Table 5-43: GPIO Port n Output Drive Strength Bit 0 Register

GPIO Port n Output Drive Strength Bit 0				GPIOOn_SRSEL	[0x00AC]
Bits	Field	Access	Reset	Description	
31:0	sr_sel	RO	0	Reserved	

Table 5-44: GPIO Port n Output Drive Strength Bit 0 Register

GPIO Port n Output Drive Strength Bit 0				GPIOOn_DSO	[0x00B0]
Bits	Field	Access	Reset	Description	
31:0	config	R/W	0	<b>GPIO Output Drive Strength Selection 0</b> See <i>Table 5-4</i> for details on setting the GPIO output drive strength and other electrical characteristics.	

Table 5-45: GPIO Port n Output Drive Strength Bit 1 Register

GPIO Port n Output Drive Strength Bit 1				GPIOOn_DS1	[0x00B4]
Bits	Field	Access	Reset	Description	
31:0	config	R/W	0	<b>GPIO Output Drive Strength Selection 1</b> See <i>Table 5-4</i> for details on setting the GPIO output drive strength and other electrical characteristics.	

Table 5-46: GPIO Port n Pulldown/Pullup Strength Select Register

GPIO Port n Pulldown/Pullup Strength Select				GPIOOn_PS	[0x00B8]
Bits	Field	Access	Reset	Description	
31:0	pull_sel	R/W	0	<b>GPIO Pulldown/Pullup Strength Select</b> This field selects the strength of the pullup or pulldown resistor for a pin configured for input mode. 0: Weak pulldown/pullup resistor for input pin. 1: Strong pulldown/pullup resistor for input pin. <i>Note: Refer to the data sheet for specific electrical characteristics of the Pulldown/Pullup resistances.</i>	

Table 5-47: GPIO Port n Voltage Select Register

GPIO Port n Voltage Select				GPIOn_VSEL	[0x00C0]
Bits	Field	Access	Reset	Description	
31:0	v_sel	R/W	0	<b>GPIO Supply Voltage Select</b> This field selects the voltage rail used for the GPIO pin. 0: $V_{DDIO}$ 1: $V_{DDIOH}$	

Preliminary Draft 05/21/2021

## 6. Flash Controller (FLC)

The MAX78000 flash controller manages read, write, and erase accesses to the internal flash and provides the following features:

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

### 6.1 Instances

The device includes one instance of the FLC. The 512KB of internal flash memory is programmable through serial wire debug interface (in-system) or directly with software (in-application).

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

*Table 6-1: MAX78000 Internal Flash Memory Organization*

Instance Number	Page Number	Size (per page)	Start Address	End Address
FLC0	1	8,192 Bytes	0x1000 0000	0x1000 1FFF
	2	8,192 Bytes	0x1000 2000	0x1000 3FFF
	3	8,192 Bytes	0x1000 4000	0x1000 5FFF
	4	8,192 Bytes	0x1000 6000	0x1000 7FFF
	...	...	...	...
	63	8,192 Bytes	0x1007 C000	0x1007 DFFF
	64	8,192 Bytes	0x1007 E000	0x1007 FFFF

### 6.2 Usage

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

#### 6.2.1 Clock Configuration

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

*Equation 6-1: FLC Clock Frequency*

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

### 6.2.2 Lock Protection

A locking mechanism prevents accidental memory writes and erases. All write and erase operations require the [\*FLCn\\_CTRL.unlock\*](#) field are set to 2 prior to starting the operation. Writing any other value to this field, [\*FLCn\\_CTRL.unlock\*](#), 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, [\*FLCn\\_INTR.af\*](#), to indicate an access violation occurred.*

### 6.2.3 Flash Write Width

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

Table 6-2: Valid Addresses Flash Writes

	FLCn_ADDR[31:0]																															
Bit Number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
128-bit Write	1	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	

### 6.2.4 Flash Write

Writes to a flash 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 [\*FLCn\\_INTR.afie\*](#) and [\*FLCn\\_INTR.doneie\*](#) bits.
2. Read the [\*FLCn\\_CTRL.pend\*](#) bit until it returns 0.
3. Configure the [\*FLCn\\_CLKDIV.clkdiv\*](#) field to achieve a 1MHz frequency based on the selected SYS\_CLK frequency.
4. Set the [\*FLCn\\_ADDR\*](#) register to a valid target address. See [Table 6-2](#) for details.
5. Set [\*FLCn\\_DATA3\*](#), [\*FLCn\\_DATA2\*](#), [\*FLCn\\_DATA1\*](#), and [\*FLCn\\_DATA0\*](#) to the data to write.
  - a. [\*FLCn\\_DATA3\*](#) is the most significant word, and [\*FLCn\\_DATA0\*](#) 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 [\*FLCn\\_CTRL.unlock\*](#) field to 2 to unlock the flash.
7. Set the [\*FLCn\\_CTRL.wr\*](#) field to 1.
  - a. The hardware automatically clears this field when the write operation is complete.
8. The [\*FLCn\\_INTR.done\*](#) field is set to 1 by hardware when the write completes, and an interrupt is generated if the [\*FLCn\\_INTR.doneie\*](#) is set to 1.
  - a. If an error occurred, the [\*FLCn\\_INTR.af\*](#) field is set to 1 by hardware, and an interrupt is generated if the [\*FLCn\\_INTR.afie\*](#) field is set to 1.
9. Set the [\*FLCn\\_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.*

### 6.2.5 Page Erase

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

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

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

### 6.2.6 Mass Erase

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

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

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

## 6.3 Registers

See *Table 2-4* for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own independent set of the registers shown in *Table 6-3*. Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn\_CTRL resolves to PERIPHERAL0\_CTRL and PERIPHERAL1\_CTRL for instances 0 and 1, respectively.

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

*Table 6-3: Flash Controller Register Summary*

Offset	Register Name	Access	Description
[0x0000]	<i>FLCn_ADDR</i>	R/W	Flash Controller Address Pointer Register
[0x0004]	<i>FLCn_CLKDIV</i>	R/W	Flash Controller Clock Divisor Register
[0x0008]	<i>FLCn_CTRL</i>	R/W	Flash Controller Control Register

Offset	Register Name	Access	Description
[0x0024]	<i>FLCn_INTR</i>	R/W	Flash Controller Interrupt Register
[0x0030]	<i>FLCn_DATA0</i>	R/W	Flash Controller Data Register 0
[0x0034]	<i>FLCn_DATA1</i>	R/W	Flash Controller Data Register 1
[0x0038]	<i>FLCn_DATA2</i>	R/W	Flash Controller Data Register 2
[0x003C]	<i>FLCn_DATA3</i>	R/W	Flash Controller Data Register 3
[0x0040]	<i>FLCn_ACNTL</i>	R/W	Flash Controller Access Control Register
[0x0080]	<i>FLCn_WELR0</i>	R/W	Flash Write/Erase Lock 0 Register
[0x0088]	<i>FLCn_WELR1</i>	R/W	Flash Write/Erase Lock 1 Register
[0x0090]	<i>FLCn_RLR0</i>	R/W	Flash Read Lock 0 Register
[0x0098]	<i>FLCn_RLR1</i>	R/W	Flash Read Lock 1 Register

### 6.3.1 Register Details

Table 6-4: Flash Controller Address Pointer Register

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

Table 6-5: Flash Controller Clock Divisor Register

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

Table 6-6: Flash Controller Control Register

Flash Controller Control			FLCn_CTRL		[0x0008]
Bits	Name	Access	Reset	Description	
31:28	unlock	R/W	0	<b>Flash Unlock</b> Write the unlock code, 2, prior to any flash write or erase operation to unlock the flash. Writing any other value to this field locks the internal flash. 2: Flash unlock code	
27:26	-	RO	-	<b>Reserved</b>	
25	lve	R/W	0	<b>Low Voltage Enable</b> 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.	

Flash Controller Control				FLCn_CTRL	[0x0008]
Bits	Name	Access	Reset	Description	
24	pend	RO	0	<b>Flash Busy Flag</b> When this field is set, writes to all of the flash registers, except the <a href="#">FLCn_INTR</a> register, are ignored by the Flash Controller. This bit will be cleared by hardware once the flash becomes accessible. <i>Note: If the Flash Controller is busy (FLCn_CTRL.pend = 1), reads, writes, and erase operations are not allowed and result in an access failure (FLCn_INTR.af = 1).</i> 0: Flash idle 1: Flash busy	
23:16	-	RO	0	<b>Reserved</b>	
15:8	erase_code	R/W	0	<b>Erase Code</b> Prior to an erase operation, this field must be set to 0x55 for a page erase or 0xAA for a mass erase. The flash must be unlocked prior to setting the erase code. This field is automatically cleared after the erase operation is complete. 0x00: Erase disabled. 0x55: Page erase code. 0xAA: Enable mass erase.	
7:3	-	RO	0	<b>Reserved</b>	
2	pge	R/W1	0	<b>Page Erase</b> Write a 1 to this field to initiate a page erase at the address in <a href="#">FLCn_ADDR.addr</a> . The flash must be unlocked prior to attempting a page erase. See <a href="#">FLCn_CTRL.unlock</a> for details. The Flash Controller hardware clears this bit when a page erase operation is complete. 0: No page erase operation in process or page erase is complete. 1: Write a 1 to initiate a page erase. If this field reads 1, a page erase operation is in progress.	
1	me	R/W1	0	<b>Mass Erase</b> Write a 1 to this field to initiate a mass erase of the internal flash memory. The flash must be unlocked prior to attempting a mass erase. See <a href="#">FLCn_CTRL.unlock</a> for details. The Flash Controller hardware clears this bit when the mass erase operation completes. 0: No operation 1: Initiate mass erase	
0	wr	R/W10	0	<b>Write</b> If this field reads 0, no write operation is pending for the flash. To initiate a write operation, set this bit to 1, and the Flash Controller will write to the address set in the <a href="#">FLCn_ADDR</a> register. 0: No write operation in process or the write operation is complete. 1: Write 1 to initiate a write operation. If this field reads 1, a write operation is in progress. <i>Note: This field is protected and cannot be set to 0 by application code.</i>	

Table 6-7: Flash Controller Interrupt Register

Flash Controller Interrupt				FLCn_INTR	[0x0024]
Bits	Name	Access	Reset	Description	
31:10	-	RO	0	<b>Reserved</b>	

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

Table 6-8: Flash Controller Data 0 Register

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

Table 6-9: Flash Controller Data Register 1

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

Table 6-10: Flash Controller Data Register 2

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

Table 6-11: Flash Controller Data Register 3

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

Table 6-12: Flash Controller Access Control Register

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

Table 6-13: Flash Write/Lock 0 Register

Flash Write/Lock 0			FLCn_WELR0		[0x0080]
Bits	Name	Access	Reset	Description	
31:0	welr0	R/W1C	0xFFFFFFFF	<b>Flash Write/Lock Bit</b> Each bit in this register maps to a page of the internal flash. <i>FLCn_WELR0</i> [0] maps to page 0 of the flash, and <i>FLCn_WELR0</i> [31] maps to page 31. Each flash page is 8,192 bytes. Write a 1 to a bit position in this register, and the corresponding page of flash is immediately locked. The page protection can only be unlocked by an external reset or a POR.  0: The corresponding page of flash is write protected. 1: The corresponding page of flash is <i>not</i> write protected.	

Table 6-14: Flash Write/Lock 1 Register

Flash Write/Lock 1			FLCn_WELR1		[0x0088]
Bits	Name	Access	Reset	Description	
31:0	welr1	R/W1C	0xFFFF FFFF	<b>Flash Write/Lock Bit</b> Each bit in this register maps to a page of the internal flash. <i>FLCn_WELR1</i> [0] maps to page 32 of the flash and <i>FLCn_WELR1</i> [31] maps to page 63 of flash. Each flash page is 8,192 bytes. Write a 1 to a bit position in this register and the corresponding page of flash is immediately 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 <i>not</i> write protected.	

*Table 6-15: Flash Read Lock 0 Register*

Flash Read Lock 0			FLCn_RLR0		[0x0090]
Bits	Name	Access	Reset	Description	
31:0	rlr0	R/W1C	0xFFFF FFFF	<b>Read Lock Bit</b> Each bit in this register maps to a page of the internal flash. <i>FLCn_RLR0</i> [0] maps to page 0 of the flash, and <i>FLCn_RLR0</i> [31] maps to page 31 of flash. Each flash page is 8,192 bytes. Write a 1 to a bit position in this register, and the corresponding page of flash is immediately read protected. The page's read protection can only be unlocked by an external reset or a POR. 0: The corresponding flash page is read protected. 1: The corresponding flash page is <i>not</i> read protected.	

*Table 6-16: Flash Read Lock 1 Register*

Flash Read Lock 1			FLCn_RLR1		[0x0098]
Bits	Name	Access	Reset	Description	
31:0	rlr1	R/W1C	0xFFFF FFFF	<b>Read Lock Bit</b> Each bit in this register maps to a page of the internal flash. <i>FLCn_RLR1</i> [0] maps to page 32 of the flash, and <i>FLCn_RLR1</i> [31] maps to page 63 of flash. Each flash page is 8,192 bytes. Write a 1 to a bit position in this register, and the corresponding page of flash is immediately read protected. The page's read protection can only be unlocked by an external reset or a POR. 0: The corresponding flash page is read protected. 1: The corresponding flash page is <i>not</i> read protected.	

## 7. Debug Access Port (DAP)

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

*GCR\_SYSST.iceclock* = 0 if the device provides a customer-accessible DAP.

### 7.1 Instances

The DAP interface communicates through the serial wire debug (SWD), or JTAG interface signals shown in *Table 7-1*.

*Table 7-1: MAX78000 DAP Instances*

Instance	Pin	Alternate Function	SWD Signal	JTAG Signal	Pullup
DAP0	P0.0	AF1	SWDIO	N/A	10kΩ to V <sub>DDIO</sub>
	P0.1	AF1	SWCLK	N/A	

### 7.2 Access Control

#### 7.2.1 Factory Disabled DAP

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

#### 7.2.2 Software Accessible DAP

Device versions that provide a DAP (*GCR\_SYSST.iceclock* = 0) always have their interface(s) enabled and running unless the software explicitly sets *GCR\_SYSCTRL.sw\_dis* field to 1. The read-only field *GCR\_SYSST.iceclock* is cleared to 0, and the software has read and write access to the *GCR\_SYSCTRL.sw\_dis* field. The *GCR\_SYSCTRL.sw\_dis* field resets to 0 after every POR to allow access to the DAP during development.

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

### 7.3 Pin Configuration

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

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

## 8. Semaphores

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

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

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

### 8.1 Instances

There is one instance of the semaphore peripheral, shown in [Table 8-1](#).

Table 8-1: MAX78000 Semaphore Instances

Instance	Number of Semaphores
SEMA	8

## 8.2 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 occurring if both cores attempt to modify the registers at the same time.

### 8.2.1 Reset

Globally reset the semaphore peripheral by setting [\*GCR\\_RST1.smphr\*](#) to 1.

### 8.2.2 CM4 Semaphore Interrupt Generation

The [\*SEMA IRQ0\*](#) register provides the ability to generate a CM4 semaphore interrupt. Setting the [\*SEMA IRQ0.cm4\\_irq\*](#) bit to 1 and then setting the [\*SEMA IRQ0.en\*](#) bit to 1 generates a CM4 semaphore 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.

### 8.2.3 RV32 Semaphore Interrupt Generation

The [\*SEMA IRQ1\*](#) register provides the ability to generate a RV32 semaphore interrupt. Setting the [\*SEMA IRQ1.rv32\\_irq\*](#) bit to 1 and then setting the [\*SEMA IRQ1.en\*](#) bit to 1 generates a RV32 semaphore 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.

## 8.3 Registers

See *Table 2-4* for the base address of this peripheral/module. See *Table 2-1* 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 8-2: Semaphore Register Summary*

Offset	Register	Name
[0x0000]	<i>SEMA_SEMAPHORES0</i>	Semaphore 0 Register
[0x0004]	<i>SEMA_SEMAPHORES1</i>	Semaphore 1 Register
[0x0008]	<i>SEMA_SEMAPHORES2</i>	Semaphore 2 Register
[0x000C]	<i>SEMA_SEMAPHORES3</i>	Semaphore 3 Register
[0x0010]	<i>SEMA_SEMAPHORES4</i>	Semaphore 4 Register
[0x0014]	<i>SEMA_SEMAPHORES5</i>	Semaphore 5 Register
[0x0018]	<i>SEMA_SEMAPHORES6</i>	Semaphore 6 Register
[0x0020]	<i>SEMA_SEMAPHORES7</i>	Semaphore 7 Register
[0x0040]	<i>SEMA IRQ0</i>	Semaphore Interrupt 0 Register
[0x0044]	<i>SEMA_MAIL0</i>	Semaphore Mailbox 0 Register
[0x0048]	<i>SEMA IRQ1</i>	Semaphore Interrupt 1 Register
[0x004C]	<i>SEMA_MAIL1</i>	Semaphore Mailbox 1 Register
[0x0100]	<i>SEMA_STATUS</i>	Semaphore Status Register

### 8.3.1 Register Details

*Table 8-3: Semaphore 0 Register*

Semaphore 0		SEMA_SEMAPHORES0			[0x0000]
Bits	Field	Access	Reset	Description	
31:1	-	RO	0	Reserved	
0	status	*	0	<b>Semaphore Status</b> 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 <i>SEMA_STATUS.status0</i> field. 0: Semaphore is available 1: Semaphore is taken	

*Table 8-4: Semaphore 1 Register*

Semaphore 1		SEMA_SEMAPHORES1			[0x0004]
Bits	Field	Access	Reset	Description	
31:1	-	RO	0	Reserved	
0	status	*	0	<b>Semaphore Status</b> 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 <i>SEMA_STATUS.status1</i> field. 0: Semaphore is available 1: Semaphore is taken	

Table 8-5: Semaphore 2 Register

Semaphore 2			SEMA_SEMAPHORES2		[0x0008]
Bits	Field	Access	Reset	Description	
31:1	-	RO	0	Reserved	
0	status	*	0	<b>Semaphore Status</b> 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 <a href="#">SEMA_STATUS.status2</a> field. 0: Semaphore is available 1: Semaphore is taken	

Table 8-6: Semaphore 3 Register

Semaphore 3			SEMA_SEMAPHORES3		[0x000C]
Bits	Field	Access	Reset	Description	
31:1	-	RO	0	Reserved	
0	status	*	0	<b>Semaphore Status</b> 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 <a href="#">SEMA_STATUS.status3</a> field. 0: Semaphore is available 1: Semaphore is taken	

Table 8-7: Semaphore 4 Register

Semaphore 4			SEMA_SEMAPHORES4		[0x0010]
Bits	Field	Access	Reset	Description	
31:1	-	RO	0	Reserved	
0	status	*	0	<b>Semaphore Status</b> 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 <a href="#">SEMA_STATUS.status4</a> field. 0: Semaphore is available 1: Semaphore is taken	

Table 8-8: Semaphore 5 Register

Semaphore 5			SEMA_SEMAPHORES5		[0x0014]
Bits	Field	Access	Reset	Description	
31:1	-	RO	0	Reserved	
0	status	*	0	<b>Semaphore Status</b> 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 <a href="#">SEMA_STATUS.status5</a> field. 0: Semaphore is available 1: Semaphore is taken	

Table 8-9: Semaphore 6 Register

Semaphore 6			SEMA_SEMAPHORES6		[0x0018]
Bits	Field	Access	Reset	Description	
31:1	-	RO	0	Reserved	

Semaphore 6			SEMA_SEMAPHORES6		[0x0018]
Bits	Field	Access	Reset	Description	
0	status	*	0	<b>Semaphore Status</b> 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 <a href="#">SEMA_STATUS.status6</a> field. 0: Semaphore is available 1: Semaphore is taken	

Table 8-10: Semaphore 7 Register

Semaphore 7			SEMA_SEMAPHORES7		[0x001C]
Bits	Field	Access	Reset	Description	
31:1	-	RO	0	<b>Reserved</b>	
0	status	*	0	<b>Semaphore Status</b> 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 <a href="#">SEMA_STATUS.status7</a> field. 0: Semaphore is available 1: Semaphore is taken	

Table 8-11: Semaphore Interrupt 0 Register

Semaphore Interrupt 0			SEMA_IRQ0		[0x0040]
Bits	Field	Access	Reset	Description	
31:17	-	RO	0	<b>Reserved</b>	
16	cm4_irq	R/W	0	<b>CM4 Interrupt</b> 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 <a href="#">SEMA_IRQ0.en</a> bit to 1.	
15:1	-	RO	0	<b>Reserved</b>	
0	en	R/W	0	<b>Interrupt Enable</b> Set this field to enable interrupt generation on semaphore events. 0: Interrupt disabled 1: Interrupt enabled	

Table 8-12: Semaphore Mailbox 0 Register

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

Table 8-13: Semaphore Interrupt 1 Register

Semaphore Interrupt 1			SEMA_IRQ1		[0x0048]
Bits	Field	Access	Reset	Description	
31:17	-	RO	0	<b>Reserved</b>	
16	rv32_irq	R/W	0	<b>RV32 Interrupt</b> 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 <a href="#">SEMA_IRQ1.en</a> bit to 1. 0: RV32 interrupt event not active or received by RV32. 1: RV32 interrupt event is generated when the <a href="#">SEMA_IRQ1.en</a> bit is also set to 1.	
15:1	-	RO	0	<b>Reserved</b>	
0	en	R/W	0	<b>Interrupt Enable</b> Set this field to generate a RV32 semaphore interrupt when the <a href="#">SEMA_IRQ1.rv32_irq</a> 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: Interrupt disabled 1: Interrupt enabled	

Table 8-14: Semaphore Mailbox 1 Register

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

Table 8-15: Semaphore Status Register

Semaphore Status			SEMA_STATUS		[0x0100]
Bits	Field	Access	Reset	Description	
31:8	-	RO	0	<b>Reserved</b>	
7	status7	R	0	<b>Semaphore 7 Status</b> This field mirrors the semaphore 7 status field. Reads from this field do not affect the corresponding semaphore's status field. 0: <a href="#">SEMA_SEMAPHORES7.status</a> is 0 1: <a href="#">SEMA_SEMAPHORES7.status</a> is 1	
6	status6	R	0	<b>Semaphore 6 Status</b> This field mirrors the semaphore 6 status field. Reads from this field do not affect the corresponding semaphore's status field. 0: <a href="#">SEMA_SEMAPHORES6.status</a> is 0 1: <a href="#">SEMA_SEMAPHORES6.status</a> is 1	

Semaphore Status			SEMA_STATUS		[0x0100]
Bits	Field	Access	Reset	Description	
5	status5	R	0	<b>Semaphore 5 Status</b> This field mirrors the semaphore 5 status field. Reads from this field do not affect the corresponding semaphore's status field. 0: SEMA_SEMAPHORES5.status is 0 1: SEMA_SEMAPHORES5.status is 1	
4	status4	R	0	<b>Semaphore 4 Status</b> This field mirrors the semaphore 4 status field. Reads from this field do not affect the corresponding semaphore's status field. 0: SEMA_SEMAPHORES4.status is 0 1: SEMA_SEMAPHORES4.status is 1	
3	status3	R	0	<b>Semaphore 3 Status</b> This field mirrors the semaphore 3 status field. Reads from this field do not affect the corresponding semaphore's status field. 0: SEMA_SEMAPHORES3.status is 0 1: SEMA_SEMAPHORES3.status is 1	
2	status2	R	0	<b>Semaphore 2 Status</b> This field mirrors the semaphore 2 status field. Reads from this field do not affect the corresponding semaphore's status field. 0: SEMA_SEMAPHORES2.status is 0 1: SEMA_SEMAPHORES2.status is 1	
1	status1	R	0	<b>Semaphore 1 Status</b> This field mirrors the semaphore 1 status field. Reads from this field do not affect the corresponding semaphore's status field. 0: SEMA_SEMAPHORES1.status is 0 1: SEMA_SEMAPHORES1.status is 1	
0	status0	R	0	<b>Semaphore 0 Status</b> This field mirrors the semaphore 0 status field. Reads from this field do not affect the corresponding semaphore's status field. 0: SEMA_SEMAPHORES0.status is 0 1: SEMA_SEMAPHORES0.status is 1	

## 9. Standard DMA (DMA)

The Standard Direct Memory Access controller (DMA) is a hardware feature that provides the ability to perform high-speed, block memory transfers of data independent of the device 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 memory address,
- to a transmit FIFO from a memory address, or
- from a source memory address to a destination memory address.

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

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

### 9.1 Instances

There is one instance of the DMA, generically referred to as DMA. Each instance provides four channels, generically referred to as DMA\_CH $n$ . Each instance of the DMA has a set of interrupt registers common to all its channels and a set of registers unique to each channel instance.

*Table 9-1: MAX78000 DMA and Channel Instances*

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

## 9.2 DMA Channel Operation (DMA\_CH)

### 9.2.1 Channel Arbitration and DMA Bursts

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

Granting is done based on priority; a higher priority request is always granted. Within a given priority level, requests are granted on a round-robin basis. The *DMA\_CH $n$ \_CTRL.pri* field determines the DMA channel priority.

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

A DMA channel is enabled using the *DMA\_CH $n$ \_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 completed.

Only an error condition can interrupt an ongoing data transfer.

### 9.2.2 Source and Destination Addressing

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

The *DMA\_CHn\_CTRL*.srcinc field is ignored when the DMA source is peripheral memory, and the *DMA\_CHn\_CTRL*.dstinc field is ignored when the DMA destination is peripheral memory.

*Table 9-2: DMA Source and Destination by Peripheral*

<i>DMA_CHn_CTRL</i> .request	Peripheral	DMA Source	DMA Destination
0	Memory-to-Memory	<i>DMA_CHn_SRC</i>	<i>DMA_CHn_DST</i>
1	SPI1	SPI1 Receive FIFO	<i>DMA_CHn_DST</i>
2-3	Reserved	-	-
4	UART0	UART0 Receive FIFO	<i>DMA_CHn_DST</i>
5	UART1	UART1 Receive FIFO	<i>DMA_CHn_DST</i>
6	Reserved	-	-
7	I <sup>2</sup> C 0	I <sup>2</sup> C0 Receive FIFO	<i>DMA_CHn_DST</i>
8	I <sup>2</sup> C 1	I <sup>2</sup> C1 Receive FIFO	<i>DMA_CHn_DST</i>
9	ADC	ADC Data Register	<i>DMA_CHn_DST</i>
10	I <sup>2</sup> C 2	I <sup>2</sup> C2 Receive FIFO	<i>DMA_CHn_DST</i>
11-12	Reserved	-	-
13	PCIF RX	PCIF Receive FIFO	<i>DMA_CHn_DST</i>
14	UART 2	UART2 Receive FIFO	<i>DMA_CHn_DST</i>
15	SPI 0	SPI0 Receive FIFO	<i>DMA_CHn_DST</i>
16-27	Reserved	-	-
28	LPUART0 (UART 3)	UART3 Receive FIFO	<i>DMA_CHn_DST</i>
29	Reserved	-	-
30	I <sup>2</sup> S	I <sup>2</sup> S Data Register	<i>DMA_CHn_DST</i>
31-32	Reserved	-	-
33	SPI 1	<i>DMA_CHn_SRC</i>	SPI1 Transmit FIFO

<i>DMA_CHn_CTRL.request</i>	Peripheral	DMA Source	DMA Destination
34-35	Reserved	-	-
36	UART 0	<i>DMA_CHn_SRC</i>	UART0 Transmit FIFO
37	UART 1	<i>DMA_CHn_SRC</i>	UART1 Transmit FIFO
38	Reserved	-	-
39	I <sup>2</sup> C 0	<i>DMA_CHn_SRC</i>	I <sup>2</sup> C0 Transmit FIFO
40	I <sup>2</sup> C 1	<i>DMA_CHn_SRC</i>	I <sup>2</sup> C1 Transmit FIFO
41	Reserved	-	-
42	I <sup>2</sup> C 2	<i>DMA_CHn_SRC</i>	I <sup>2</sup> C2 Transmit FIFO
43	Reserved	-	-
44	CRC	<i>DMA_CHn_SRC</i>	CRC Data Register
45	PCIF TX	<i>DMA_CHn_SRC</i>	PCIF Data Register
46	UART2	<i>DMA_CHn_SRC</i>	UART2 Transmit FIFO
47	SPI0	<i>DMA_CHn_SRC</i>	SPI0 Transmit FIFO
48-59	Reserved	-	-
60	LPUART 0 (UART 3)	<i>DMA_CHn_SRC</i>	UART3 Transmit FIFO
61	Reserved	-	-
62	I <sup>2</sup> S	<i>DMA_CHn_SRC</i>	I <sup>2</sup> S Data Register
63	Reserved	-	-

### 9.2.3 Data Movement from Source to DMA

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

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

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

#### 9.2.4 Data Movement from DMA to Destination

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

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

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

## 9.3 Usage

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

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

## 9.4 Count-To-Zero (CTZ) Condition

When an AHB channel burst completes, a CTZ condition exists if *DMA\_CHn\_CNT* is decremented to 0.

At this point, there are two responses are possible depending on the value of the *DMA\_CHn\_CTRL.rlden*:

1. If *DMA\_CHn\_CTRL.rlden* = 1, then the *DMA\_CHn\_SRC*, *DMA\_CHn\_DST*, and *DMA\_CHn\_CNT* registers are loaded from the reload registers, and the channel remains active and continues operating using the newly-loaded address/count values and the previously programmed configuration values.
2. If *DMA\_CHn\_CTRL.rlden* = 0, then the channel is disabled, and *DMA\_CHn\_STATUS.status* is cleared.

## 9.5 Chaining Buffers

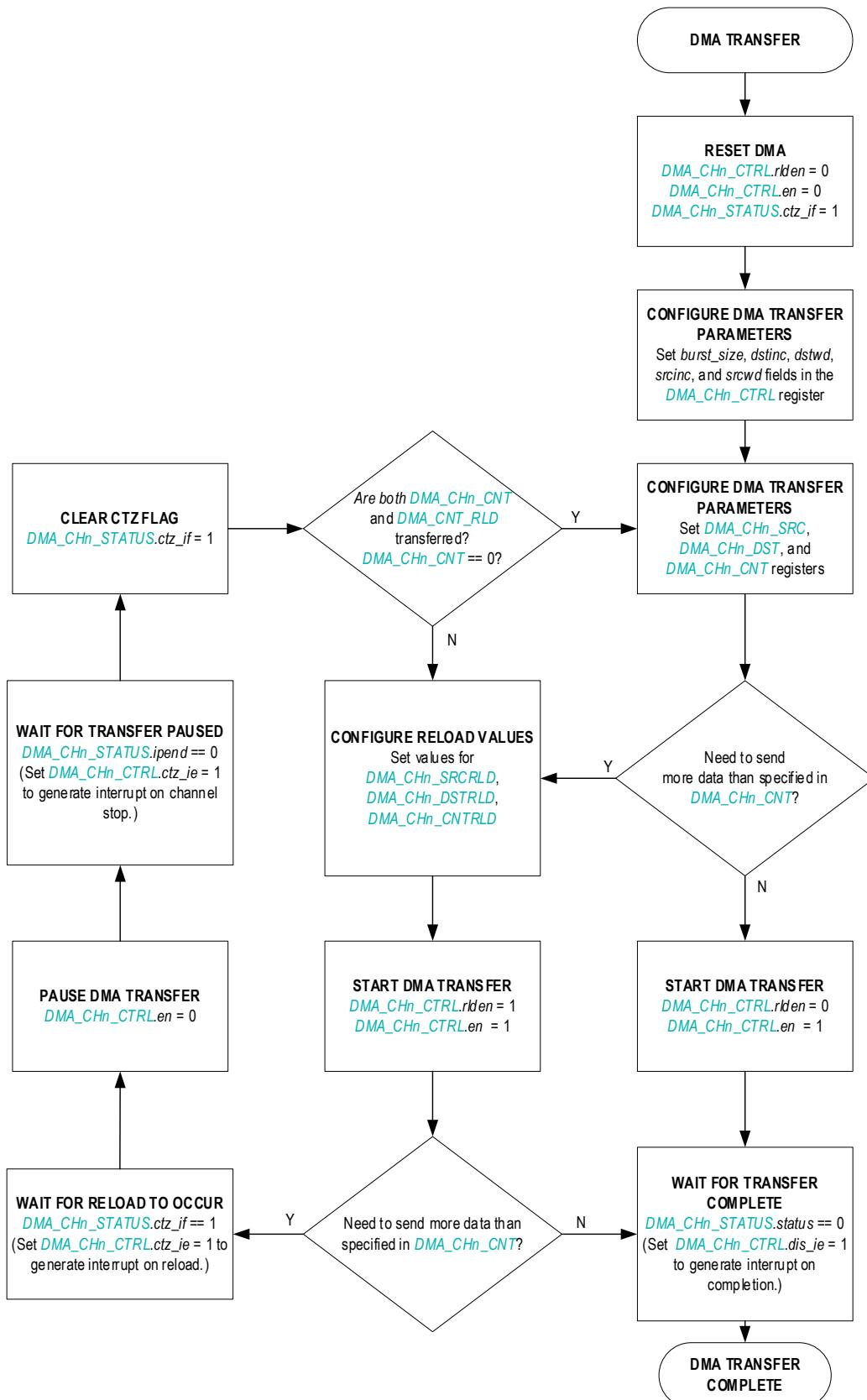
Chaining buffers reduce the DMA interrupt response time and allow the DMA to service requests without intermediate processing from the CPU. [Figure 9-1: DMA Block-Chaining Flowchart](#) shows the procedure for generating a DMA transfer using one or more chain buffers.

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

- *DMA\_CHn\_SRC*
- *DMA\_CHn\_DST*
- *DMA\_CHn\_CNT*
- *DMA\_CHn\_SRCRLD*
- *DMA\_CHn\_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/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 9-1: DMA Block-Chaining Flowchart



## 9.6 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).

## 9.7 Channel Timeout Detect

Each channel can optionally generate an interrupt when its associated peripheral does not request a transfer in a user-configurable period. When the timeout start conditions are met, an internal 10-bit counter begins incrementing at a frequency determined by the AHB clock, the *DMA\_CHn\_CTRL.to\_clkdiv* field, and the *DMA\_CHn\_CTRL.to\_per* field. See *Table 9-5* for details. A channel timeout event is generated if the timer is not reset by one of the events listed below before the timeout period expires.

*Table 9-5: DMA Channel Timeout Configuration*

<i>DMA_CHn_CTRL.to_clkdiv</i>	Timeout Period (μs)
0	Channel timeout disabled.
1	$\frac{2^8 \times [\text{Value from } \textit{DMA\_CHn\_CTRL.to\_per}]}{f_{HCLK}}$
2	$\frac{2^{16} \times [\text{Value from } \textit{DMA\_CHn\_CTRL.to\_per}]}{f_{HCLK}}$
3	$\frac{2^{24} \times [\text{Value from } \textit{DMA\_CHn\_CTRL.to\_per}]}{f_{HCLK}}$

*DMA\_CHn\_CTRL.to\_wait* controls the start of the timeout period as follows:

- If *DMA\_CHn\_CTRL.to\_wait* = 0, the timer begins immediately counting after *DMA\_CHn\_CTRL.to\_per* is configured to a value other than 0, and the channel is enabled.
- If *DMA\_CHn\_CTRL.to\_wait* = 1, the timer begins counting when the first DMA request is received from the peripheral.

The timer is reset whenever:

- The DMA request line programmed for the channel is activated.
- The channel is disabled for any reason (*DMA\_CHn\_STATUS.status* = 0).

If the timeout timer period expires, hardware sets *DMA\_CHn\_STATUS.to\_if* = 1 to indicate a channel timeout event has occurred. A channel timeout does not disable the DMA channel.

## 9.8 Memory-to-Memory DMA

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

## 9.9 Registers

See *Table 2-4* for the base address of this peripheral/module. See *Table 2-1* 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 9-6: DMA Register Summary*

Offset	Register	Description
[0x0000]	<i>DMA_INTEN</i>	DMA Channel Interrupt Enable
[0x0004]	<i>DMA_INTFL</i>	DMA Interrupt Status register

### 9.9.1 Register Details

*Table 9-7: DMA Interrupt Enable Register*

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

*Table 9-8: DMA Interrupt Flag Register*

DMA Interrupt Flag				DMA_INTFL	[0x0004]
Bits	Field	Access	Reset	Description	
31:0	ch<n>	RO	0	<b>DMA Channel <i>n</i> Interrupt Flag</b> Each bit in this field represents an interrupt for the corresponding channel interrupt <n>. To clear an interrupt, clear the corresponding active interrupt bit in the <i>DMA_CHn_STATUS</i> register. An interrupt bit in this field is set if the corresponding interrupt enable field is set in the <i>DMA_INTEN</i> register and a channel's interrupt occurs. Register bits associated with unimplemented channels should be ignored. 0: No interrupt 1: Interrupt pending	

## 9.10 DMA Channel Registers

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

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

### 9.10.1 DMA Channel Register Details

See *Table 2-4* for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own independent set of the registers shown in *Table 9-10*. Register names for a specific instance are defined by replacing "n" with the instance number. As an example, a register PERIPHERALn\_CTRL resolves to PERIPHERAL0\_CTRL and PERIPHERAL1\_CTRL for instances 0 and 1, respectively.

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

*Table 9-10: DMA Channel Registers Summary*

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

*Table 9-11: DMA\_CH n Control Register*

DMA Channel n Control			DMA_CHn_CTRL		[0x0100]
Bits	Field	Access	Reset	Description	
31	ctz_ie	R/W	0	<b>CTZ Interrupt Enable</b> 0: Disabled 1: Enabled. <i>DMA_INFL.ch&lt;n&gt;</i> is set to 1 whenever a CTZ event occurs.	
30	dis_ie	R/W	0	<b>Channel Disable Interrupt Enable</b> 0: Disabled 1: Enabled. <i>DMA_INFL.ch&lt;n&gt;</i> bit is set to 1 whenever <i>DMA_CHn_STATUS.status</i> changes from 1 to 0.	
29	-	RO	0	<b>Reserved</b>	

DMA Channel n Control				DMA_CHn_CTRL	[0x0100]
Bits	Field	Access	Reset	Description	
28:24	burst_size	R/W	0	<b>Burst Size</b> The number of bytes transferred into and out of the DMA FIFO in a single burst. 0b00000: 1 byte 0b00001: 2 bytes 0b00010: 3 bytes ... 0b11111: 32 bytes	
23	-	RO	0	<b>Reserved</b>	
22	dstinc	R/W	0	<b>Destination Increment Enable</b> This bit enables the automatic increment of the <a href="#">DMA_CHn_DST</a> 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	<b>Destination Width</b> Indicates the width of each AHB transaction to the destination peripheral or memory (the actual width might be less than this if there are insufficient bytes in the DMA FIFO for the full width). 0: One byte 1: Two bytes 2: Four bytes 3: Reserved	
19	-	RO	0	<b>Reserved</b>	
18	srcinc	R/W	0	<b>Source Increment on AHB Transaction Enable</b> This bit enables the automatic increment of the <a href="#">DMA_CHn_SRC</a> 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	<b>Source Width</b> This field indicates the width of each AHB transaction from the source peripheral or memory. The actual width might be less than this if the <a href="#">DMA_CHn_CNT</a> register indicates a smaller value. 0: One byte 1: Two bytes 2: Four bytes 3: Reserved	
15:14	to_clkdiv	R/W	0	<b>Timeout Timer Clock Pre-Scale Select</b> This field selects the pre-scale divider for the timer clock input. 0: Timer disabled. 1: $\frac{f_{HCLK}}{28}$ 2: $\frac{f_{HCLK}}{216}$ 3: $\frac{f_{HCLK}}{224}$	

DMA Channel n Control				DMA_CHn_CTRL	[0x0100]
Bits	Field	Access	Reset	Description	
13:11	to_per	R/W	0	<b>Timeout Period Select</b> 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 to 4 1: 7 to 8 2: 15 to 16 3: 31 to 32 4: 63 to 64 5: 127 to 128 6: 255 to 256 7: 511 to 512	
10	to_wait	R/W	0	<b>Request DMA Timeout Timer Wait Enable</b> This field controls when the timeout timer starts, either immediately when the timeout timer is enabled or after the first DMA transaction occurs. 0: Start the timer immediately when enabled. 1: Delay the timer's start until after the first DMA transaction occurs.	
9:4	request	R/W	0	<b>Request Select</b> Selects the source and destination for the transfer as shown in <a href="#">Source and Destination Addressing</a> .	
3:2	pri	R/W	0	<b>Channel Priority</b> This field sets the priority of the channel relative to other channels of the DMA peripheral. Channels of the same priority are serviced in a round-robin fashion. 0: High 1: Medium-high 2: Medium-low 3: Low	
1	rlden	R/W	0	<b>Reload Enable</b> Setting this bit to 1 allows reloading the <a href="#">DMA_CHn_SRC</a> , <a href="#">DMA_CHn_DST</a> , and <a href="#">DMA_CHn_CNT</a> registers upon a CTZ. When this bit is set to 0, and a CTZ occurs, the channel is disabled, and the <a href="#">DMA_CHn_STATUS.status</a> bit is set to 0. 0: The channel is disabled when a CTZ occurs, and the <a href="#">DMA_CHn_STATUS.status</a> is set to 0. 1: Automatically reload the <a href="#">DMA_CHn_SRC</a> , <a href="#">DMA_CHn_DST</a> , and <a href="#">DMA_CHn_CNT</a> registers on a CTZ.	
0	en	R/W	0	<b>Channel Enable</b> This bit is automatically cleared when <a href="#">DMA_CHn_STATUS.status</a> changes from 1 to 0. 0: Disabled 1: Enabled	

Table 9-12: DMA Status Register

DMA Channel n Status				DMA_CHn_STATUS	[0x0104]
Bits	Field	Access	Reset	Description	
31:7	-	RO	0	<b>Reserved</b>	

DMA Channel n Status				DMA_CHn_STATUS	[0x0104]
Bits	Field	Access	Reset	Description	
6	to_if	R/W1C	0	<b>Timeout Interrupt Flag</b> Timeout. Write 1 to clear. 0: No time out. 1: A channel time out has occurred	
5	-	RO	0	<b>Reserved</b>	
4	bus_err	R/W1C	0	<b>Bus Error</b> 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	<b>Reload Interrupt Flag</b> Reload. Write 1 to clear. 0: Reload has not occurred. 1: Reload occurred.	
2	ctz_if	R/W1C	0	<b>CTZ Interrupt Flag</b> Write 1 to clear. 0: CTZ has not occurred. 1: CTZ has occurred.	
1	ipend	RO	0	<b>Channel Interrupt Pending</b> 0: No interrupt 1: Interrupt pending	
0	status	RO	0	<b>Channel Status</b> This bit indicates when it is safe to change the configuration, address, and count registers for the channel. Whenever this bit is cleared by hardware, the <i>DMA_CHn_CTRL.en</i> bit is also cleared. 0: Disabled 1: Enabled.	

Table 9-13: DMA Channel n Source Register

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

Table 9-14: DMA Channel n Destination Register

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

Table 9-15: DMA Channel n Count Register

DMA Channel n Count			DMA_CHn_CNT		[0x0110]
Bits	Field	Access	Reset	Description	
31:24	-	RO	0	<b>Reserved</b>	
23:0	cnt	R/W	0	<b>DMA Counter</b> 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. Suppose a CTZ condition occurs while <i>DMA_CHn_CTRL.rlden</i> = 1, then this register is reloaded with the contents of the <i>DMA_CHn_CNTRLD.cnt_rld</i> field.	

Table 9-16: DMA Channel n Source Reload Register

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

Table 9-17: DMA Channel n Destination Reload Register

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

Table 9-18: DMA Channel n Count Reload Register

DMA Channel n Count Reload			DMA_CHn_CNTRLD		[0x011C]
Bits	Field	Access	Reset	Description	
31:24	-	RO	0	<b>Reserved</b>	
23:0	cnt	R/W	0	<b>Count Reload Value</b> If <a href="#">DMA_CHn_CTRL.rlden</a> = 1, then the value of this register is loaded into <a href="#">DMA_CHn_CNT</a> upon a CTZ condition.	

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

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

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

### 10.1 Features

- Maximum 8MHz ADC clock rate
- Two reference source options
  - ◆ An internal 1.22V bandgap
  - ◆  $\frac{V_{DDA}}{2}$  supply
- 8 external analog inputs configurable as 4 two-input comparators
- 8 internal power supply monitor inputs
- Fixed 10-bit word conversion time of 1024 ADC clock cycles
- Programmable out-of-range (limit) detection
- Interrupt generation for limit detection, conversion start, conversion complete, and internal reference powered on
- Serial ADC data measurements
- ADC conversion 10-bit output either MSB or LSB aligned

### 10.2 Instances

*Table 10-1: MAX78000 ADC Input Pins for the 81-CTBGA Package*

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



## 10.3 Architecture

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

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

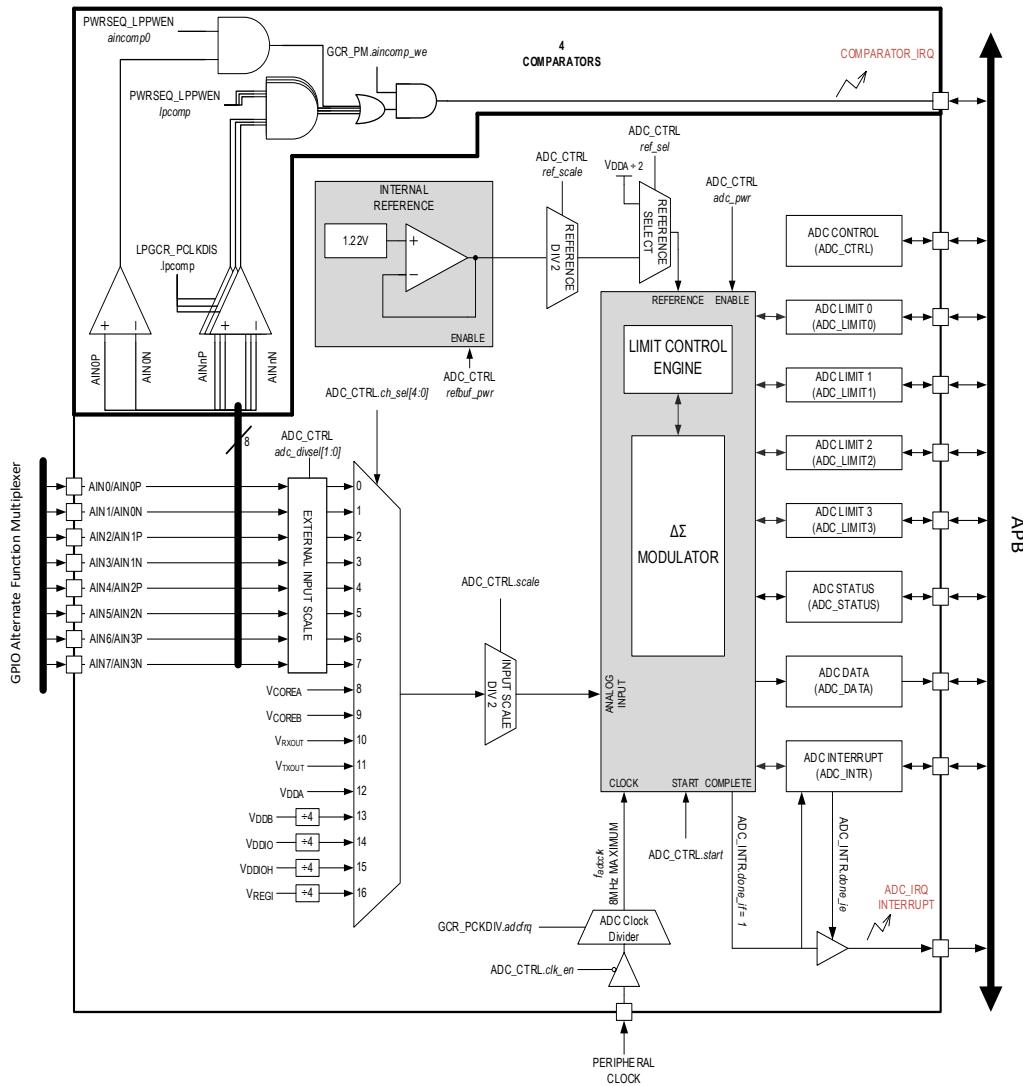
$$t_{adc\_sample} = 1024 \times \left( \frac{1}{f_{adcclk}} \right)$$

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

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

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

*Figure 10-1: Analog to Digital Converter Block Diagram*



Preliminary Draft 05/21/2021

## 10.4 Clock Configuration

The ADC clock,  $adcclk$ , is controlled by the **GCR\_PCLKDIV.adcfreq** register field. Configure this field to achieve the target ADC sample frequency. The maximum clock frequency supported by the ADC is 8MHz. The divisor selection, **GCR\_PCLKDIV.adcfreq**, for the ADC depends on the peripheral clock. *Equation 10-2* shows the calculation for the ADC clock.

*Equation 10-2: ADC Clock Frequency*

$$f_{adcclk} = \frac{f_{PCLK}}{GCR\_PCKDIV.adcfreq}$$

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

*Table 10-2: MAX78000 ADC Clock Frequency and ADC Conversion Time with the System Clock set to the IPO*

<b>GCR_PCLKDIV.adcfreq</b>	<b>ADC Clock Frequency (Hz) <math>f_{adcclk}</math></b>	<b>10-Bit Word Conversion Time (<math>\mu\text{s}</math>) <math>t_{adc\_sample}</math></b>
0 – 7	Invalid	Invalid
8	6,250,000	164
9	5,555,555	184
10	5,000,000	205
11	4,545,454	225
12	4,166,666	246
13	3,846,153	266
14	3,571,428	287
15	3,333,333	307

## 10.5 Power-Up Sequence

Complete the following steps to configure the ADC:

1. Disable the ADC clock by setting the *ADC\_CTRL.clk\_en* field to 0.
2. Set the ADC clock ( $f_{adcclk}$ ) using the *GCR\_PCLKDIV.adcfrq* field. See *Clock Configuration* for details.
3. Enable the ADC clock by setting the *ADC\_CTRL.clk\_en* field to 1
4. Clear the ADC reference ready interrupt flag by writing a 1 to *ADC\_INTR.ref\_ready\_if*.
5. Optionally enable the ADC reference ready interrupt by setting the *ADC\_INTR.ref\_ready\_ie* field to 1 and enable the ADC interrupt handler (ADC\_IRQn).
6. Select one of the following ADC reference sources:
  - a. Internal 1.22V bandgap reference (*ADC\_CTRL.ref\_sel* = 0).
  - b.  $\frac{V_{DDA}}{2}$  reference (*ADC\_CTRL.ref\_sel* = 1).
7. Complete the following steps to enable power to the ADC and optionally the internal ADC reference:
  - a. Set *ADC\_CTRL.pwr* to 1 to turn on the ADC.
  - b. Set *ADC\_CTRL.refbuf\_pwr* to 1 to turn on the internal reference buffer if using the internal reference.
  - c. Wait until hardware sets the *ADC\_INTR.ref\_ready\_if* field to 1 indicating the internal reference is fully powered on and ready.
  - d. Clear the ADC reference ready interrupt flag by writing 1 to *ADC\_INTR.ref\_ready\_if*.
  - e. Optionally disable the ADC reference ready interrupt by clearing the *ADC\_INTR.ref\_ready\_ie* field to 0.

## 10.6 Conversion

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

1. Select the ADC input channel for the conversion by setting the *ADC\_CTRL.ch\_sel* field. See *ADC Channel Select* for details.
2. Optionally set input and reference scaling. See *Scale Limitations for All Other Input Channels* for details on each input channel's scale requirements.
3. Set the data alignment for the conversion output data using the *ADC\_CTRL.data\_align* field.
  - a. 0 for LSB alignment or 1 for MSB alignment. See *Table 10-3* for alignment details of the *ADC\_DATA* register.
4. Clear the ADC done interrupt flag by writing 1 to the *ADC\_INTR.done\_if* field.
5. Optionally enable the ADC done interrupt (*ADC\_INTR.done\_ie* = 1) and enable the ADC interrupt vector (ADC\_IRQn).
6. Start the ADC conversion by setting the *ADC\_CTRL.start* field to 1.
7. Poll the *ADC\_INTR.done\_if* flag until it reads 1 or wait for the ADC interrupt to occur if enabled.
8. Read the data from the *ADC\_DATA.data* field and clear the ADC done interrupt flag by writing 1 to the *ADC\_INTR.done\_if* field.

### 10.6.1 Data Conversion Output Alignment

The ADC outputs a total of 10-bits per conversion and stores the data in the DATA register LSB justified by default. *Table 10-3* shows the ADC data alignment based on the value of the *ADC\_CTRL.data\_align* bit.

Table 10-3: ADC Data Register Alignment Options

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

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

## 10.6.2 Data Conversion Value Equations

Use the following equations to calculate the ADC data value for a conversion for the selected channel. If using the internal reference,  $V_{REF} = 1.22V$ ; otherwise  $V_{REF} = V_{DDA}$ .

*Equation 10-3: ADC Data Calculation for Input Signal  $ADC\_CTRL.ch\_sel = 0$  through 7 (AIN0 - AIN7)*

$$ADC\_DATA = \text{round} \left\{ \left( \frac{\left( \frac{\text{Input Signal}}{2^{scale}} * (adc\_divsel + 1) \right)}{\left( \frac{V_{REF}}{2^{ref\_scale}} \right)} \right) \times (2^{10} - 1) \right\}$$

*Note: Must satisfy Equation 10-6.*

*Equation 10-4: ADC Data Equation for Input Signal  $ADC\_CTRL.ch\_sel = 8$  through 12 ( $V_{COREA}$ ,  $V_{COREB}$ ,  $V_{RXOUT}$ ,  $V_{TXOUT}$ ,  $V_{DDA}$ )*

$$ADC\_DATA = \text{round} \left\{ \left( \frac{\left( \frac{\text{Input Signal}}{2^{scale}} \right)}{\left( \frac{V_{REF}}{2^{ref\_scale}} \right)} \right) \times (2^{10} - 1) \right\}$$

*Note: See Table 10-4 for limitations.*

*Equation 10-5: ADC Data Calculation Input Signal  $ADC\_CTRL.ch\_sel = 14$  through 16 ( $V_{DDIO}$ ,  $V_{DDIOH}$ ,  $V_{REGI}$ )*

$$ADC\_DATA = \text{round} \left\{ \left( \frac{\left( \frac{\text{Input Signal}}{4} \right)}{\left( \frac{V_{REF}}{2^{ref\_scale}} \right)} \right) \times (2^{10} - 1) \right\}$$

*Note: See Table 10-4 for limitations.*

## 10.7 Reference Scaling and Input Scaling

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

### 10.7.1 AIN0 – AIN7 Scale Limitations

The external inputs, AIN0 through AIN7, support scaling of the input by 50%, the reference by 50%, or both by 50%. Also, the scaling can further be modified by additional factors of 2, 3, or 4 as defined by *ADC\_CTRL.adc\_divsel*. The scale settings for the given input signal and reference must satisfy *Equation 10-6* to be valid:

*Equation 10-6: Input and Reference Scale Requirements Equation*

$$\frac{AIN_n}{2^{scale}} < \frac{V_{REF}}{2^{ref\_scale}}$$

### 10.7.2 Scale Limitations for All Other Input Channels

For the remaining internal input channels, the scale settings must either both be disabled, or both be enabled as shown in *Table 10-4*.

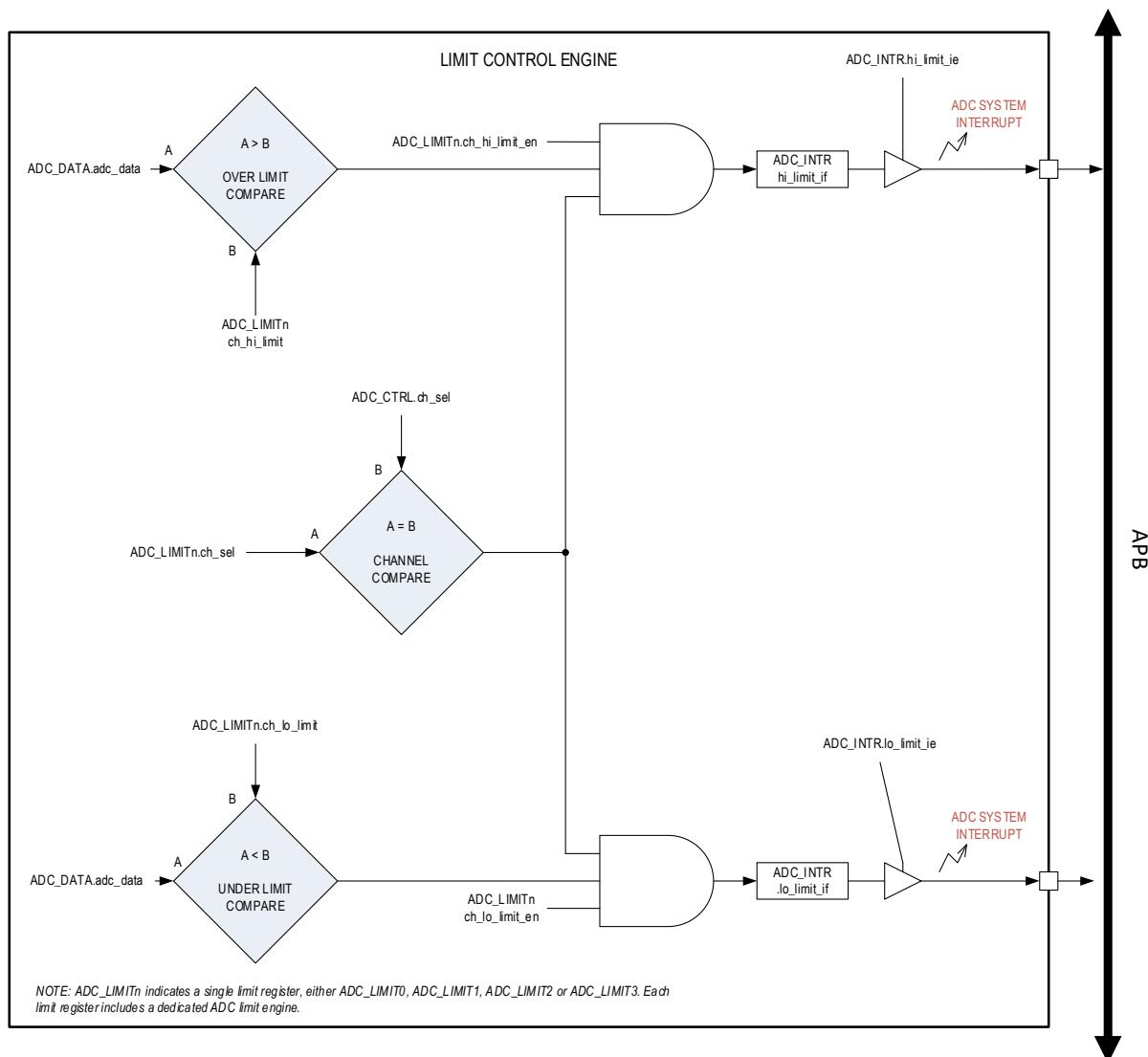
*Table 10-4: Input and Reference Scale Support by ADC Input Channel*

ADC Channel	ADC Input Signal	<i>ADC_CTRL.scale</i>	<i>ADC_CTRL.ref_scale</i>
8	<i>V<sub>COREA</sub></i>	0	0
		1	1
9	<i>V<sub>COREB</sub></i>	0	0
		1	1
10	<i>V<sub>RXOUT</sub></i>	0	0
		1	1
11	<i>V<sub>TXOUT</sub></i>	0	0
		1	1
12	<i>V<sub>DDA</sub></i>	0	0
		1	1
14	<i>V<sub>DDIO</sub></i> 4	0	0
		1	1
15	<i>V<sub>DDIOH</sub></i> 4	0	0
		1	1
16	<i>V<sub>REGI</sub></i> 4	0	0
		1	1

## 10.8 Data Limits and Out of Range Interrupts

Channel limits are implemented to minimize power consumption for power supply monitoring. The ADC includes four limit registers, *ADC\_LIMIT0* to *ADC\_LIMIT3*, that can be used to set a high limit, low limit, and the ADC channel number to apply the limits against. A block diagram of the limit engine for each of the four limit registers is shown in *Figure 10-2*.

*Figure 10-2: ADC Limit Engine*



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

Complete the following steps to enable a high and low limit for an ADC input channel using the [ADC\\_LIMIT0](#) register. Perform these steps after the ADC is configured for measurement, and the configuration is identical for all four limit registers except for the limit register name:

1. Verify the ADC is not actively taking a measurement by checking [ADC\\_STATUS.active](#) until it reads 0.
2. Set [ADC\\_LIMIT0.ch\\_sel](#) field to the selected channel for the high and low limit.
3. Set the high limit, [ADC\\_LIMIT0.ch\\_hi\\_limit](#), to the selected 10-bit trip point. When enabled, an ADC measurement greater than this field on the channel selected ([ADC\\_LIMIT0.ch\\_sel](#)) generates an ADC interrupt.
4. Set the low limit, [ADC\\_LIMIT0.ch\\_lo\\_limit](#), to the selected 10-bit low trip point. When enabled, an ADC measurement lower than this field on the channel selected ([ADC\\_LIMIT0.ch\\_sel](#)) generates an ADC interrupt.
5. Enable the high limit, the low limit, or both interrupt signals by writing a 1 to [ADC\\_LIMIT0.ch\\_high\\_limit\\_en](#), [ADC\\_LIMIT0.ch\\_low\\_limit\\_en](#), or both. Note: Each limit register is independently enabled for high- and low-limit interrupts.
6. Clear the ADC interrupt high and low interrupt flags by writing 1 to [ADC\\_INTR.hi\\_limit\\_if](#) and [ADC\\_INTR.lo\\_limit\\_if](#).
7. Enable the high, low, or both interrupts for the ADC by setting [ADC\\_INTR.hi\\_limit\\_if](#) to 1, [ADC\\_INTR.lo\\_limit\\_ie](#) to 1, or both.
8. If an ADC conversion occurs that is above or below the enabled limits, an ADC\_IRQ is generated with the [ADC\\_LIMIT0.adc\\_high\\_limit\\_if](#), [ADC\\_LIMIT0.adc\\_low\\_limit\\_if](#), or both set to 1. The [ADC\\_CTRL.ch\\_sel](#) value indicates the channel that caused the interrupt, and the value of the ADC conversion that is out of bounds is in the [ADC\\_DATA.data](#) field.

## 10.9 Power-Down Sequence

Complete the following steps to power-down the ADC:

1. Set [ADC\\_CTRL.pwr](#) to 0, disabling the ADC converter power.
2. [ADC\\_CTRL.refbuf\\_pwr](#) to 0, disabling the internal reference buffer power.
3. Set [ADC\\_CTRL.clk\\_en](#) to 0, disabling the ADC internal clock.

## 10.10 Comparator Operation

### 10.10.1 Comparator 0 Usage

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

### 10.10.2 Low-Power Comparators 1, 2 and 3 Usage

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

### 10.10.3 Using Comparator 0 as a Wakeup Source

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

1. Enable comparator wakeup events by setting *GCR\_PM.aincomp\_we* to 1.
2. Enable comparator 0 as a wakeup source by setting *PWRSEQ\_LPPWST.comp0* to 1.
3. If desired, provide an IRQ handler for the comparators (*LPCOMP\_IRQn*).

After the device exists a low-power mode, determine if the wakeup event was the result of Comparator 0 by checking the *PWRSEQ\_LPPWST.comp0* and the *MCR\_CMPO\_CTRL.if*. Wakeup events generated by Comparator 0 from *STANDBY* and *BACKUP* mode result in the *PWRSEQ\_LPPWST.comp0* bit being set. Write 1 to clear the *PWRSEQ\_LPPWST.comp0* bit and the *MCR\_CMPO\_CTRL.if* bit.

### 10.10.4 Low-Power Comparators 1, 2 and 3 Wakeup

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

Enable wakeup events from the low-power comparators by setting the *GCR\_PM.aincomp\_we* field to 1. If a comparator event occurs and wakes the device from a low-power operating mode, the *PWRSEQ\_LPPWST.comp* field is set to 1. Clear the comparator wakeup status flag by writing 1 to *PWRSEQ\_LPPWST.comp*.

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

## 10.11 ADC Registers

See *Table 2-4* for the base address of this peripheral/module. See *Table 2-1* 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 10-5. ADC Registers Summary*

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

### 10.11.1 ADC Register Details

*Table 10-6: ADC Control Register*

ADC Control		ADC_CTRL			[0x0000]
Bits	Field	Access	Reset	Description	
31:21	-	RO	0x050	Reserved	

ADC Control			ADC_CTRL		[0x0000]																																																									
Bits	Field	Access	Reset	Description																																																										
20	data_align	R/W	0	<b>ADC Data Alignment</b> Selects the alignment of the 16-bit data conversion stored in the DATA register. 0: Data is LSB justified in 16-bit DATA register. DATA[15:10] = 0. 1: Data is MSB justified in 16-bit DATA register. DATA[5:0] = 0.																																																										
19	-	RO	0	<b>Reserved</b>																																																										
18:17	adc_divsel	R/W	0	<b>External Input Scale</b> Scales the external inputs AIN0-AIN7. All eight of external inputs are scaled by the same value 0: No scaling. 1: Divide by 2 2: Divide by 3 3: Divide by 4																																																										
16:12	ch_sel	R/W	0	<b>ADC Channel Select</b> Selects the active channel for the next ADC conversion. <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>ch_sel</th> <th>ADC Input Channel</th> <th>Input</th> </tr> </thead> <tbody> <tr><td>0x00</td><td>0</td><td>AIN0</td></tr> <tr><td>0x01</td><td>1</td><td>AIN1</td></tr> <tr><td>0x02</td><td>2</td><td>AIN2</td></tr> <tr><td>0x03</td><td>3</td><td>AIN3</td></tr> <tr><td>0x04</td><td>4</td><td>AIN4</td></tr> <tr><td>0x05</td><td>5</td><td>AIN5</td></tr> <tr><td>0x06</td><td>6</td><td>AIN6</td></tr> <tr><td>0x07</td><td>7</td><td>AIN7</td></tr> <tr><td>0x08</td><td>8</td><td><math>V_{COREA}</math></td></tr> <tr><td>0x09</td><td>9</td><td><math>V_{COREB}</math></td></tr> <tr><td>0x0A</td><td>10</td><td><math>V_{RXOUT}</math></td></tr> <tr><td>0x0B</td><td>11</td><td><math>V_{TXOUT}</math></td></tr> <tr><td>0x0C</td><td>12</td><td><math>V_{DDA}</math></td></tr> <tr><td>0x0D</td><td>13</td><td><math>\frac{V_{DDB}}{4}</math></td></tr> <tr><td>0x0E</td><td>14</td><td><math>\frac{V_{DDIO}}{4}</math></td></tr> <tr><td>0x0F</td><td>15</td><td><math>\frac{V_{DDIOH}}{4}</math></td></tr> <tr><td>0x10</td><td>16</td><td><math>\frac{V_{REGI}}{4}</math></td></tr> <tr><td>0x11 – 0x1F</td><td>Reserved</td><td>Reserved</td></tr> </tbody> </table>	ch_sel	ADC Input Channel	Input	0x00	0	AIN0	0x01	1	AIN1	0x02	2	AIN2	0x03	3	AIN3	0x04	4	AIN4	0x05	5	AIN5	0x06	6	AIN6	0x07	7	AIN7	0x08	8	$V_{COREA}$	0x09	9	$V_{COREB}$	0x0A	10	$V_{RXOUT}$	0x0B	11	$V_{TXOUT}$	0x0C	12	$V_{DDA}$	0x0D	13	$\frac{V_{DDB}}{4}$	0x0E	14	$\frac{V_{DDIO}}{4}$	0x0F	15	$\frac{V_{DDIOH}}{4}$	0x10	16	$\frac{V_{REGI}}{4}$	0x11 – 0x1F	Reserved	Reserved	
ch_sel	ADC Input Channel	Input																																																												
0x00	0	AIN0																																																												
0x01	1	AIN1																																																												
0x02	2	AIN2																																																												
0x03	3	AIN3																																																												
0x04	4	AIN4																																																												
0x05	5	AIN5																																																												
0x06	6	AIN6																																																												
0x07	7	AIN7																																																												
0x08	8	$V_{COREA}$																																																												
0x09	9	$V_{COREB}$																																																												
0x0A	10	$V_{RXOUT}$																																																												
0x0B	11	$V_{TXOUT}$																																																												
0x0C	12	$V_{DDA}$																																																												
0x0D	13	$\frac{V_{DDB}}{4}$																																																												
0x0E	14	$\frac{V_{DDIO}}{4}$																																																												
0x0F	15	$\frac{V_{DDIOH}}{4}$																																																												
0x10	16	$\frac{V_{REGI}}{4}$																																																												
0x11 – 0x1F	Reserved	Reserved																																																												
11	clk_en	R/W	0	<b>ADC Clock Enable</b> 0: Disabled 1: Enabled																																																										
10	-	RO	0	<b>Reserved</b>																																																										
9	scale	R/W	0	<b>ADC Input Scale</b> Scales ADC input by 50 percent. 0: ADC input is not scaled 1: ADC input is scaled by $\frac{1}{2}$ . <i>Note: See <a href="#">Data Conversion Output Alignment</a> for valid settings for each ADC input.</i>																																																										
8	ref_scale	R/W	0	<b>Reference Scale</b> Scales the internal bandgap reference by 50 percent. 0: Internal bandgap reference is not scaled. 1: Internal bandgap reference is scaled by $\frac{1}{2}$ . <i>Note: See <a href="#">Data Conversion Output Alignment</a> for valid settings for each ADC input</i>																																																										

ADC Control			ADC_CTRL		[0x0000]
Bits	Field	Access	Reset	Description	
7:5	-	RO	0	<b>Reserved</b>	
4	ref_sel	R/W	0	<b>ADC Reference Select</b> 0: Internal bandgap reference is used for the ADC reference 1: $V_{DDA} \div 2$ is used for the ADC reference	
3	refbuf_pwr	R/W	0	<b>Reference Buffer Power Enable</b> 0: Disabled 1: Enabled	
2	-	RO	0	<b>Reserved</b>	
1	pwr	R/W	0	<b>ADC Power Enable</b> Set this field to 1 to enable power to the ADC peripheral. 0: Disabled 1: Enabled	
0	start	R/W	0	<b>Start ADC Conversion</b> Write this bit to 1 to start an ADC conversion. When the conversion is complete, the hardware automatically sets this bit to 0 indicating the conversion is complete. 0: ADC inactive or data conversion complete. 1: Start ADC conversion and remains set until complete.	

Table 10-7: ADC Status Register

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

Table 10-8: ADC Data Register

ADC Data			ADC_DATA		[0x0008]
Bits	Field	Access	Reset	Description	
15:0	data	RO	0	<b>ADC Data</b> This field holds the ADC conversion output data. See <a href="#">Table 10-3</a> for details.	

Table 10-9: ADC Interrupt Control Register

ADC Interrupt Control			ADC_INTR		[0x000C]
Bits	Field	Access	Reset	Description	
31:23	-	RO	0	<b>Reserved</b>	
22	pending	RO	0	<b>ADC Interrupt Pending</b> 0: No ADC interrupt pending. 1: At least one ADC interrupt is pending, and the corresponding interrupt enable bit is set.	
21	-	RO	0	<b>Reserved</b>	

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

Table 10-10: ADC Limit 0 to 3 Registers

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

<b>ADC Limit 0</b>			<b>ADC_LIMIT0</b>	[0x0010]
<b>ADC Limit 1</b>			<b>ADC_LIMIT1</b>	[0x0014]
<b>ADC Limit 2</b>			<b>ADC_LIMIT2</b>	[0x0018]
<b>ADC Limit 3</b>			<b>ADC_LIMIT3</b>	[0x001C]
Bits	Field	Access	Reset	Description
27:24	ch_sel	R/W	0	<b>ADC Channel for Limit Monitoring</b> Sets the ADC input channel for high- and low-limit thresholds. See <a href="#">ADC_CTRL.ch_sel</a> for valid values for this field.
23:22	-	RO	0	<b>Reserved</b>
21:12	ch_hi_limit	R/W	0x3FF	<b>High Limit Threshold</b> Sets the threshold for high-limit comparisons. This field is a 10-bit value compared against any ADC conversion on the channel set in the <i>ch_sel</i> field. ADC conversions greater than this field are over threshold and can result in interrupt assertion if the <i>ch_hi_limit_en</i> field is set. Valid values for this field are 0x000 to 0x3FF.
11:10	-	RO	0	<b>Reserved</b>
9:0	ch_lo_limit	R/W	0	<b>Low Limit Threshold</b> Sets the threshold for low-limit comparisons. This field is a 10-bit value compared against any ADC conversion on the channel set in the <i>ch_sel</i> field. ADC conversions less than this field are under threshold and can result in interrupt assertion if the <i>ch_lo_limit_en</i> field is set. Valid values for this field are 0x000 to 0x3FF.

## 10.12 Low-Power Comparator Registers

See [Table 2-4](#) for the base address of this peripheral/module. See [Table 2-1](#) 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 10-11: Low-Power Comparator Registers Summary*

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

### 10.12.1 Low-Power Comparator Register Details

*Table 10-12: Low-Power Comparator n Registers*

<b>Low-Power Comparator 1</b>			<b>LPCOMP1</b>	[0x0000]
<b>Low-Power Comparator 2</b>			<b>LPCOMP2</b>	[0x0004]
<b>Low-Power Comparator 3</b>			<b>LPCOMP3</b>	[0x0008]
Bits	Field	Access	Reset	Description
31:16	-	RO	0	<b>Reserved</b>
15	if	R/W1C	0	<b>Low-Power Comparator n Interrupt Flag</b> This field is set to 1 by hardware when the comparator output changes to the active state as set using the <i>pol</i> field. Write 1 to clear this flag. 0: No interrupt 1: Interrupt occurred

<b>Low-Power Comparator 1</b>			<b>LPCOMP1</b>		[0x0000]
<b>Low-Power Comparator 2</b>			<b>LPCOMP2</b>		[0x0004]
<b>Low-Power Comparator 3</b>			<b>LPCOMP3</b>		[0x0008]
<b>Bits</b> <b>Field</b> <b>Access</b> <b>Reset</b> <b>Description</b>					
14	out	RO	*	<b>Low-Power Comparator n Output</b> This field is the comparator's output state. 0: Output low. 1: Output high.	
13:7	-	RO	0	<b>Reserved</b>	
6	int_en	R/W	0	<b>Low-Power Comparator n Interrupt Enable</b> Set this field to 1 to enable the IRQ for specific low-power comparator. 0: Disabled 1: Enabled	
5	pol	R/W	0	<b>Comparator n Interrupt Polarity Select</b> 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	<b>Reserved</b>	
0	en	R/W	0	<b>Low-Power Comparator n Enable</b> Set this field to 1 to enable the comparator 0: Disabled 1: Enable	

## 11. 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. 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:

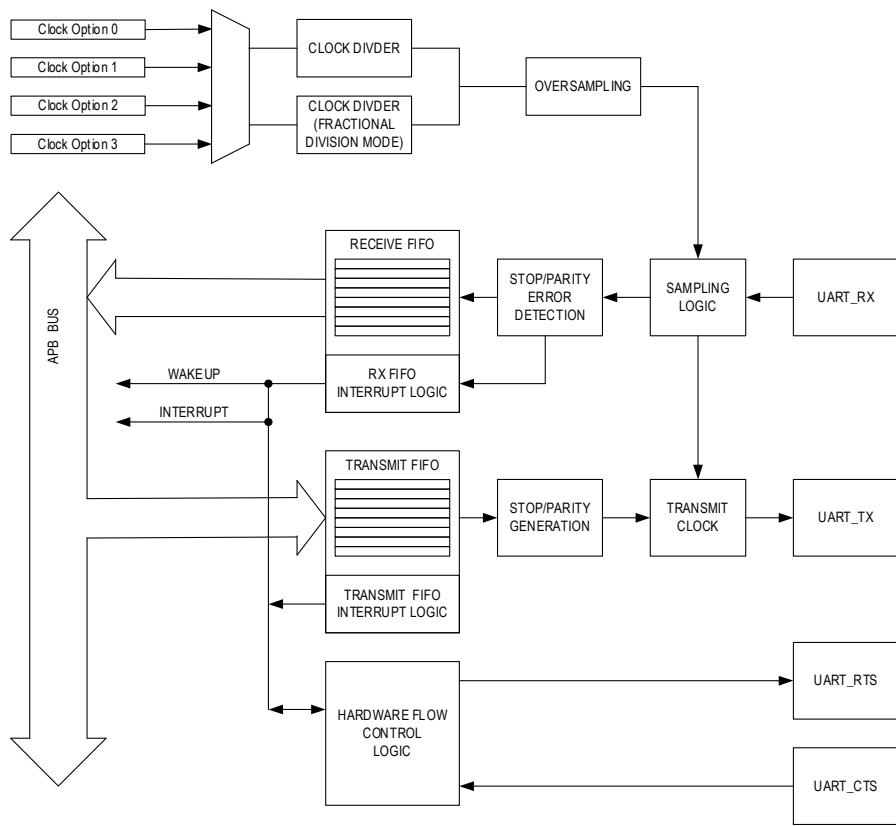
- Flexible baud rate generation up to 12.5Mbps for UART.
- Programmable character size of 5-bits to 8-bits.
- Stop bit settings of 1, 1.5, or 2-bits.
- Parity settings of even, odd, mark (always 1), space (always 0), and no parity.
- Automatic parity error detection with selectable parity bias.
- Automatic 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*, *LPM*, and *UPM* 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 11-1* shows a high-level diagram of the UART peripheral.

*Figure 11-1: UART Block Diagram*



Note: Refer to [Table 11-1](#) for the clock options supported by each UART instance.

## 11.1 Instances

Instances of the peripheral are shown in [Table 11-1](#). The standard UARTs and the LPUARTs are functionally similar; for common functionality they are referred to as UART. The LPUART instance supports fractional division mode (FDM) and is referenced as LPUART for feature specific options.

*Table 11-1: MAX78000 UART/LPUART Instances*

Instance	Register Access Name	LPUART	Power Modes	Clock Option				Hardware Flow Control	Transmit FIFO Depth	Receive FIFO Depth
				0	1	2	3			
UART0	UART0	No	ACTIVE SLEEP	PCLK	-	IBRO	-	Yes	8	8
UART1	UART1									
UART2	UART2									
LPUART0	UART3	Yes	ACTIVE SLEEP LPM UPM	-	-	IBRO	ERTCO	No		

## 11.2 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.

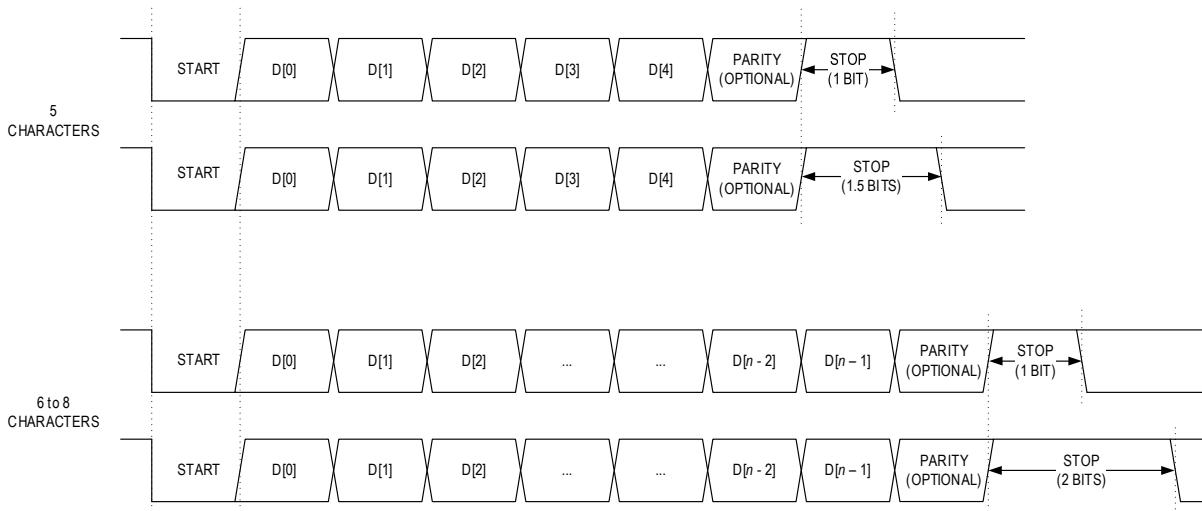
When DMA is enabled, the following describes the behavior of the DMA requests:

- 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.

## 11.3 UART Frame

[Figure 11-2](#) 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 11-2: UART Frame Structure*



## 11.4 FIFOs

Separate receive and transmit FIFOs are provided. They are both accessed through the same [\*UARTn\\_FIFO.data\*](#) field. The current level of the TX FIFO is read from [\*UARTn\\_STATUS.tx\\_lvl\*](#), and the current level of the RX FIFO is read from [\*UARTn\\_STATUS.rx\\_lvl\*](#). Data for character sizes less than 7 bits are right justified

### 11.4.1 TX FIFO Operation

Writing data to [\*UARTn\\_DMA.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 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.

### 11.4.2 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.

### 11.4.3 Flushing

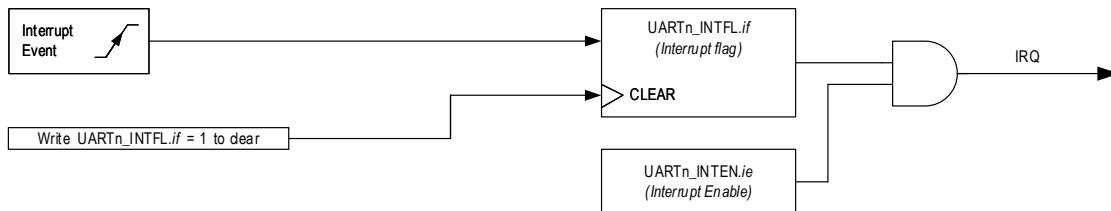
The FIFOs are flushed on the following conditions:

- 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 *GCR\_RST0 uart0*, *GCR\_RST0 uart1*, or *GCR\_RST0 uart2* field to 1

## 11.5 Interrupt Events

The peripheral generates interrupts for the events shown in *Table 11-2*. Unless noted otherwise, each instance has its own set of interrupts, and higher-level flag and enable fields as shown in *Table 11-2*.

*Figure 11-3: UART Interrupt Functional Diagram*



Some activity may cause more than one event; setting one or more event flags. 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 11-2: MAX78000 Interrupt Events*

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

### 11.5.1 Frame Error

A frame error is generated when the UART sampling circuitry detects an invalid bit. Each bit is sampled three times, as shown in *Figure 11-4*, 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 11-3](#) 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 11-4](#) for details.

*Table 11-3: Frame Error Detection for Standard UARTs and LPUART*

<a href="#"><i>UARTn_CTRL.par_en</i></a>	<a href="#"><i>UARTn_CTRL.par_md</i></a>	<a href="#"><i>UARTn_CTRL.par_eo</i></a>	Start Samples	Data Samples	Parity Samples	Stop Samples
0	N/A	N/A			Not Present	
1	0	0	3 of 3 must be 0	2/3 must match	3/3 = 1 if even number "1" 3/3 = 0 if odd number "0"	3 of 3 must be 1
	0	1			3/3 = 1 if odd number "1" 3/3 = 0 if even number "0"	
	1	0			3/3 = 1 if even number "0" 3/3 = 0 if odd number "1"	
	1	1			3/3 = 1 if odd number "0" 3/3 = 0 if even number "1"	

*Table 11-4: Frame Error Detection for LPUARTs with [\*UARTn\\_CTRL.fdm\*](#) = 1 and [\*UARTn\\_CTRL.dpfe\\_en\*](#) = 1*

<a href="#"><i>UARTn_CTRL.par_en</i></a>	<a href="#"><i>UARTn_CTRL.par_md</i></a>	<a href="#"><i>UARTn_CTRL.par_eo</i></a>	Start Samples	Data Samples	Parity Samples	Stop Samples
0	N/A	N/A			Not Present	
1	0	0	3 of 3 must be 0	3 of 3 must match	3 of 3 = 1 if even number of 1s 3 of 3 = 0 if odd number 0s	3 of 3 must be 1
	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	

### 11.5.2 Parity Error

Set [\*UARTn\\_CTRL.par\\_en\*](#) = 0 to enable parity checking of the received frame. If the calculated parity does not match the parity bit, then the corresponding interrupt flag is set. The data received is saved to the receive FIFO when a parity error occurs.

### 11.5.3 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.

### 11.5.4 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.

### 11.5.5 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\*](#).

### 11.5.6 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 11-1\*](#).

*Note: When this condition occurs, verify the number of bytes in the transmit FIFO ([\*UARTn\\_STATUS.tx\\_lvl\*](#)) prior to re-filling.*

*Equation 11-1: UART Transmit FIFO Half-Empty Condition*

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

## 11.6 LPUART Wakeup Events

LPUART instances can receive characters while in the low-power modes listed in [\*Table 11-1\*](#). If enabled, each of the receive FIFO conditions shown in [\*Table 11-5\*](#) 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 11-5: MAX78000 Wakeup Events*

Receive FIFO Condition	Wakeup Flag <a href="#"><i>UARTn_WKFL</i></a>	Wakeup Enable <a href="#"><i>UARTn_WKEN</i></a>	Low-Power Peripheral Wakeup Flag	Low-Power Peripheral Wakeup Enable	Low-Power Clock Disable
Threshold	<a href="#"><i>rx_thd</i></a>	<a href="#"><i>rx_thd</i></a>	<a href="#"><i>PWRSEQ_LPPWST uart3</i></a>	<a href="#"><i>PWRSEQ uart3</i></a>	<a href="#"><i>LPGCR_PCLKDIS uart3</i></a>
Full	<a href="#"><i>rx_full</i></a>	<a href="#"><i>rx_full</i></a>			
Not Empty	<a href="#"><i>rx_ne</i></a>	<a href="#"><i>rx_ne</i></a>			

### 11.6.1 RX FIFO Threshold

This condition persists while [\*UARTn\\_STATUS.rx\\_lvl\*](#) ≥ [\*UARTn\\_CTRL.rx\\_thd\\_val\*](#).

### 11.6.2 RX FIFO Full

This condition persists while [\*UARTn\\_STATUS.rx\\_lvl\*](#) ≥ [\*C\\_RX\\_FIFO\\_DEPTH\*](#).

### 11.6.3 RX Not Empty

This condition persists while [\*UARTn\\_STATUS.rx\\_lvl\*](#) > 0.

## 11.7 Inactive State

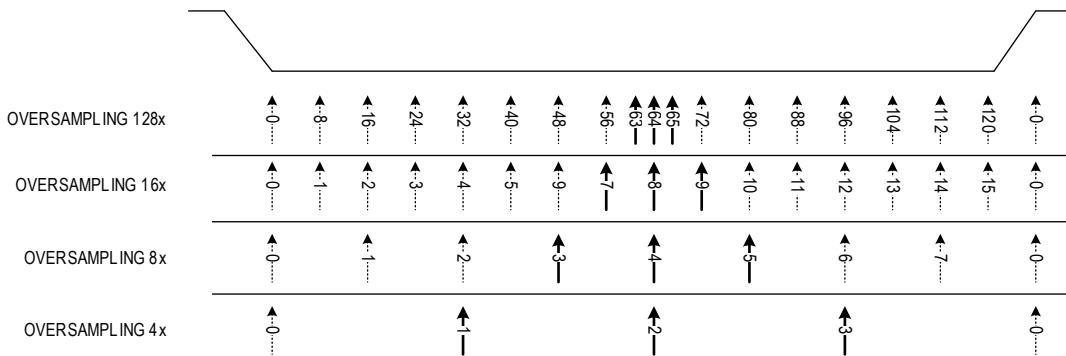
The following conditions result in the UART being inactive:

- 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

## 11.8 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 11-4*.

*Figure 11-4: Oversampling Example*



Whenever `UARTn_CLKDIV.clkdiv < 0x10` (i.e., division rate less than 8.0), OSR is not used and the over sampling 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`).

## 11.9 Baud Rate Generation

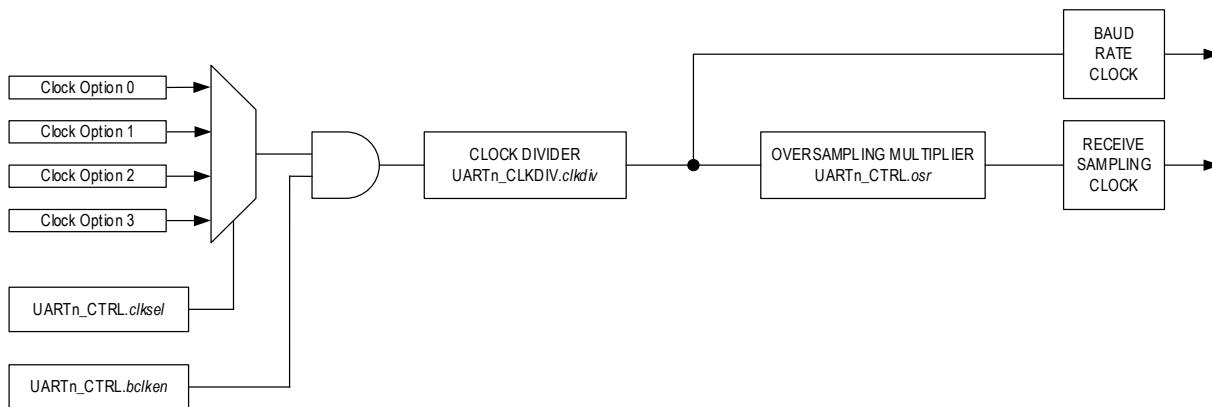
The baud rate is determined by the selected UART clock source and value of the clock divisor. Multiple clock sources are available for each UART instance. See *Table 11-1* for available clock sources.

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

### 11.9.1 UART Clock Sources

Standard UART instances operate only in *ACTIVE* and *SLEEP*. Standard UART instances can only wake the device from *SLEEP*. *Figure 11-5* shows the baud rate generation path for standard UARTs.

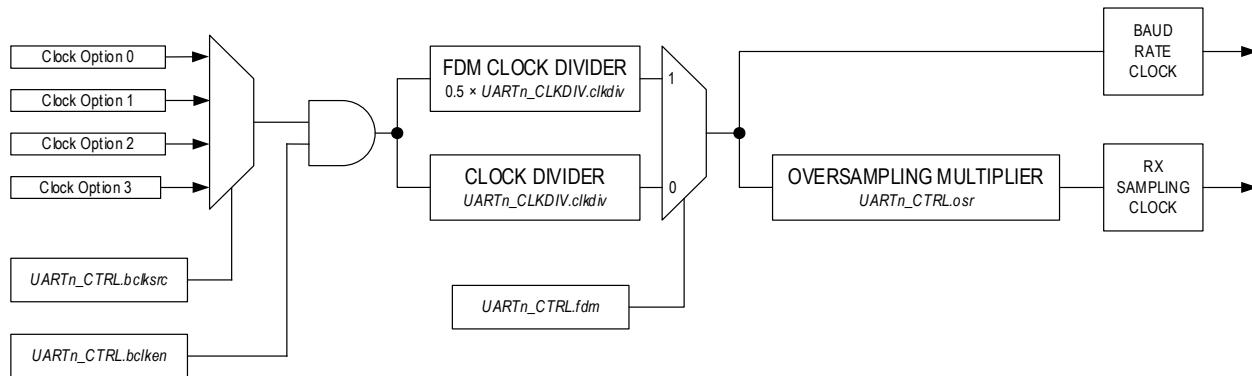
*Figure 11-5: UART Baud Rate Generation*



### 11.9.2 LPUART Clock Sources

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

*Figure 11-6: LPUART Timing Generation*



### 11.9.3 Baud Rate Calculation

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

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

*Equation 11-2: UART Clock Divisor Formula*

$$\text{UART}_n.\text{CTRL}.fdm = 0:$$

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

*Equation 11-3: LPUART Clock Divisor Formula for *UARTn\_CTRL.fdm* = 1*

$$\text{UART}_n.\text{CTRL}.fdm = 1:$$

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

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

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

### 11.9.4 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*, *LPM*, and *UPM*. Receipt of a valid frame loads the receive FIFO and increments *UARTn\_STATUS.rx\_lvl*. If a wakeup event, shown in *Table 11-5*, is enabled, the device exits the current low-power mode and

returns to *ACTIVE* if the wakeup event occurs. See section *Baud Rate Calculation* and *Equation 11-3* for details on setting the baud rate for LPUART instances with *UARTn\_CTRL.fdm* set to 1.

*Table 11-6: LPUART Low Baud Rate Generation Examples (*UARTn\_CTRL.fdm* = 1)*

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

#### 11.9.4.1 Configuring a LPUART for Low-Power Modes of Operation

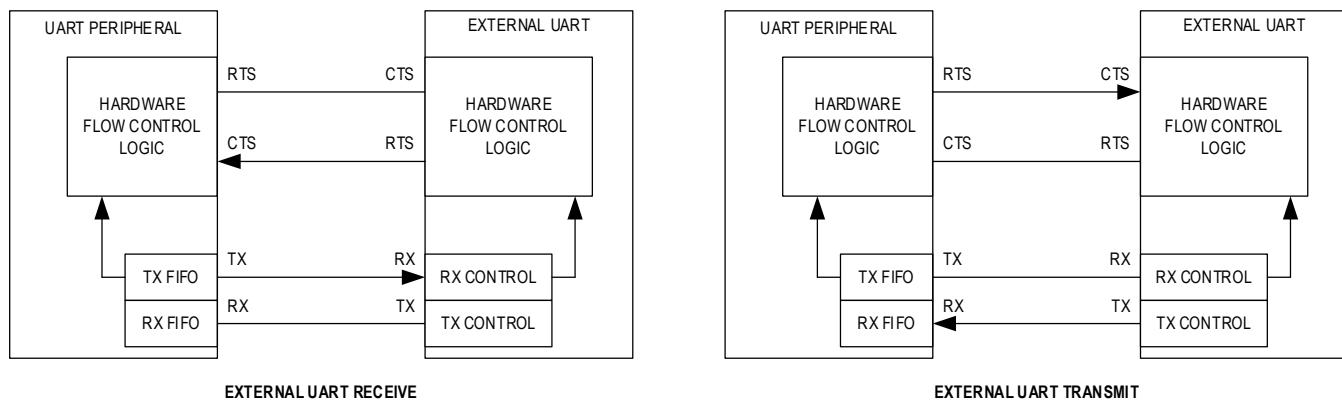
Use the following procedure to receive characters at 9600 or lower baud rates while in low-power modes:

1. Clear *UARTn\_CTRL.bclken* = 0 to disable the baud clock. The hardware immediately clears *UARTn\_CTRL.bclkrdy* to 0.
2. Set *PWRSEQ\_LPCN.x32ken* = 1 to ensure the 32kHz clock source remains active in *LPM* and *UPM* modes.
3. Ensure *UARTn\_CTRL.ucagm* = 1.
4. Configure *UARTn\_CTRL.bclksrc* to select the *ERTCO*.
5. Set *UARTn\_CTRL.fdm* to 1 to enable FDM.
6. Set *UARTn\_CLKDIV.clkdiv* to the calculated clock divisor shown in *Table 11-6* for the required baud rate.
7. Set *UARTn\_CTRL.desm* to 1 to enable receive dual edge sampling mode.
8. Choose the desired wakeup conditions from *Table 11-5*.
  - 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.
9. Set the *UARTn\_CTRL.bclken* field to 1 to enable the baud clock.
10. Poll the *UARTn\_CTRL.bclkrdy* field until it reads 1.
11. Enter the desired low-power mode.

## 11.10 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 11-7*.

*Figure 11-7: Hardware Flow Control 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 is able to receive more data. The external device then asserts CTS again allowing additional data to be sent.

Hardware flow control 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.

### 11.10.1 Automated HFC

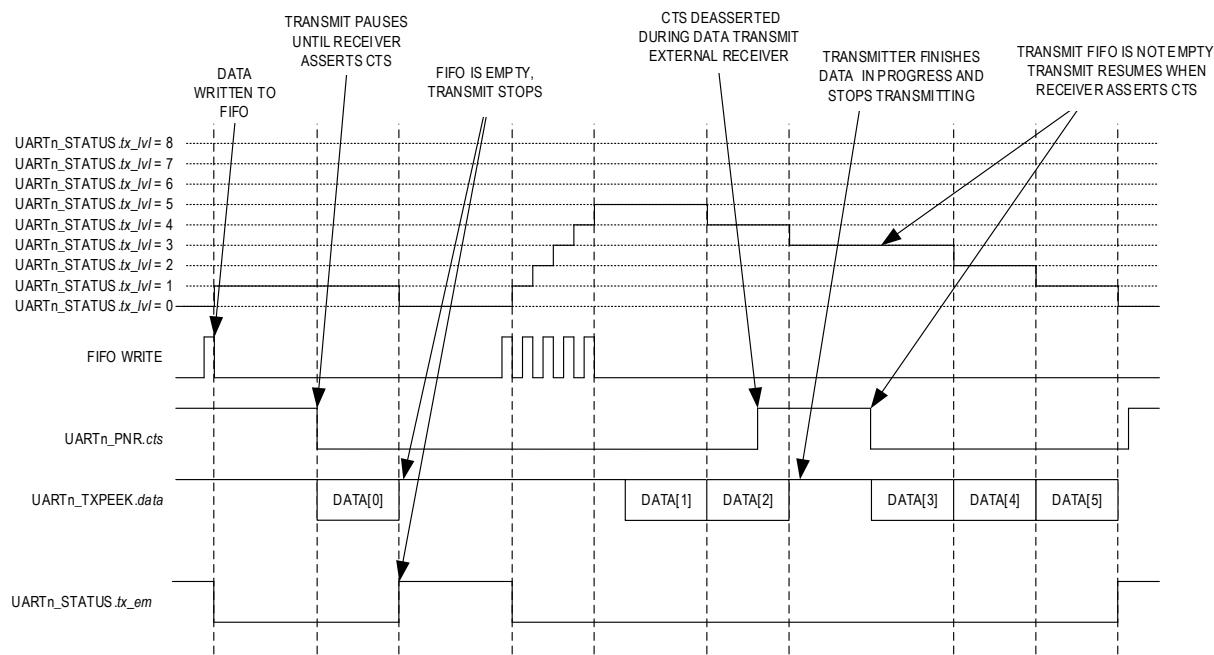
Setting `UARTn_CTRL.hfc_en = 1` enables automated HFC. When automated HFC is enabled, the hardware manages the CTS and RTS signals. The deassertion of the RTS signal is configurable using the `UARTn_CTRL.rtsdc` field:

- `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 de-asserts the CTS pin, the transmitter finishes transmission of the current character and then waits until the CTS pin state is asserted before continuing transmission. *Figure 11-8* 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. FIFO management must be handled by the software. See section *Interrupt Events* for additional information.

*Figure 11-8: Hardware Flow Control Signaling for Transmitting to an External Receiver*



### 11.10.2 Application Controlled HFC

Application controlled HFC requires the software to manually control the RTS output pin and monitor the CTS input pin. Using application 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 application controlled HFC.

#### 11.10.2.1 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 application controlled HFC.

Interrupt support for CTS input signal change events is supported even when automated HFC is disabled. The 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.*

## 11.11 Registers

See [Table 2-4](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 11-7](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn\_CTRL resolves to PERIPHERAL0\_CTRL and PERIPHERAL1\_CTRL for instances 0 and 1, respectively.

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

All registers and fields apply to both UART and LPUART instances unless specified otherwise.

*Table 11-7: UART/LPUART Register Summary*

Offset	Register	Name
[0x0000]	<a href="#">UARTn_CTRL</a>	UART Control Register
[0x0004]	<a href="#">UARTn_STATUS</a>	UART Status Register
[0x0008]	<a href="#">UARTn_INT_EN</a>	UART Interrupt Enable Register
[0x000C]	<a href="#">UARTn_INT_FL</a>	UART Interrupt Flag Register
[0x0010]	<a href="#">UARTn_CLKDIV</a>	UART Clock Divisor Register
[0x0014]	<a href="#">UARTn_OSR</a>	UART Oversampling Control Register
[0x0018]	<a href="#">UARTn_TXPEEK</a>	UART Transmit FIFO
[0x001C]	<a href="#">UARTn_PNR</a>	UART Pin Control Register
[0x0020]	<a href="#">UARTn_FIFO</a>	UART FIFO Data Register
[0x0030]	<a href="#">UARTn_DMA</a>	UART DMA Control Register
[0x0034]	<a href="#">UARTn_WKEN</a>	UART Wakeup Interrupt Enable Register
[0x0038]	<a href="#">UARTn_WKFL</a>	UART Wakeup Interrupt Flag Register

### 11.11.1 Register Details

*Table 11-8: UART Control Register*

UART Control				UARTn_CTRL	[0x0000]
Bits	Field	Access	Reset	Description	
31:23	-	DNM	0	<b>Reserved</b>	
22	desm	R/W	0	<b>Receive Dual Edge Sampling Mode</b> 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	<b>Fractional Division Mode</b> 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	
20	ucagm	R/W	0	<b>UART Clock Auto Gating Mode</b> <i>Note: Software must set this field to 1 for proper operation.</i> 0: No gating 1: UART clock is paused during transmit and receive idle states	
19	bclkrdy	R	0	<b>Baud Clock Ready</b> 0: Baud clock not ready 1: Baud clock ready	

UART Control				UARTn_CTRL	[0x0000]
Bits	Field	Access	Reset	Description	
18	dpfe_en	R/W	0	<b>Data/Parity Bit Frame Error Detection Enable</b> LPUART instances only. This field is reserved in standard UART instances. 0: Disable. Do not detect frame errors on receive between start bit and stop bit. 1: Enable. Detect frame errors when receive changes at the center of bit time.	
17:16	bclksrc	R/W	0	<b>Baud Clock Source</b> Selects the baud clock source. See <a href="#">Table 11-1</a> 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	<b>Baud Clock Enable</b> 0: Disabled 1: Enabled	
14	rtsdc	R	0	<b>Hardware Flow Control RTS Deassert Condition</b> 0: Deassert RTS when receive FIFO Level = C_RX_FIFO_DEPTH (FIFO full) 1: Deassert RTS while receive FIFO Level >= <a href="#">UARTn_CTRL.rx_thd_val</a>	
13	hfc_en	R/W	0	<b>Hardware Flow Control Enable</b> 0: Disabled 1: Enabled	
12	stopbits	R/W	0	<b>Number of Stop Bits</b> 0: 1 stop bit 1: 1.5 stop bits (for 5-bit mode) or 2 stop bits (for 6-, 7-, and 8-bit mode)	
11:10	char_size	R/W	0	<b>Character Length</b> 0: 5 bits 1: 6 bits 2: 7 bits 3: 8 bits	
9	rx_flush	W1	0	<b>Receive FIFO Flush</b> Write 1 to flush the receive FIFO. This bit will always read 0. 0: N/A 1: Flush FIFO	
8	tx_flush	W1	0	<b>Transmit FIFO Flush</b> Write 1 to flush the transmit FIFO. This bit will always read 0. 0: N/A 1: Flush FIFO	
7	cts_dis	R/W	1	<b>CTS Sampling Disable</b> 0: Enabled 1: Disabled	
6	par_md	R/W	0	<b>Parity Value Select</b> 0: Parity calculation is based on 1 bits. (Mark) 1: Parity calculation is based on 0 bits. (Space)	
5	par_eo	R/W	0	<b>Parity Odd/Even Select</b> 0: Even parity 1: Odd parity	
4	par_en	R/W	0	<b>Transmit Parity Generation Enable</b> 0: Parity transmission disabled. 1: Parity bit is calculated and transmitted after the last character bit.	

UART Control				UARTn_CTRL	[0x0000]
Bits	Field	Access	Reset	Description	
3:0	rx_thd_val	R/W	0	<b>Receive FIFO Threshold</b> Valid settings are from 1 to (C_RX_FIFO_DEPTH – 1). 0: Reserved for future use 1: 1 2: 2 3: 3 4: 4 5: 5 6: 6 7: 7 8: 8 9 - 15: Reserved for future use	

Table 11-9: UART Status Register

UART Status				UARTn_STATUS	[0x0004]
Bits	Name	Access	Reset	Description	
31:16	-	RO	0	<b>Reserved</b>	
15:12	tx_lvl	RO	0	<b>Transmit FIFO Level</b> Number of characters in the transmit FIFO. 0 - 8: Number of bytes in the transmit FIFO 9 - 15: Reserved for Future Use	
11:8	rx_lvl	RO	0	<b>Receive FIFO Level</b> Number of characters in the Receive FIFO. 0 - 8: Number of bytes in the receive FIFO 9 - 15: Reserved for Future Use	
7	tx_full	RO	0	<b>Transmit FIFO Full</b> 0: Not full 1: Full	
6	tx_em	RO	1	<b>Transmit FIFO Empty</b> 0: Not empty 1: Empty	
5	rx_full	RO	0	<b>Receive FIFO Full</b> 0: Not full 1: Full	
4	rx_em	RO	1	<b>Receive FIFO Empty</b> 0: Not empty 1: Empty	
3:2	-	RO	0	<b>Reserved</b>	
1	rx_busy	RO	0	<b>Receive Busy</b> 0: UART is not receiving a character 1: UART is receiving a character	
0	tx_busy	RO	0	<b>Transmit Busy</b> 0: UART is not transmitting data 1: UART is transmitting data	

Table 11-10: UART Interrupt Enable Register

UART Interrupt Enable Register				UARTn_INT_EN	[0x0008]
Bits	Name	Access	Reset	Description	
31:7	-	RO	0	<b>Reserved</b>	

UART Interrupt Enable Register				UARTn_INT_EN	[0x0008]
Bits	Name	Access	Reset	Description	
6	tx_he	R/W	0	<b>Transmit FIFO Half-Empty Event Interrupt Enable</b> 0: Disabled 1: Enabled	
5	-	RO	0	<b>Reserved</b>	
4	rx_thd	R/W	0	<b>Receive FIFO Threshold Event Interrupt Enable</b> 0: Disabled 1: Enabled	
3	rx_ov	R/W	0	<b>Receive FIFO Overrun Event Interrupt Enable</b> 0: Disabled 1: Enabled	
2	cts_ev	R/W	0	<b>CTS Signal Change Event Interrupt Enable</b> 0: Disabled 1: Enabled	
1	rx_par	R/W	0	<b>Receive Parity Event Interrupt Enable</b> 0: Disabled 1: Enabled	
0	rx_ferr	R/W	0	<b>Receive Frame Error Event Interrupt Enable</b> 0: Disabled 1: Enabled	

Table 11-11: UART Interrupt Flag Register

UART Interrupt Flag				UARTn_INT_FL	[0x000C]
Bits	Name	Access	Reset	Description	
31:7	-	RO	0	<b>Reserved</b>	
6	tx_he	R/W1C	0	<b>Transmit FIFO Half-Empty Interrupt Flag</b> 0: Disabled 1: Enabled	
5	-	RO	0	<b>Reserved</b>	
4	rx_thd	R/W1C	0	<b>Receive FIFO Threshold Interrupt Flag</b> 0: Disabled 1: Enabled	
3	rx_ov	R/W1C	0	<b>Receive FIFO Overrun Interrupt Flag</b> 0: Disabled 1: Enabled	
2	cts_ev	R/W1C	0	<b>CTS Signal Change Interrupt Flag</b> 0: Disabled 1: Enabled	
1	rx_par	R/W1C	0	<b>Receive Parity Error Interrupt Flag</b> 0: Disabled 1: Enabled	
0	rx_ferr	R/W1C	0	<b>Receive Frame Error Interrupt Flag</b> 0: Disabled 1: Enabled	

Table 11-12: UART Clock Divisor Register

UART Clock Divisor				UARTn_CLKDIV	[0x0010]
Bits	Name	Access	Reset	Description	
31:20	-	RO	0	<b>Reserved</b>	

UART Clock Divisor				UARTn_CLKDIV	[0x0010]
Bits	Name	Access	Reset	Description	
19:0	clkdiv	R/W	0	<b>Baud Rate Divisor</b> This field sets the divisor used to generate the baud tick from the baud clock. For LPUART instances, if <a href="#">UARTn_CTRL.fdm</a> = 1, the fractional divisors are in increments of 0.5. The over-sampling rate must be no greater than this divisor. See section <a href="#">Baud Rate Generation</a> for information on how to use this field.	

Table 11-13: UART Oversampling Control Register

UART Oversampling Control				UARTn_OSR	[0x0014]
Bits	Name	Access	Reset	Description	
31:3	-	RO	0	<b>Reserved</b>	
2:0	osr	R/W	0	<b>LPUART Over Sampling Rate</b> For LPUART instances with FDM enabled ( <a href="#">UARTn_CTRL.fdm</a> = 1): 0: 8 × 1: 12 × 2: 16 × 3: 20 × 4: 24 × 5: 28 × 6: 32 × 7: 36 ×  For LPUART instances with FDM disabled ( <a href="#">UARTn_CTRL.fdm</a> = 0): 0: 128 × 1: 64 × 2: 32 × 3: 16 × 4: 8 × 5: 4 × 6 - 7: Reserved for Future Use	

Table 11-14: UART Transmit FIFO Register

UART Transmit FIFO				UARTn_TXPEEK	[0x0018]
Bits	Name	Access	Reset	Description	
31:8	-	RO	0	<b>Reserved</b>	
7:0	data	RO	0	<b>Transmit FIFO Data</b> Read the transmit FIFO next data without effecting the contents of the transmit FIFO. If there are no entries in the transmit FIFO, this field reads 0. <i>Note: The parity bit is available from this field.</i>	

Table 11-15: UART Pin Control Register

UART Pin Control				UARTn_PNR	[0x001C]
Bits	Name	Access	Reset	Description	
31:2	-	RO	0	<b>Reserved</b>	
1	rts	R/W	1	<b>RTS Pin Output State</b> 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	<b>CTS Pin State</b> Returns the current sampled state of the GPIO associated with the CTS signal. 0: CTS state is 0 1: CTS state is 1	

Table 11-16: UART Data Register

UART Data				UARTn_FIFO	[0x0020]
Bits	Name	Access	Reset	Description	
31:9	-	RO	0	<b>Reserved</b>	
8	rx_par	R	0	<b>Receive FIFO Byte Parity</b> If parity feature is disabled, this bit always read 0. If a parity error occurred during the reception of the character at the output end of the receive FIFO (this is returned by reading the <a href="#">UARTn_FIFO.data</a> field), this bit reads 1, otherwise it reads 0.	
7:0	data	R/W	0	<b>Transmit/Receive FIFO Data</b> 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 11-17: UART DMA Register

UART DMA				UARTn_DMA	[0x0030]
Bits	Name	Access	Reset	Description	
31:10	-	RO	0	<b>Reserved</b>	
9	rx_en	0	0	<b>Receive DMA Channel Enable</b> 0: Disabled 1: Enabled	
8:5	rx_thd_val	0	0	<b>Receive FIFO Level DMA Threshold</b> If <a href="#">UARTn_STATUS.rx_lvl</a> < <a href="#">UARTn_DMA.rx_thd</a> , 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	<b>Transmit DMA Channel Enable</b> 0: Disabled 1: Enabled	
3:0	tx_thd_val	R/W	0	<b>Transmit FIFO Level DMA Threshold</b> If <a href="#">UARTn_STATUS.tx_lvl</a> < <a href="#">UARTn_DMA.tx_thd</a> , the transmit DMA channel sends a signal to the DMA indicating that the UART transmit FIFO is ready to receive data from memory.	

Table 11-18: UART Wakeup Enable

UART Wakeup Enable				UARTn_WKEN	[0x0034]
Bits	Name	Access	Reset	Description	
31:3	-	RO	0	<b>Reserved</b>	

UART Wakeup Enable			UARTn_WKEN		[0x0034]
Bits	Name	Access	Reset	Description	
2	rx_thd	R/W	0	<b>Receive FIFO Threshold Wake-up Event Enable</b> 0: Disabled 1: Enabled	
1	rx_full	R/W	0	<b>Receive FIFO Full Wake-up Event Enable</b> 0: Disabled 1: Enabled	
0	rx_ne	R/W	0	<b>Receive FIFO Not Empty Wake-up Event Enable</b> 0: Disabled 1: Enabled	

Table 11-19. UART Wakeup Flag Register

UART Wakeup Flag			UARTn_WKFL		[0x0038]
Bits	Name	Access	Reset	Description	
31:3	-	RO	0	<b>Reserved</b>	
2	rx_thd	R/W	0	<b>Receive FIFO Threshold Wake-up Event</b> 0: Disabled 1: Enabled	
1	rx_full	R/W	0	<b>Receive FIFO Full Wake-up Event</b> 0: Disabled 1: Enabled	
0	rx_ne	R/W	0	<b>Receive FIFO Not Empty Wake-up Event</b> 0: Disabled 1: Enabled	

## 12. Serial Peripheral Interface (SPI)

The Serial Peripheral Interface (SPI) is a highly configurable, flexible, and efficient synchronous interface between multiple SPI devices on a single bus. The SPI bus uses a single clock signal, and a single, dual or quad data lines, and one or more slave select lines for communication with external SPI devices.

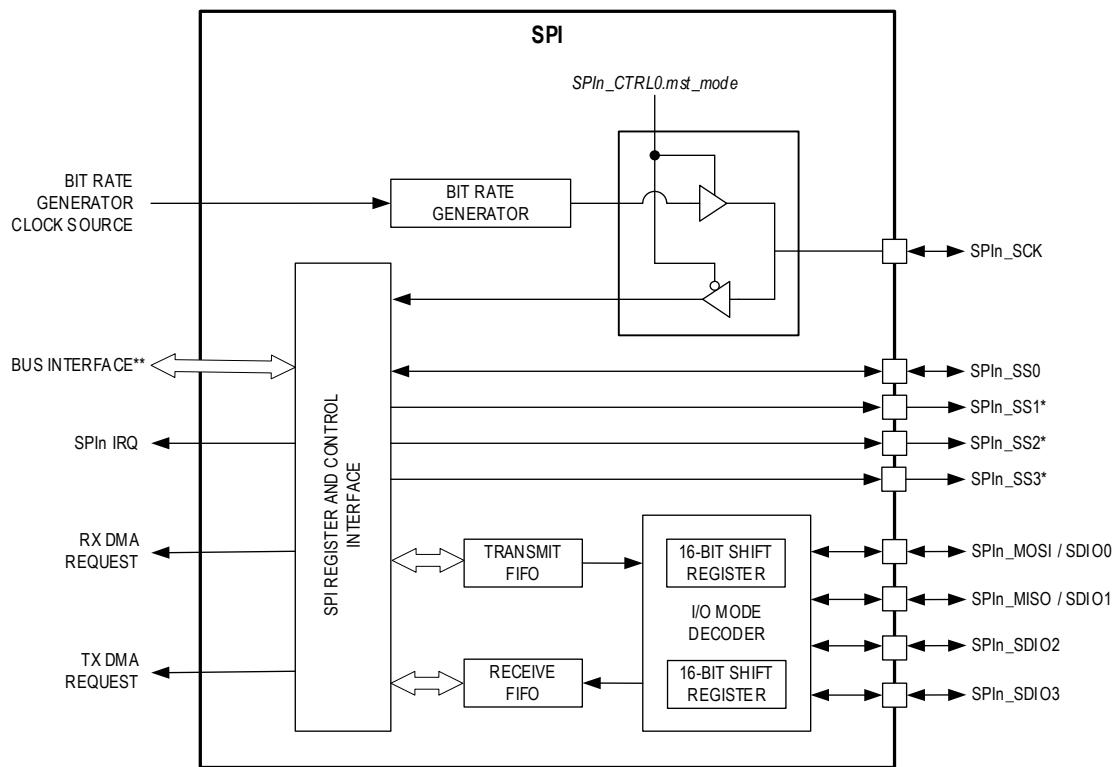
The provided SPI ports support full-duplex, bi-direction I/O and each SPI includes a Bit Rate Generator (BRG) for generating the clock signal when operating in master mode. Each SPI port operates independently and requires minimal processor overhead. All instances of the SPI peripheral support both master and slave modes, and support single master and multi-master networks.

Features include:

- Dedicated Bit Rate Generator for precision serial clock generation in Master Mode
  - ◆ Up to  $\frac{f_{PCLK}}{2}$  for instances on the APB bus
  - ◆ Up to  $\frac{f_{HCLK}}{2}$  for instances on the AHB bus
  - ◆ Programmable SCK duty cycle timing
- Full-duplex, synchronous communication of 2 to 16-bit characters
  - ◆ 1-bit and 9-bit characters are not supported
  - ◆ 2-bit and 10-bit characters do not support maximum clock speed. *SPIn\_CLKCTRL.clkdiv* must be > 0
- 3-wire and 4-wire SPI operation for single-bit communication
- Single, Dual or Quad I/O operation
- Byte-wide Transmit and Receive FIFOs with 32-byte depth
  - ◆ For character sizes greater than 8, each character uses 2 entries per character resulting in 16 entries for the Transmit and Receive FIFO
- Transmit and Receive DMA support
- SPI modes 0, 1, 2, 3
- Configurable slave select lines
  - ◆ Programmable slave select level
- Programmable slave select timing with respect to SCK starting edge and ending edge
- Multi-master mode fault detection

*Figure 12-1* shows a high-level block diagram of the SPI peripheral. See *Table 12-1* for the peripheral-specific peripheral bus assignment and bit rate generator clock source.

*Figure 12-1: SPI Block Diagram*



\* The number of slave select signals can vary for each instance of the peripheral.

\*\* The bus interface (APB or AHB) can vary for each instance of the peripheral.

## 12.1 Instances

There are two instances of the SPI peripheral as shown in *Table 12-1*. *Table 12-2* lists the locations of the SPI signals for each of the SPI instances.

*Table 12-1: MAX78000 SPI Instances*

Instance	Formats				Hardware Bus	Bit Rate Generator Clock Source Frequency	Slave Select Signals
	3-Wire	4-Wire	Dual	Quad			
SPI0	Yes	Yes	Yes	Yes	AHB	$f_{SYS\_CLK}$	3
SPI1	Yes	Yes	Yes	Yes	APB	$f_{PCLK}$	1

*Note:* Refer to the MAX78000 data sheet for the definitive list of alternate function assignments for each peripheral.

Table 12-2: MAX78000 SPI Peripheral Pins

Instance	Signal Description	Alternate Function	Alternate Function Number	81 CTBGA
SPI0	SPI Clock	SPI0_SCK	AF1	P0.7
	Slave Select 0	SPI0_SS0	AF1	P0.4
	Slave Select 1	SPI0_SS1	AF2	P0.11
	Slave Select 2	SPI0_SS2	AF2	P0.10
	MOSI (SDIO0)	SPI0_MOSI	AF1	P0.5
	MISO (SDIO1)	SPI0_MISO	AF1	P0.6
	SDIO2	SPI0_SDIO2	AF1	P0.8
	SDIO3	SPI0_SDIO3	AF1	P0.9
SPI1	SPI Clock	SPI1_SCK	AF1	P0.23
	Slave Select 0	SPI1_SS0	AF1	P0.20
	MOSI (SDIO0)	SPI1_MOSI	AF1	P0.21
	MISO (SDIO1)	SPI1_MISO	AF1	P0.22
	SDIO2	SPI1_SDIO2	AF1	P0.24
	SDIO3	SPI1_SDIO3	AF1	P0.25

## 12.2 Format

### 12.2.1 Four-Wire SPI

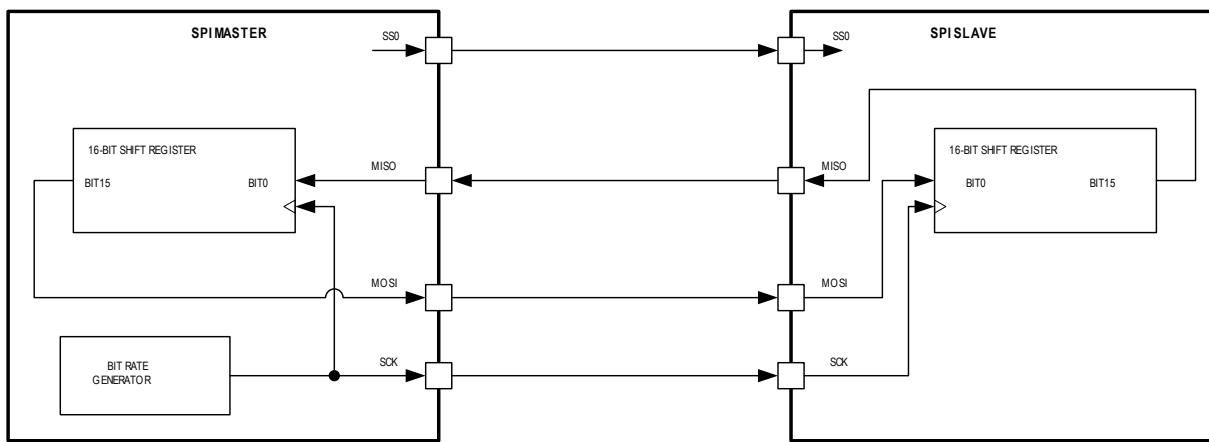
SPI devices operate as either a master or slave device. In four-wire SPI, four signals are required for communication as shown in *Table 12-3*.

Table 12-3: Four-Wire Format Signals

Signal	Description	Direction
SCK	Serial Clock	The master generates the Serial Clock signal, which is an output from the master and an input to the slave.
MOSI	Master Output Slave Input	In master mode, this signal is used as an output for sending data to the slave. In slave mode this is the input data from the master.
MISO	Master Input Slave Output	In master mode, this signal is used as an input for receiving data from the slave. In slave mode, this signal is an output for transmitting data to the master.
SS	Slave Select	In master mode, this signal is an output used to select a slave device prior to communication. Peripherals may have multiple slave select outputs to communicate with one or more external slave devices.  In slave mode, SPIn_SS0 is a dedicated input that indicates an external master is going to start communication. Other slave select signals into the peripheral are ignored in slave mode.

In a typical SPI network, the master device selects the slave device using the slave select output. The master starts the communication by selecting the slave device by asserting the slave select output. The master then starts the SPI clock through the SCK output pin. When a slave device's slave select pin is deasserted, the device is required to put the SPI pins in tri-state mode.

*Figure 12-2: 4-Wire SPI Connection Diagram*



### 12.2.2 Three-Wire SPI

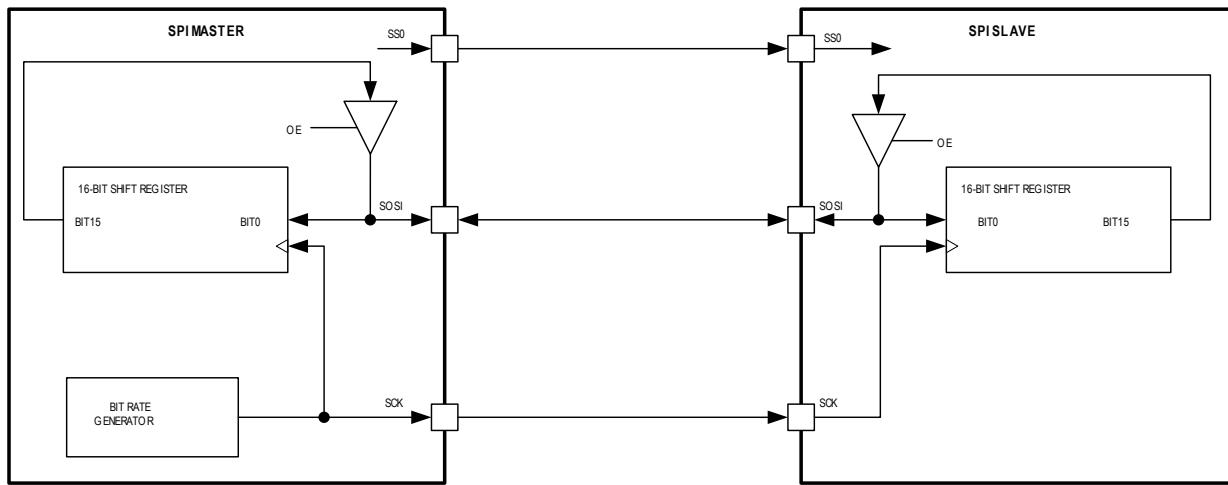
The signals in three-wire SPI operation are shown in *Table 12-4*. The MOSI signal is used as a bi-directional, half-duplex I/O referred to as Slave Input Slave Output (SISO). Three-wire SPI also uses a serial clock signal generated by the master and a slave select pin controlled by the master.

*Table 12-4: Three-Wire Format Signals*

Signal	Description	Direction
SCK	Serial Clock	The master generates the serial clock signal, which is an output from the master and an input to the slave.
MOSI	Master Output Slave Input	This is a half-duplex, bidirectional I/O pin used for communication between the SPI master and slave. This signal is used to transmit data from the master to the slave and to receive data from the slave by the master.
SS	Slave Select	In master mode, this signal is an output used to select a slave device prior to communication. In slave mode, SPI <sub>n</sub> _SS <sub>in</sub> is a dedicated input that indicates an external master is going to start communication. Other slave select signals into the peripheral are ignored in slave mode

A three-wire SPI network is shown in *Figure 12-3*. The master device selects the slave device using the slave select output. The communication starts with the master asserting the slave select line and then starting the clock (SCK). In three-wire SPI communication, the master and slave must both know the intended direction of the data to prevent bus contention. For a write, the master drives the data out the SISO pin. For a read, the master must release the SISO line and let the slave drive the SISO line. The direction of transmission is controlled using the FIFO. Writing to the FIFO starts the three-wire SPI write and reading from the FIFO starts a three-wire SPI read transaction.

*Figure 12-3: Generic 3-Wire SPI Master to Slave Connection*



## 12.3 Pin Configuration

Before configuring the SPI peripheral, first disable any SPI activity for the port by clearing the `SPIn_CTRL0.en` field to 0.

### 12.3.1 SPIn Alternate Function Mapping

Pin selection and configuration is required to use the SPI port. The following information applies to SPI master and slave operation as well as three-wire, four-wire, dual and quad mode communications. Determine the pins required for the SPI type and mode in the application, and configure the required GPIO as described in the following sections. Refer to the MAX78000 data sheet for pin availability for a specific package.

When the SPI port is disabled, `SPIn_CTRL0.en` = 0, the GPIO pins enabled for SPI alternate function are placed in high-impedance input mode.

### 12.3.2 Four-wire Format Configuration

Four-wire SPI uses SCK, MISO, MOSI, and one or more SS pins. Four-wire SPI may use more than one slave select pin for a transaction, resulting in more than four wires total, however the communication is referred to as four wire for legacy reasons.

*Note: Select the pins mapped to the SPI external device in the design and modify the setup accordingly. There is no restriction on which alternate function is used for a specific SPI pin and each SPI pin can be used independently from the other pins chosen. However, it is recommended that only one set of GPIO port pins be used for any network.*

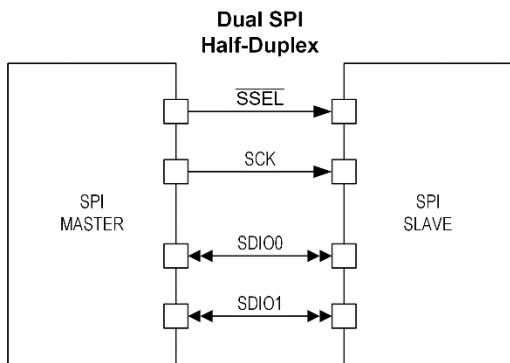
### 12.3.3 Three-Wire Format Configuration

Three-wire SPI uses SCK and MOSI, and one or more slave select pins for an SPI transaction. Three-wire SPI configuration is identical to the four-wire configuration except SPIn\_MISO does not need to be set up for the SPI alternate function. The direction of communication in three-wire SPI mode is controlled by the SPIn Transmit and Receive FIFO enables. Enabling the Receive FIFO and disabling the Transmit FIFO indicates a read transaction. Enabling the Transmit FIFO and disabling the Receive FIFO indicates a write transaction. It is an illegal condition to enable both the Transmit and Receive FIFOs in three-wire SPI operation.

### 12.3.4 Dual-Mode Format Configuration

In dual-mode SPI, two I/O pins are used to transmit 2-bits of data per SCK clock cycle. The communication is half-duplex and the direction of the data transmission must be known by both the master and slave for a given transaction. Dual-mode SPI uses SCK, SDIO0, SDIO1, and one or more slave select lines as shown in [Figure 12-4](#). The configuration of the GPIO pins for dual mode SPI is identical to four-wire SPI and the mode is controlled by setting `SPIIn_CTRL2.data_width` to 1 indicating to the SPIn hardware to use SDIO0 and SDIO1 for half-duplex communication rather than full-duplex communication.

*Figure 12-4: Dual Mode SPI Connection Diagram*



### 12.3.5 Quad-Mode Format Pin Configuration

Quad-mode SPI uses four I/O pins to transmit four-bits of data per transaction. In quad-mode SPI, the communication is half-duplex, and the master and slave must know the direction of transmission for each transaction. Quad-mode SPI uses SCK, SDIO0, SDIO1, SDIO2, SDIO3, and one or more slave select pins.

Quad-mode SPI transmits four bits per SCK cycle. Selection of Quad mode SPI is selected by setting `SPIIn_CTRL2.data_width` to 2.

## 12.4 Clock Configuration

### 12.4.1 Serial Clock

The SCK signal synchronizes data movement in and out of the device. The master drives SCK as an output to the slave's SCK pin. When SPIn is set to master mode, the SPIn bit rate generator creates the serial clock and outputs it on the configured SPIn\_SCK pin. When SPIn is configured for slave operation, the SPIn\_SCK pin is an input from the external master and the SPIn hardware synchronizes communications using the SCK input. Operating as a slave, if a SPIn slave select input is not asserted, the SPIn ignores any signals on the serial clock and serial data lines.

In both master and slave devices, data is shifted on one edge of the SCK and is sampled on the opposite edge where data is stable. Data availability and sampling time is controlled using the SPI phase control field, `SPIIn_CTRL2.clkpha`. The SCK clock polarity field, `SPIIn_CTRL2.clkpol`, controls if the SCK signal is active high or active low.

The SPIn peripheral supports four combinations of SCK phase and polarity referred to as SPI Modes 0, 1, 2, and 3. Clock Polarity (`SPIIn_CTRL2.clkpol`) selects an active low/high clock and has no effect on the transfer format. Clock Phase (`SPIIn_CTRL2.clkpha`) selects one of two different transfer formats.

For proper data transmission, the clock phase and polarity must be identical for the SPI master and slave. The master always places data on the MOSI line a half-cycle before the SCK edge for the slave to latch the data. Refer to section [Clock Phase and Polarity Control](#) for additional details.

### 12.4.2 Peripheral Clock

See [Table 12-1](#) 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 SPIn peripheral clock. The SPIn provides an internal clock, *SPIn\_CLK*, that is used within the SPI peripheral for the base clock to control the module and generate the SCK clock when in master mode. Set the SPIn internal clock using the field *SPIn\_CLKCTRL.scale* as shown in [Equation 12-1](#). Valid settings for *SPIn\_CLKCTRL.scale* are 0 to 8, allowing a divisor of 1 to 256.

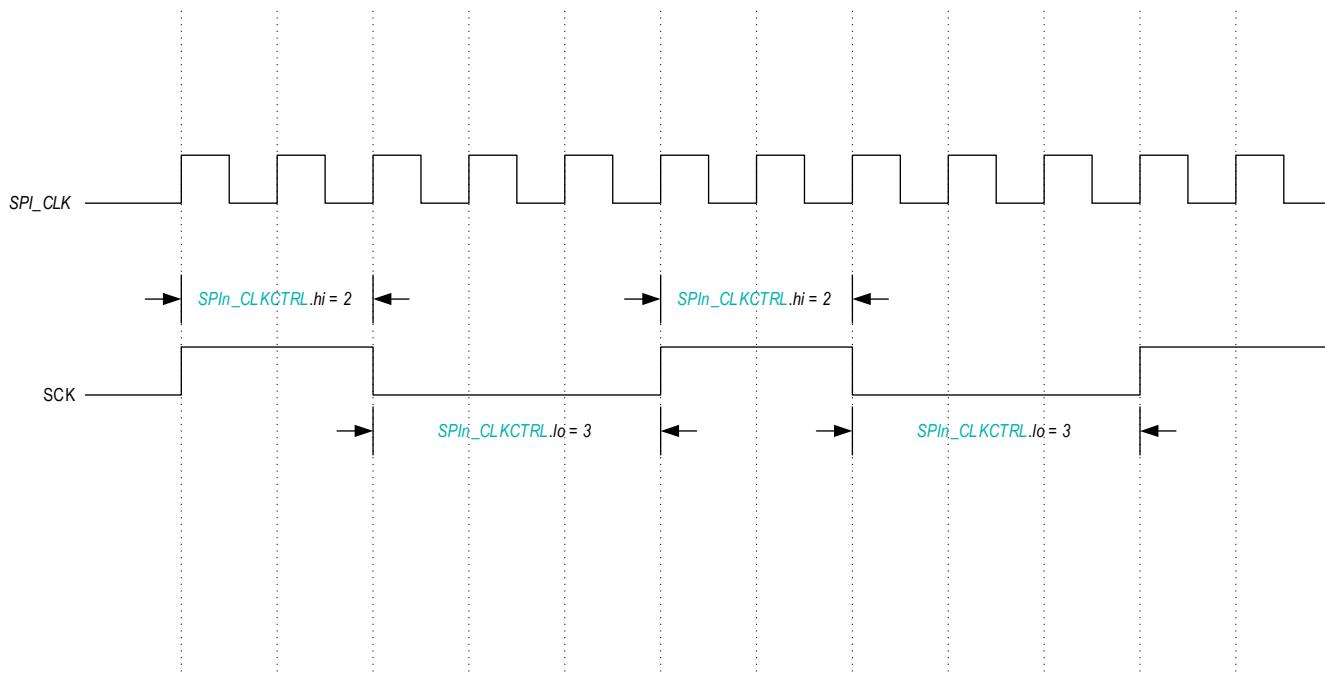
*Equation 12-1: SPI Peripheral Clock*

$$f_{\text{SPI\_CLK}} = \frac{f_{\text{INPUT\_CLK}}}{2^{\text{clkdiv}}}$$

### 12.4.3 Master Mode Serial Clock Generation

In master and multi-master mode, the SCK clock is generated by the master. The SPIn provides control for both the high time and low time of the SCK clock. This control allows setting the high and low times for the SCK to duty cycles other than 50% if required. The SCK clock uses the SPIn peripheral clock as a base value, and the high and low values are a count of the number of  $f_{\text{SPI\_CLK}}$  clocks. [Figure 12-5](#), visually represents the use of the *SPIn\_CLKCTRL.hi* and *SPIn\_CLKCTRL.lo* fields for a non-50% duty cycle serial clock generation. See [Equation 12-2](#) and [Equation 12-3](#) for calculating the SCK high and low time from the *SPIn\_CLKCTRL.hi* and *SPIn\_CLKCTRL.lo* field values.

*Figure 12-5: SCK Clock Rate Control*



*Equation 12-2: SCK High Time*

$$t_{\text{SCK\_HI}} = t_{\text{SPIn\_CLK}} \times \text{SPIn\_CLKCTRL.hi}$$

*Equation 12-3: SCK Low Time*

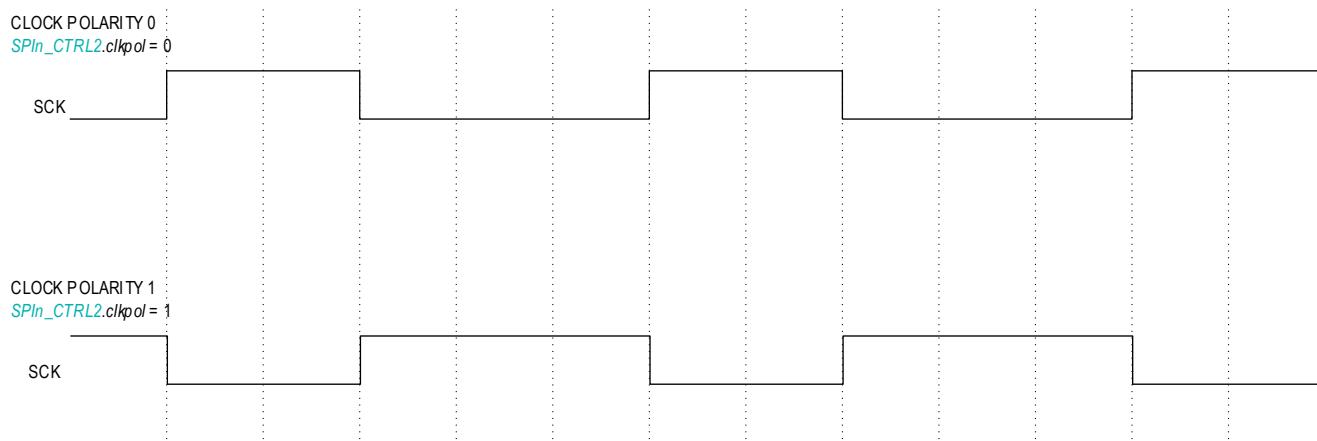
$$t_{\text{SCK\_LOW}} = t_{\text{SPIn\_CLK}} \times \text{SPIn\_CLKCTRL.lo}$$

### 12.4.4 Clock Phase and Polarity Control

SPI supports four combinations of clock and phase polarity as shown in [Table 12-5](#). Clock polarity is controlled using the bit *SPIn\_CTRL2.clkpol* and determines if the clock is active high or active low as shown in [Figure 12-6](#). Clock polarity does not affect the transfer format for SPI. Clock phase determines when the data must be stable for sampling. Setting the clock phase to 0, *SPIn\_CTRL2.clkpha* = 0, dictates the SPI data is sampled on the initial SPI clock edge regardless of clock polarity.

Phase 1, `SPIn_CTRL2.clkpha = 1`, results in data sample occurring on the second edge of the clock regardless of clock polarity.

Figure 12-6: SPI Clock Polarity



For proper data transmission, the clock phase and polarity must be identical for the SPI master and slave. The master always places data on the MOSI line a half-cycle before the SCK edge for the slave to latch the data.

Table 12-5: SPI Modes Clock Phase and Polarity Operation

SPI Mode	<code>SPI<sub>n</sub>_CTRL2 clkpha</code>	<code>SPI<sub>n</sub>_CTRL2 clkpol</code>	SCK Transmit Edge	SCK Receive Edge	SCK Idle State
0	0	0	Falling	Rising	Low
1	0	1	Rising	Falling	High
2	1	0	Rising	Falling	Low
3	1	1	Falling	Rising	High

#### 12.4.5 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.

#### 12.4.6 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
  - ◆ Slave mode only, Master mode stalls the serial clock
- Transmit FIFO Overrun
- Receive FIFO Underrun
- Receive FIFO Overrun
  - ◆ Slave Mode only, Master Mode stalls the serial clock
- SPIn supports interrupts for the internal state of the SPI as well as external signals. The following transmission interrupts are supported:
  - SS<sub>n</sub> asserted or deasserted
  - SPI transaction complete
    - ◆ Master mode only
  - Slave mode transaction aborted
  - Multi-master fault

The SPI<sub>n</sub> port can wake up the microcontroller from low-power modes when the wake event is enabled. SPI<sub>n</sub> events that can wake the microcontroller are:

- Receive FIFO full
- Transmit FIFO empty
- Receive FIFO threshold
- Transmit FIFO threshold

## 12.5 Registers

See [Table 2-4](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 12-6](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERAL<sub>n</sub>\_CTRL resolves to PERIPHERAL0\_CTRL and PERIPHERAL1\_CTRL for instances 0 and 1, respectively.

See [Table 2-1](#) 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 12-6: SPI<sub>n</sub> Register Summary*

Offset	Register Name	Access	Description
[0x0000]	<a href="#">SPI<sub>n</sub>_FIFO32</a>	R/W	SPI FIFO Data Register
[0x0000]	<a href="#">SPI<sub>n</sub>_FIFO16</a>	R/W	SPI 16-bit FIFO Data Register
[0x0000]	<a href="#">SPI<sub>n</sub>_FIFO8</a>	R/W	SPI 8-bit FIFO Data Register
[0x0004]	<a href="#">SPI<sub>n</sub>_CTRL0</a>	R/W	SPI Master Signals Control Register
[0x0008]	<a href="#">SPI<sub>n</sub>_CTRL1</a>	R/W	SPI Transmit Packet Size Register
[0x000C]	<a href="#">SPI<sub>n</sub>_CTRL2</a>	R/W	SPI Static Configuration Register
[0x0010]	<a href="#">SPI<sub>n</sub>_SSTIME</a>	R/W	SPI Slave Select Timing Register
[0x0014]	<a href="#">SPI<sub>n</sub>_CLKCTRL</a>	R/W	SPI Master Clock Configuration Register
[0x001C]	<a href="#">SPI<sub>n</sub>_DMA</a>	R/W	SPI DMA Control Register
[0x0020]	<a href="#">SPI<sub>n</sub>_INTFL</a>	R/W1C	SPI Interrupt Flag Register
[0x0024]	<a href="#">SPI<sub>n</sub>_INTEN</a>	R/W	SPI Interrupt Enable Register
[0x0028]	<a href="#">SPI<sub>n</sub>_WKFL</a>	R/W1C	SPI Wakeup Flags Register

Offset	Register Name	Access	Description
[0x002C]	<i>SPIn_WKEN</i>	R/W	<i>SPI Wakeup Enable Register</i>
[0x0030]	<i>SPIn_STAT</i>	RO	<i>SPI Status Register</i>

### 12.5.1 Register Details

Table 12-7: SPI FIFO32 Register

SPI FIFO Data				<i>SPIn_FIFO32</i>	[0x0000]
Bits	Name	Access	Reset	Description	
31:0	data	R/W	0	<b>SPIn FIFO Data Register</b> This register is used for the SPI Transmit and Receive FIFO. Reading from this register returns characters from the Receive FIFO and writing to this register adds characters to the Transmit FIFO. Read and write this register in either 1-byte, 2-byte, or 4-byte widths only. Reading from an empty FIFO or writing to a full FIFO results in undefined behavior.	

Table 12-8: SPI 16-bit FIFO Register

SPI FIFO Data				<i>SPIn_FIFO16</i>	[0x0000]
Bits	Name	Access	Reset	Description	
31:16	-	R/W	0	<b>Reserved</b>	
15:0	data	R/W	0	<b>SPIn 16-bit FIFO Data Register</b> This register is used for the SPI Transmit and Receive FIFO. Reading from this register returns characters from the Receive FIFO and writing to this register adds characters to the Transmit FIFO. Read and write this register in 2-byte width only for 16-bit FIFO access. Reading from an empty FIFO or writing to a full FIFO results in undefined behavior.	

Table 12-9: SPI 8-bit FIFO Register

SPI 8-bit FIFO Data				<i>SPIn_FIFO8</i>	[0x0000]
Bits	Name	Access	Reset	Description	
31:0	-	RO	0	<b>Reserved</b>	
7:0	data	R/W	0	<b>SPIn 8-bit FIFO Data Register</b> This register is used for the SPI Transmit and Receive FIFO. Reading from this register returns characters from the Receive FIFO and writing to this register adds characters to the Transmit FIFO. Read and write this register in 1-byte width only for 8-bit FIFO access. Reading from an empty FIFO or writing to a full FIFO results in undefined behavior.	

Table 12-10: SPI Control 0 Register

SPI Control 0				<i>SPIn_CTRL0</i>	[0x0004]
Bits	Name	Access	Reset	Description	
31:20	-	RO	0	<b>Reserved</b>	

SPI Control 0				SPIn_CTRL0	[0x0004]
Bits	Name	Access	Reset	Description	
19:16	ss_active	R/W	0	<b>Master Slave Select</b> The SPIn includes up to four slave select lines for each port. This field selects which slave select pin is active when the next SPI transaction is started ( <i>SPIn_CTRL0.start</i> = 1). One or more slave select pins can be selected for each SPI transaction by setting the bit for each slave select pin. For example, use SPIn_SS0 and SPIn_SS2 by setting this field to 0b0101 or select all slave selects by setting this field to 0b1111. <i>Note: This field is only used when the SPIn is configured for Master Mode (<i>SPIn_CTRL0.mst_mode</i> = 1).</i>	
15:9	-	RO	0	<b>Reserved</b>	
8	ss_ctrl	R/W	0	<b>Master Slave Select Control</b> This field controls the behavior of the slave select pins at the completion of a transaction. The default behavior, <i>ss_ctrl</i> = 0, deasserts the slave select pin at the completion of the transaction. Set this field to 1 to leave the slave select pins asserted at the completion of the transaction. If the external device supports this behavior, leaving the slave select pins asserted allows multiple transactions without the delay associated with deassertion of the slave select pin between transactions. 0: Slave Select is deasserted at the end of a transmission 1: Slave Select stays asserted at the end of a transmission	
7:6	-	RO	0	<b>Reserved</b>	
5	start	R/W1O	0	<b>Master Start Data Transmission</b> Set this field to 1 to start a SPI master mode transaction. 0: No master mode transaction active. 1: Master initiates a data transmission. Ensure that all pending transactions are complete before setting this field to 1. <i>Note: This field is only used when the SPIn is configured for Master Mode (<i>SPIn_CTRL0.master</i> = 1).</i>	
4	ss_io	R/W	0	<b>Master Slave Select Signal Direction</b> Set the I/O direction for 0: Slave Select is an output 1: Slave Select is an input <i>Note: This field is only used when the SPIn is configured for Master Mode (<i>SPIn_CTRL0.mst_mode</i> = 1).</i>	
3:2	-	RO	0	<b>Reserved</b>	
1	mst_mode	R/W	0	<b>SPI Master Mode Enable</b> This field selects between slave mode and master mode operation for the SPI port. Write this field to 0 to operate as an SPI slave. Setting this field to 1 sets the port as an SPI master. 0: Slave mode SPI operation. 1: Master mode SPI operation.	
0	en	R/W	0	<b>SPI Enable/Disable</b> This field enables and disables the SPIn port. Disable the SPIn port by setting this field to 0. Disabling the SPIn port does not affect the SPIn FIFOs or register settings. Access to SPIn registers is always available. 0: SPIn port is disabled 1: SPIn port is enabled	

*Table 12-11: SPI Control 1 Register*

SPI Transmit Packet Size				SPI <sub>n</sub> _CTRL1	[0x0008]
Bits	Name	Access	Reset	Description	
31:16	rx_num_char	R/W	0	<b>Number of Receive Characters</b> Number of characters to receive in receive FIFO.  <i>Note: If the SPI<sub>n</sub> port is set to operate in 4-wire mode, this field is ignored and the SPI<sub>n</sub>_CTRL1.tx_num_chars field is used for both the number of characters to receive and transmit.</i>	
15:0	tx_num_char	R/W	0	<b>Number of Transmit Characters</b> Number of characters to transmit from transmit FIFO.  <i>Note: If the SPI<sub>n</sub> port is set to operate in 4-wire mode, this field is used for both the number of characters to receive and transmit.</i>	

*Table 12-12: SPI Control 2 Register*

SPI Control 2				SPI <sub>n</sub> _CTRL2	[0x000C]
Bits	Name	Access	Reset	Description	
31:20	-	RO	0	<b>Reserved</b>	
19:16	ss_pol	R/W	0	<b>Slave Select Polarity</b> Controls the polarity of each individual SS signal where each bit position corresponds to a SS signal. SPI <sub>n</sub> _SS0 is controlled with bit position 0 and SPI <sub>n</sub> _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	<b>Three-Wire SPI Enable</b> Set this field to 1 to enable three-wire SPI communication. Set this field to 0 for four-wire full-duplex SPI communication. 0: Four-wire full-duplex mode enabled. 1: Three-wire mode enabled  <i>Note: This field is ignored for Dual SPI, SPI<sub>n</sub>_CTRL2.data_width =1, and Quad SPI, SPI<sub>n</sub>_CTRL2.data_width =2.</i>	
14	-	RO	0	<b>Reserved</b>	

SPI Control 2				SPI <sub>n</sub> _CTRL2	[0x000C]
Bits	Name	Access	Reset	Description	
13:12	data_width	R/W	0b00	<p><b>SPI Data Width</b>            This field controls the number of data lines used for SPI communications.</p> <p><b>Three-wire SPI:</b> <i>data_width</i> = 0            Set this field to 0, indicating SPI<sub>n</sub>_MOSI is used for half-duplex communication.</p> <p><b>Four-wire full-duplex SPI:</b> <i>data_width</i> = 0            Set this field to 0, indicating SPI<sub>n</sub>_MOSI and SPI<sub>n</sub>_MISO are used for the SPI data output and input respectively.</p> <p><b>Dual Mode SPI:</b> <i>data_width</i> = 1            Set this field to 1, indicating SPI<sub>n</sub>_SDIO0 and SPI<sub>n</sub>_SDIO1 are used for half-duplex communication.</p> <p><b>Quad Mode SPI:</b> <i>data_width</i> = 2            Set this field to 2, indicating SPI<sub>n</sub>_SDIO0, SPI<sub>n</sub>_SDIO1, SPI<sub>n</sub>_SDIO2 and SPI<sub>n</sub>_SDIO3 are used for half-duplex communication.</p> <ul style="list-style-type: none"> <li>0: 1-bit per SCK cycle (Three-wire half-duplex SPI and Four-wire full-duplex SPI)</li> <li>1: 2-bits per SCK cycle (Dual Mode SPI)</li> <li>2: 4-bits per SCK cycle (Quad Mode SPI)</li> <li>3: Reserved</li> </ul> <p><i>Note:</i> When this field is set to 0, use the field SPI<sub>n</sub>_CTRL2.three_wire to select either Three-Wire SPI or Four-Wire SPI operation.</p>	
11:8	numbits	R/W	0	<p><b>Number of Bits per Character</b>            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.</p> <ul style="list-style-type: none"> <li>0: 16-bits per character</li> <li>1: 1-bit per character (not supported)</li> <li>2: 2-bits per character</li> <li>...</li> <li>14: 14-bits per character</li> <li>15: 15-bits per character</li> </ul> <p><i>Note:</i> 1-bit and 9-bit character lengths are not supported.</p> <p><i>Note:</i> 2-bit and 10-bit character lengths do not support maximum SCK speeds in Master mode. SPI<sub>n</sub>_CLK.scale must be &gt; 0</p> <p><i>Note:</i> For Dual and Quad mode SPI, the character size should be divisible by the number of bits per SCK cycle.</p>	
7:2	-	RO	0	<b>Reserved</b>	
1	clkpol	R/W	0	<p><b>Clock Polarity</b>            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.</p> <ul style="list-style-type: none"> <li>0: Standard SCK for use in SPI Mode 0 and Mode 1</li> <li>1: Inverted SCK for use in SPI Mode 2 and Mode 3</li> </ul>	
0	clkpha	R/W	0	<p><b>Clock Phase</b>            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</p>	

Table 12-13: SPI Slave Select Timing Register

SPI Slave Select Timing				SPI <sub>n</sub> _SSTIME	[0x0010]
Bits	Name	Access	Reset	Description	
31:24	-	RO	0	<b>Reserved</b>	

SPI Slave Select Timing				SPI <sub>n</sub> _SSTIME	[0x0010]
Bits	Name	Access	Reset	Description	
23:16	inact	R/W	0	<b>Inactive Stretch</b> This field controls the number of system clocks the bus is inactive between the end of a transaction (Slave Select inactive) and the start of the next transaction (Slave Select active). 0: 256 1: 1 2: 2 3:3 ... ... 254: 254 255: 255	<small>Note: The SPI<sub>n</sub>_SSTIME register bit settings only apply when SPI<sub>n</sub> is operating in Master mode (SPI<sub>n</sub>_CTRL0.master = 1)</small>
15:8	post	R/W	0	<b>Slave Select Hold Post Last SCK</b> Number of system clock cycles that SS remains active after the last SCK edge. 0: 256 1: 1 2: 2 3:3 ... ... 254: 254 255: 255	<small>Note: The SPI<sub>n</sub>_SSTIME register bit settings only apply when SPI<sub>n</sub> is operating in Master mode (SPI<sub>n</sub>_CTRL0.master = 1)</small>
7:0	pre	R/W	0	<b>Slave Select Delay to First SCK</b> Set the number of system clock cycles the Slave Select is held active prior to the first SCK edge. 0: 256 1: 1 2: 2 3:3 ... ... 254: 254 255: 255	<small>Note: The SPI<sub>n</sub>_SSTIME register bit settings only apply when SPI<sub>n</sub> is operating in Master mode (SPI<sub>n</sub>_CTRL0.master = 1)</small>

Table 12-14: SPI Master Clock Configuration Registers

SPI Master Clock Configuration				SPI <sub>n</sub> _CLKCTRL	[0x0014]
Bits	Name	Access	Reset	Description	
31:20	-	RO	0	Reserved	

SPI Master Clock Configuration				<b>SPI<sub>n</sub>_CLKCTRL</b>	[0x0014]
Bits	Name	Access	Reset	Description	
19:16	clkdiv	R/W	0	<b>SPI Peripheral Clock Scale</b> Scales the SPI input clock (PCLK) by $2^{\text{scale}}$ to generate the SPI <sub>n</sub> peripheral clock. $f_{\text{SPInCLK}} = \frac{f_{\text{SPIn\_INPUT\_CLK}}}{2^{\text{clkdiv}}}$ Valid values for scale are 0 to 8 inclusive. Values greater than 8 are reserved for future use. <i>Note: 1-bit and 9-bit character lengths are not supported.</i> <i>Note: If SPI<sub>n</sub>_CLKCTRL.clkdiv = 0, SPI<sub>n</sub>_CLKCTRL.hi = 0, and SPI<sub>n</sub>_CLKCTRL.lo = 0, character sizes of 2 and 10 bits are not supported.</i>	
15:8	hi	R/W	0	<b>SCK Hi Clock Cycles Control</b> 0: Hi duty cycle control disabled. Only valid if SPI <sub>n</sub> _CLK.clkdiv = 0. 1 - 15: The number of SPI <sub>n</sub> peripheral clocks, $f_{\text{SPInCLK}}$ , that SCK is high. <i>Note: 1-bit and 9-bit character lengths are not supported.</i> <i>Note: If SPI<sub>n</sub>_CLKCTRL.clkdiv = 0, SPI<sub>n</sub>_CLKCTRL.hi = 0, and SPI<sub>n</sub>_CLKCTRL.lo = 0, character sizes of 2 and 10 bits are not supported.</i>	
7:0	lo	R/W	0	<b>SCK Low Clock Cycles Control</b> This field controls the SCK low clock time and is used to control the overall SCK duty cycle in combination with the SPI <sub>n</sub> _CLK.hi field. 0: Low duty cycle control disabled. Setting this field to 0 is only valid if SPI <sub>n</sub> _CLK.clkdiv = 0. 1 to 15: The number of SPI <sub>n</sub> peripheral clocks, $f_{\text{SPInCLK}}$ , that the SCK signal is low. <i>Note: 1-bit and 9-bit character lengths are not supported.</i> <i>Note: If SPI<sub>n</sub>_CLKCTRL.clkdiv = 0, SPI<sub>n</sub>_CLKCTRL.hi = 0, and SPI<sub>n</sub>_CLKCTRL.lo = 0, character sizes of 2 and 10 bits are not supported.</i>	

Table 12-15: SPI DMA Control Registers

SPI DMA Control			SPI <sub>n</sub> _DMA		[0x001C]
Bits	Name	Access	Reset	Description	
31	dma_rx_en	R/W	0	<b>Receive DMA Enable</b> 0: Disabled. Any pending DMA requests are cleared 1: Enabled	
30:24	dma_rx_en	R	0	<b>Number of Bytes in the Receive FIFO</b> Read returns the number of bytes currently in the receive FIFO	
23	rx_flush	W1O	-	<b>Clear the Receive FIFO</b> 1: Clear the receive FIFO and any pending receive FIFO flags in SPI <sub>n</sub> _INTFL. This should be done when the receive FIFO is inactive. Writing a 0 has no effect.	
22	rx_fifo_en	R/W	0	<b>Receive FIFO Enabled</b> 0: Disabled 1: Enabled	
21	-	RO	0	<b>Reserved</b>	
20:16	rx_thd_val	R/W	0x00	<b>Receive FIFO Threshold Level</b> 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 by setting SPI <sub>n</sub> _DMA.dma_tx_en, and SPI <sub>n</sub> _INTFL.rx_thd_val becomes set. Valid values are 0 to 30. <i>Note: 31 is an invalid setting and reserved for future use.</i>	

SPI DMA Control		SPIn_DMA			[0x001C]
Bits	Name	Access	Reset	Description	
15	dma_tx_en	R/W	0	<b>Transmit DMA Enable</b> 0: Disabled. Any pending DMA requests are cleared 1: transmit DMA is enabled	
14:8	tx_lvl	R	0	<b>Number of Bytes in the Transmit FIFO</b> Read this field to determine the number of bytes currently in the transmit FIFO.	
7	tx_flush	R/W	0	<b>Transmit FIFO Clear</b> Set this bit to clear the transmit FIFO and all transmit FIFO flags in the <i>SPIn_INTFL</i> register. <i>Note:</i> The transmit FIFO should be disabled ( <i>SPIn_DMA.tx_fifo_en</i> = 0) prior to setting this field. <i>Note:</i> Setting this field to 0 has no effect.	
6	tx_fifo_en	R/W	0	<b>Transmit FIFO Enabled</b> 0: Disabled 1: Enabled	
5	-	RO	0	<b>Reserved</b>	
4:0	tx_thd_val	R/W	0x10	<b>Transmit FIFO Threshold Level</b> Set this value to the desired transmit FIFO threshold level. When the transmit FIFO count ( <i>SPIn_DMA.tx_lvl</i> ) falls below this value, a DMA request is triggered if enabled by setting <i>SPIn_DMA.dma_tx_en</i> and <i>SPIn_INTFL.tx_thd_val</i> becomes set.	

Table 12-16: SPI Interrupt Status Flags Registers

SPI Interrupt Status Flags			SPIn_INTFL		[0x0020]
Bits	Name	Access	Reset	Description	
31:16	-	RO	0	<b>Reserved</b>	
15	rx_un	R/W1C	0	<b>Receive FIFO Underrun Flag</b> Set when a read is attempted from an empty receive FIFO.	
14	rx_ov	R/W1C	0	<b>Receive FIFO Overrun Flag</b> Set if SPI is in Slave Mode, and a write to a full receive FIFO is attempted. If the SPI is in Master Mode, this bit is not set as the SPI stalls the clock until data is read from the receive FIFO.	
13	tx_un	R/W1C	0	<b>Transmit FIFO Underrun Flag</b> Set if SPI is in Slave Mode, and a read from empty transmit FIFO is attempted. If SPI is in Master Mode, this bit is not set as the SPI stalls the clock until data is written to the empty transmit FIFO.	
12	tx_ov	R/W1C	0	<b>Transmit FIFO Overrun Flag</b> Set when a write is attempted to a full transmit FIFO.	
11	mst_done	R/W1C	0	<b>Master Data Transmission Done Flag</b> Set if SPIn is in Master Mode, and all transactions have completed. <i>SPIn_CTRL1.tx_num_char</i> has been reached.	
10	-	RO	0	<b>Reserved</b>	
9	abort	R/W1C	0	<b>Slave Mode Transaction Abort Detected Flag</b> Set if the SPI is in Slave Mode, and SS is deasserted before a complete character is received.	

SPI Interrupt Status Flags				SPIn_INTFL	[0x0020]
Bits	Name	Access	Reset	Description	
8	fault	R/W1C	0	<b>Multi-Master Fault Flag</b> Set if the SPI is in Master Mode, Multi-Master Mode is enabled, and a Slave Select input is asserted. A collision also sets this flag.	
7:6	-	RO	0	<b>Reserved</b>	
5	ssd	R/W1C	0	<b>Slave Select Deasserted Flag</b>	
4	ssa	R/W1C	0	<b>Slave Select Asserted Flag</b>	
3	rx_full	R/W1C	0	<b>Receive FIFO Full Flag</b>	
2	rx_thd	R/W1C	0	<b>Receive FIFO Threshold Level Crossed Flag</b> Set when the receive FIFO exceeds the value in <i>SPIn_DMA.rx_fifo_level</i> . Cleared once receive FIFO level drops below <i>SPIn_DMA.rx_fifo_level</i> .	
1	tx_em	R/W1C	1	<b>Transmit FIFO Empty Flag</b> The transmit FIFO is empty.	
0	tx_thd	R/W1C	0	<b>Transmit FIFO Threshold Level Crossed Flag</b> Set when the transmit FIFO is less than the value in <i>SPIn_DMA.tx_fifo_level</i> . Cleared once transmit FIFO level exceeds <i>SPIn_DMA.tx_fifo_level</i> ,	

Table 12-17: SPI Interrupt Enable Registers

SPI Interrupt Enable				SPIn_INTEN	[0x0024]
Bits	Name	Access	Reset	Description	
31:16	-	RO	0	<b>Reserved</b>	
15	rx_un	R/W	0	<b>Receive FIFO Underrun Interrupt Enable</b> 0: Interrupt is disabled 1: Interrupt is enabled	
14	rx_ov	R/W	0	<b>Receive FIFO Overrun Interrupt Enable</b> 0: Interrupt is disabled 1: Interrupt is enabled	
13	tx_un	R/W	0	<b>Transmit FIFO Underrun Interrupt Enable</b> 0: Interrupt is disabled 1: Interrupt is enabled	
12	tx_ov	R/W	0	<b>Transmit FIFO Overrun Interrupt Enable</b> 0: Interrupt is disabled 1: Interrupt is enabled	
11	mst_done	R/W	0	<b>Master Data Transmission Done Interrupt Enable</b> 0: Interrupt is disabled 1: Interrupt is enabled	
10	-	RO	0	<b>Reserved</b>	
9	abort	R/W	0	<b>Slave Mode Abort Detected Interrupt Enable</b> 0: Interrupt is disabled 1: Interrupt is enabled	
8	fault	R/W	0	<b>Multi-Master Fault Interrupt Enable</b> 0: Interrupt is disabled 1: Interrupt is enabled	
7:6	-	RO	0	<b>Reserved</b>	
5	ssd	R/W	0	<b>Slave Select Deasserted Interrupt Enable</b> 0: Interrupt is disabled 1: Interrupt is enabled	

SPI Interrupt Enable			SPIn_INTEN		[0x0024]
Bits	Name	Access	Reset	Description	
4	ssa	R/W	0	<b>Slave Select Asserted Interrupt Enable</b> 0: Interrupt is disabled 1: Interrupt is enabled	
3	rx_full	R/W	0	<b>Receive FIFO Full Interrupt Enable</b> 0: Interrupt is disabled 1: Interrupt is enabled	
2	rx_thd	R/W	0	<b>Receive FIFO Threshold Level Crossed Interrupt Enable</b> 0: Interrupt is disabled 1: Interrupt is enabled	
1	tx_em	R/W	0	<b>Transmit FIFO Empty Interrupt Enable</b> 0: Interrupt is disabled 1: Interrupt is enabled	
0	tx_thd	R/W	0	<b>Transmit FIFO Threshold Level Crossed Interrupt Enable</b> 0: Interrupt is disabled 1: Interrupt is enabled	

Table 12-18: SPI Wakeup Status Flags Registers

SPI Wakeup Flags			SPIn_WKFL		[0x0028]
Bits	Name	Access	Reset	Description	
31:4	-	RO	0	<b>Reserved</b>	
3	rx_full	R/W1C	0	<b>Wake on Receive FIFO Full Flag</b> 0: Wake condition has not occurred. 1: Wake condition occurred.	
2	rx_thd	R/W1C	0	<b>Wake on Receive FIFO Threshold Level Crossed Flag</b> 0: Wake condition has not occurred. 1: Wake condition occurred.	
1	tx_em	R/W1C	0	<b>Wake on Transmit FIFO Empty Flag</b> 0: Wake condition has not occurred. 1: Wake condition occurred.	
0	tx_thd	R/W1C	0	<b>Wake on Transmit FIFO Threshold Level Crossed Flag</b> 0: Wake condition has not occurred. 1: Wake condition occurred.	

Table 12-19: SPI Wakeup Enable Registers

SPI Wakeup Enable			SPIn_WKEN		[0x002C]
Bits	Name	Access	Reset	Description	
31:4	-	RO	0	<b>Reserved</b>	
3	rx_full	R/W	0	<b>Wake on Receive FIFO Full Enable</b> 0: Wake event is disabled 1: Wake event is enabled.	
2	rx_thd	R/W	0	<b>Wake on Receive FIFO Threshold Level Crossed Enable</b> 0: Wake event is disabled 1: Wake event is enabled.	
1	tx_em	R/W	0	<b>Wake on Transmit FIFO Empty Enable</b> 0: Wake event is disabled 1: Wake event is enabled.	
0	tx_thd	R/W	0	<b>Wake on Transmit FIFO Threshold Level Crossed Enable</b> 0: Wake event is disabled 1: Wake event is enabled.	

Table 12-20: SPI Slave Select Timing Registers

SPI Status			SPln_STAT		[0x0030]
Bits	Name	Access	Reset	Description	
31:1	-	RO	0	Reserved	
0	busy	R	0	<p><b>SPI Active Status</b>            SPI status flag, see value descriptions for details of each value.</p> <p>0: SPln is not active. In master mode, the <i>busy</i> flag is cleared when the last character is sent. In slave mode, the <i>busy</i> field is cleared when the configured slave select input is deasserted.</p> <p>1: SPln is active. In master mode, the <i>busy</i> flag is set when a transaction starts. In slave mode, the <i>busy</i> flag is set when a configured slave select input is asserted.</p> <p><i>Note:</i> <i>SPln_CTRL0</i>, <i>SPln_CTRL1</i>, <i>SPln_CTRL2</i>, <i>SPln_SSTIME</i>, and <i>SPln_CLKCTRL</i> should not be configured if this bit is set.</p>	

## 13. I<sup>2</sup>C Master/Slave Serial Communications Peripheral (I<sup>2</sup>C)

The I<sup>2</sup>C peripherals can be configured as either an I<sup>2</sup>C master or an I<sup>2</sup>C slave at standard data rates. For simplicity, I<sup>2</sup>Cn is used throughout this section to refer to any of the I<sup>2</sup>C peripherals.

For detailed information on I<sup>2</sup>C bus operation, refer to Maxim Application Note 4024 "SPI/I<sup>2</sup>C Bus Lines Control Multiple Peripherals" at <https://www.maximintegrated.com/en/app-notes/index.mvp/id/4024>

### 13.1 I<sup>2</sup>C Master/Slave Features

Each I<sup>2</sup>C master/slave is compliant with the I<sup>2</sup>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 master or slave device as a transmitter or receiver
- Supports I<sup>2</sup>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-master systems, including support for arbitration and clock synchronization for Standard, Fast, and Fast Plus modes
- Supports 7- and 10-bit addressing
- Supports RESTART condition
- Supports clock stretching
- Provides transfer status interrupts and flags
- Provides DMA data transfer support
- Supports I<sup>2</sup>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

### 13.2 Instances

The three instances of the peripheral are shown in *Table 13-1* lists the locations of the SDA and SCL signals for each of the I<sup>2</sup>Cn peripherals per package.

*Table 13-1: MAX78000 I<sup>2</sup>C Peripheral Pins*

I <sup>2</sup> C Instance	Alternate Function	Alternate Function #	Package 81-CTBGA
I <sup>2</sup> C0	I2C0_SCL	AF1	P0.10
	I2C0_SDA	AF1	P0.11
I <sup>2</sup> C1	I2C1_SCL	AF1	P0.16
	I2C1_SDA	AF1	P0.17
I <sup>2</sup> C2	I2C2_SCL	AF1	P0.30
	I2C2_SDA	AF1	P0.31

## 13.3 I<sup>2</sup>C Overview

### 13.3.1 I<sup>2</sup>C Bus Terminology

*Table 13-2* contains terms and definitions used in this chapter for the I<sup>2</sup>C bus terminology.

*Table 13-2: I<sup>2</sup>C Bus Terminology*

Term	Definition
Transmitter	The device that sends data to the bus.
Receiver	The device that receives data from the bus.
Master	The device that initiates a transfer, generates clock signals, and terminates a transfer.
Slave	The device addressed by a master.
Multi-master	More than one master can attempt to control the bus at the same time without corrupting the message.
Arbitration	Procedure to ensure that, if more than one master simultaneously tries to control the bus, only one can do so, and the resulting message is not corrupted.
Synchronization	Procedure to synchronize the clock signals of two or more devices.
Clock Stretching	When a slave device holds SCL low to pause a transfer until it is ready. This feature is optional according to the I <sup>2</sup> C Specification; thus, a master does not have to support slave clock stretching if none of the slaves in the system are capable of clock stretching.

### 13.3.2 I<sup>2</sup>C Transfer Protocol Operation

The I<sup>2</sup>C protocol operates over a two-wire bus: a clock circuit (SCL) and a data circuit (SDA). I<sup>2</sup>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 master sends a START or repeated START condition. It is followed by the I<sup>2</sup>C slave address of the targeted slave device plus a read/write bit. The master can transmit data to the slave (a 'write' operation) or receive data from the slave (a 'read' operation). Information is sent most significant bit (MSB) first. Following the slave address, the master indicates a read or write operation and then exchanges data with the addressed slave. An acknowledge bit is sent by the receiving device after each byte is transferred. When all necessary data bytes have been transferred, a STOP or RESTART condition is sent by the bus master to indicate the end of the transaction. After the STOP condition has been sent, the bus is idle and ready for the next transaction. After a RESTART condition is sent, the same master begins a new transmission. The number of bytes that can be transmitted per transfer is unrestricted.

### 13.3.3 START and STOP Conditions

A START condition occurs when a bus master pulls SDA from high to low while SCL is high, and a STOP condition occurs when a bus master allows SDA to be pulled from low to high while SCL is high. Because these are unique conditions that cannot occur during normal data transfer, they are used to denote the beginning and end of the data transfer.

### 13.3.4 Master Operation

I<sup>2</sup>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 slave sends a NACK in response to a write operation, the I<sup>2</sup>C master generates an interrupt. The I<sup>2</sup>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<sup>2</sup>C master stops the clock to allow time to read bytes from the receive FIFO or load bytes into the transmit FIFO.

### 13.3.5 Acknowledge and Not Acknowledge

An acknowledge bit (ACK) is generated by the receiver, whether I<sup>2</sup>C master or slave, after every byte received by pulling SDA low. The ACK bit is how the receiver tells the transmitter that the byte was successfully received, and another byte might be sent.

A Not Acknowledge (NACK) occurs if the receiver does not generate an ACK when the transmitter releases SDA. A NACK is generated by allowing SDA to float high during the acknowledge time slot. The I<sup>2</sup>C master 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.

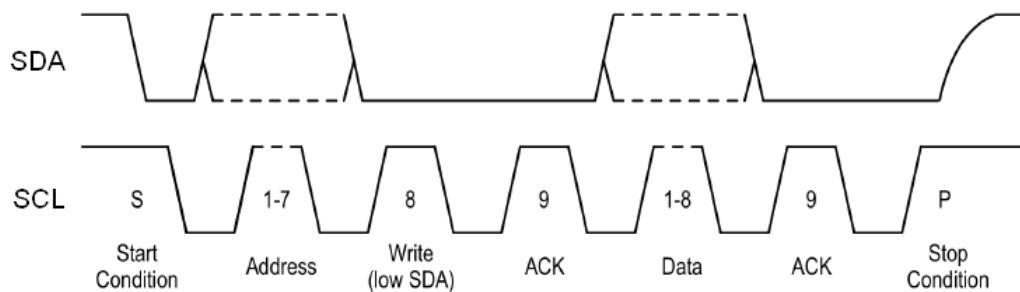
A receiver can generate a NACK after a byte transfer if any of the following conditions occur:

- No receiver is present on the bus with the transmitted address. In that case, no device responds with an acknowledge signal.
- The receiver is unable to receive or transmit because it is busy and is not ready to start communication with the master.
- During the transfer, the receiver receives data or commands it does not understand.
- During the transfer, the receiver is unable to receive any more data.
- If an I<sup>2</sup>C master has requested data from a slave, it signals the slave to stop transmitting by sending a NACK following the last byte it requires.

### **13.3.6 Bit Transfer Process**

Both SDA and SCL circuits are open-drain, bidirectional circuits. Each requires an external pullup resistor that ensures each circuit is high when idle. The I<sup>2</sup>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 13-1*.

*Figure 13-1: I<sup>2</sup>C Write Data Transfer*



An example of an I<sup>2</sup>C data transfer is as follows:

1. A bus master indicates a data transfer to a slave with a START condition.
2. The master then transmits one byte with a 7-bit slave address and a single read-write bit: a zero for a write or a one for a read.
3. During the next SCL clock following the read-write bit, the master releases SDA. During this clock period, the addressed slave responds with an ACK by pulling SDA low.
4. The master senses the ACK condition and begins transferring data. If reading from the slave, it floats SDA and allows the slave to drive SDA to send data. After each byte, the master drives SDA low to acknowledge the byte. If writing to the slave, the master drives data on the SDA circuit for each of the eight bits of the byte and then floats SDA during the ninth bit to allow the slave to reply with the ACK indication.
5. After the last byte is transferred, the master indicates the transfer is complete by generating a STOP condition. A STOP condition is generated when the master pulls SDA from a low to high while SCL is high.

## 13.4 Configuration and Usage

### 13.4.1 SCL and SDA Bus Drivers

SCL and SDA are open-drain signals. In this device, once the I<sup>2</sup>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<sup>2</sup>C slave device can hold SCL low, such as for clock stretching. Push-pull operation is enabled by setting `I2Cn_CTRL.sclppm` to 1. SDA, on the other hand, always operates in open-drain mode.

### 13.4.2 SCL Clock Configurations

The SCL frequency is dependent upon the values of the I<sup>2</sup>C's 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.*

### 13.4.3 SCL Clock Generation for Standard, Fast and Fast-Plus Modes

The master generates the I<sup>2</sup>C clock on the SCL line. When operating as a master, the software must configure the `I2Cn_CLKHI` and `I2Cn_CLKLO` registers for the desired I<sup>2</sup>C operating frequency.

The SCL high time is configured in the I<sup>2</sup>C Clock High Time register field `I2Cn_CLKHI.hi` using [Equation 13-2](#). The SCL low time is configured in the I<sup>2</sup>C Clock Low Time register field `I2Cn_CLKLO.lo` using [Equation 13-3](#). Each of these fields is 8-bits. The I<sup>2</sup>C frequency value is shown in [Equation 13-1](#).

**Equation 13-1: I<sup>2</sup>C Clock Frequency**

$$f_{I2C\_CLK} = \frac{1}{t_{I2C\_CLK}} \text{ is either } f_{PCLK} \text{ or } f_{IBRO}$$

**Equation 13-2: I<sup>2</sup>C Clock High Time Calculation**

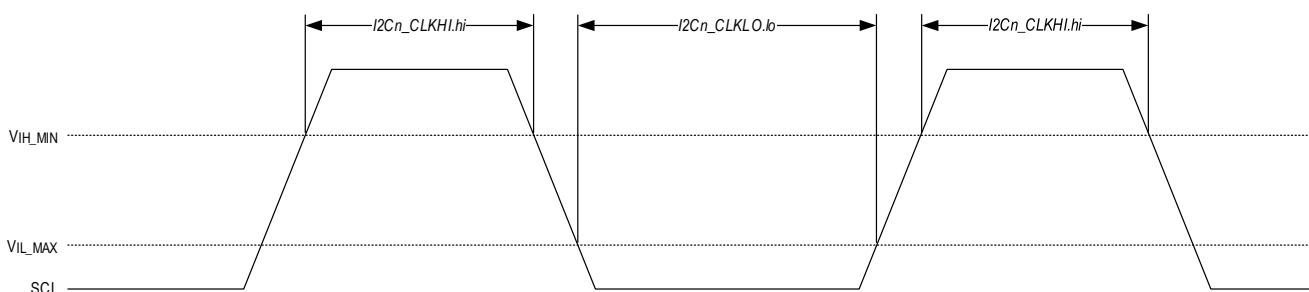
$$t_{SCL\_HI} = t_{I2C\_CLK} \times (I2Cn_CLKHI.hi + 1)$$

**Equation 13-3: I<sup>2</sup>C Clock Low Time Calculation**

$$t_{SCL\_LO} = t_{I2C\_CLK} \times (I2Cn_CLKLO.lo + 1)$$

[Figure 13-2](#) shows the association between the SCL clock low and high times for Standard, Fast, and Fast Plus I<sup>2</sup>C frequencies.

[Figure 13-2: I<sup>2</sup>C SCL Timing for Standard, Fast and Fast-Plus Modes](#)



During synchronization, external masters or external slaves may be driving SCL simultaneously. This affects the SCL duty cycle. By monitoring SCL, the controller can determine whether an external master or slave is holding SCL low. In either case, the controller waits until SCL is high before starting to count the number of SCL high cycles. Similarly, if an external

master pulls SCL low before the controller has finished counting SCL high cycles, then the controller starts counting SCL low cycles and releases SCL once the time period, *I2Cn\_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.

#### **13.4.4 SCL Clock Generation for Hs-mode**

To operate the I<sup>2</sup>C interface in Hs-mode at its maximum speed (~3.4MHz), values to be programmed into the *I2Cn\_HSCLK.hsclk\_lo* register and *I2Cn\_HSCLK.hsclk\_hi* register must be determined. Since the Hs-mode operation is entered by first using one of the lower speed modes for pre-amble, a relevant lower speed mode must also be configured. See *SCL Clock Generation for Standard, Fast and Fast-Plus Modes* for information regarding the configuration of lower speed modes.

##### **13.4.4.1 Hs-Mode Timing**

With I<sup>2</sup>C bus capacitances less than 100pf, the following specifications are extracted from the I<sup>2</sup>C-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 I<sup>2</sup>C bus clock.

$t_{HIGH\_MIN}$  = 60ns, the minimum high time for the I<sup>2</sup>C bus clock.

$t_{rCL\_MAX}$  = 40ns, the maximum rise time of the I<sup>2</sup>C bus clock.

$t_{fCL\_MAX}$  = 40ns, the maximum fall time of the I<sup>2</sup>C bus clock.

##### **13.4.4.2 Hs-Mode Clock Configuration**

The maximum Hs-mode bus clock frequency can now be determined. The system clock frequency,  $f_{SYS\_CLK}$ , must be known. Hs-mode timing information from *Hs-Mode Timing* must be used.

*Equation 13-4: I<sup>2</sup>C Target SCL Frequency*

$$\text{Desired Target Maximum I}^2\text{C Frequency: } f_{SCL} = \frac{1}{t_{SCL}}.$$

In Hs-mode, the analog glitch filter (AF\_MIN) within the device adds a minimum delay of  $t_{AF\_MIN}$  = 10ns.

*Equation 13-5: Determining the *I2Cn\_HSCLK.hsclk\_lo* Register Value*

$$I2Cn_{HS\_CLK}.hsclk\_lo = \text{MAX} \left\{ \left\lfloor \left( \frac{t_{LOW\_MIN} + t_{FCL\_MAX} + t_{I2C\_CLK} - t_{AF\_MIN}}{t_{I2C\_CLK}} \right) \right\rfloor - 1, \quad \frac{t_{SCL}}{t_{I2C\_CLK}} - 1 \right\}$$

*Equation 13-6: Determining the *I2Cn\_HSCLK.hsclk\_hi* Register Value*

$$I2Cn_{HS\_CLK}.hsclk\_hi = \left\lfloor \left( \frac{t_{HIGH\_MIN} + t_{rCL\_MAX} + t_{I2C\_CLK} - t_{AF\_MIN}}{t_{I2C\_CLK}} \right) \right\rfloor - 1$$

*Equation 13-7: The Calculated Frequency of the I<sup>2</sup>C Bus Clock Using the Results of Equation 13-5 and Equation 13-6*

$$\text{Calculated Frequency} = ((I2Cn_{HS\_CLK}.hsclk\_hi + 1) + (I2Cn_{HS\_CLK}.hsclk\_lo + 1)) * t_{I2C\_CLK}$$

*Table 13-3* shows the I<sup>2</sup>C bus clock calculated frequencies given different  $f_{SYS\_CLK}$  frequencies.

*Table 13-3: Calculated I<sup>2</sup>C Bus Clock Frequencies*

$f_{SYS\_CLK}$ (MHz)	<i>I2Cn_HSCLK.hsclk_hi</i>	<i>I2Cn_HSCLK.hsclk_lo</i>	Calculated Frequency (MHz)
100	4	9	3.3

<i>fsys_CLK</i> (MHz)	<i>I2Cn_HSCLK.hsclk_hi</i>	<i>I2Cn_HSCLK.hsclk_lo</i>	Calculated Frequency (MHz)
50	2	4	3.125
25	1	2	2.5

### 13.4.5 Addressing

After a START condition, the I<sup>2</sup>C slave address byte is transmitted by the hardware. The I<sup>2</sup>C slave address is composed of a slave address followed by a read/write bit.

Table 13-4: I<sup>2</sup>C Slave Address Format

Slave Address Bits		R/W Bit	Description
0000	000	0	General Call Address
0000	000	1	START Condition
0000	001	x	CBUS Address
0000	010	x	Reserved for different bus format
0000	011	x	Reserved for future purposes
0000	1xx	x	HS-mode master code
1111	1xx	x	Reserved for future purposes
1111	0xx	x	10-bit slave addressing

In 7-bit addressing mode, the master sends one address byte. To address a 7-bit address slave, first, clear the *I2Cn\_MSTCTRL.ex\_addr\_en* field to 0, then write the address to the transmit FIFO formatted as follows where An is address A6:A0.

Master writing to slave: 7-bit address : [A6 A5 A4 A3 A2 A1 A0 0]

Master reading from slave: 7-bit address : [A6 A5 A4 A3 A2 A1 A0 1]

In 10-bit addressing mode (*I2Cn\_MSTCTRL.ex\_addr\_en* = 1), the first byte the master sends is the 10-bit slave address byte, which includes the first two bits of the 10-bit address, followed by a 0 for the R/W bit. That is followed by a second byte representing the remainder of the 10-bit address. If the operation is a write, this is followed by data bytes to be written to the slave. If the operation is a read, it is followed by a repeated START. The software then writes the 10-bit address again with a 1 for the R/W bit. This I<sup>2</sup>C then starts receiving data from the slave device.

### 13.4.6 Master Mode Operation

The peripheral operates in master mode when master mode Enable *I2Cn\_CTRL.mst\_mode* = 1. To initiate a transfer, the master 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 master can communicate with multiple slave devices without relinquishing the bus. Instead of generating a STOP condition after communicating with the first slave, the master generates a Repeated START condition, or RESTART, by setting *I2Cn\_MSTCTRL.restart* = 1. If a transaction is in progress, the peripheral finishes the transaction before generating a RESTART. The peripheral then transmits the slave address stored in the transmit FIFO. The *I2Cn\_MSTCTRL.restart* bit is automatically cleared to 0 as soon as the master begins a RESTART condition.

*I2Cn\_MSTCTRL.start* is automatically cleared to 0 after the master has completed a transaction and sent a STOP condition.

The master can also generate a STOP condition by setting *I2Cn\_MSTCTRL.stop* = 1.

If both START and RESTART conditions are enabled at the same time, a START condition is generated first. Then, at the end of the first transaction, a RESTART condition is generated.

If both RESTART and STOP conditions are enabled at the same time, a STOP condition is not generated. Instead, a RESTART condition is generated. After the RESTART condition is generated, both bits are cleared.

If START, RESTART, and STOP are all enabled at the same time, a START condition is first generated. At the end of the first transaction, a RESTART condition is generated. The *I2Cn\_MSTCTRL.stop* bit is cleared and ignored.

A slave cannot generate START, RESTART, or STOP conditions. Therefore, when master mode is disabled, the *I2Cn\_MSTCTRL.start*, *I2Cn\_MSTCTRL.restart*, and *I2Cn\_MSTCTRL.stop* bits are all cleared to 0.

For master mode operation, the following registers should only be configured when either:

1. The I<sup>2</sup>C peripheral is disabled,  
or
2. The I<sup>2</sup>C bus is guaranteed to be idle/free.

If this peripheral is the only master on the bus, then changing the registers outside of a transaction (*I2Cn\_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\_MSTCTRL.mcode*
- *I2Cn\_CLKLO.lo*
- *I2Cn\_CLKHI.hi*
- *I2Cn\_HSCLK.hsclk\_lo*
- *I2Cn\_HSCLK.hsclk\_hi*

In contrast to the above set of registers, these registers below can be safely (re)programmed at any time:

- All interrupt flags and interrupt enable bits
- *I2Cn\_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*

### **13.4.6.1 I<sup>2</sup>C Master Mode Receiver Operation**

When in master mode, initiating a master receiver operation begins with the following sequence:

1. Write the number of data bytes to receive to the I<sup>2</sup>C receive count field (*I2Cn\_RXCTRL1.cnt*).
2. Write the I<sup>2</sup>C slave 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 slave address is transmitted by the controller from the *I2Cn\_FIFO* register.
5. The I<sup>2</sup>C controller receives an ACK from the slave, and the controller sets the address ACK interrupt flag (*I2Cn\_INTF0.addr\_ack* = 1).
6. The I<sup>2</sup>C controller receives data from the slave 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 I<sup>2</sup>C controller sends a NACK to the slave and sets the Transfer Done Interrupt Status Flag (*I2Cn\_INTF0.done* = 1).
8. If *I2Cn\_MSTCTRL.restart* or *I2Cn\_M.stop* is set, then the I<sup>2</sup>C controller sends a repeated START or STOP, respectively.

### **13.4.6.2 I<sup>2</sup>C Master Mode Transmitter Operation**

When in master mode, initiating a master transmitter operation begins with the following sequence:

1. Write the I<sup>2</sup>C slave address byte to the *I2Cn\_FIFO* register with the R/W bit set to 0
2. Write the desired data bytes to the *I2Cn\_FIFO* register, up to the size of the transmit FIFO. (e.g., If the transmit FIFO size is 8 bytes, the software may write one address byte and seven data bytes prior to starting the transaction.)
3. Send a START condition by setting *I2Cn\_MSTCTRL.start* = 1
4. The controller transmits the slave address byte written to the *I2Cn\_FIFO* register.
5. The I<sup>2</sup>C controller receives an ACK from the slave, and the controller sets the address ACK interrupt flag (*I2Cn\_INTF0.addr\_ack* = 1).
6. The *I2Cn\_FIFO* register data bytes are transmitted on the SDA line.
  - a. The I<sup>2</sup>C controller receives an ACK from the slave 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 will pause at the beginning of the byte and wait 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, the hardware sets *I2Cn\_INTF0.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.

### **13.4.6.3 I<sup>2</sup>C Multi-Master Operation**

The I<sup>2</sup>C protocol supports multiple masters on the same bus. When the bus is free, it is possible that two (or more) masters might try to initiate communication at the same time. This is a valid bus condition. If this occurs, and the two masters want to transmit different data or address different slaves, only one master can remain in master mode and complete its transaction. The other master must stop transmission and wait until the bus is idle. This process by which the winning master is determined is called bus arbitration.

To determine which master wins the arbitration, for each address or data bit, the master compares the data being transmitted on SDA to the value observed on SDA. If a master attempts to transmit a 1 on SDA (that is, the master lets SDA float) but senses a 0 instead, then that master loses arbitration, and the other master that sent a zero continues with the transaction. The losing master cedes the bus by switching off its SDA and SCL drivers.

*Note: This arbitration scheme works with any number of bus masters: if more than two masters begin transmitting simultaneously, the arbitration continues as each master cedes the bus until only one master remains transmitting. Data is not corrupted because as soon as each master realizes it has lost the arbitration, it stops transmitting on SDA, leaving the following data bits sent on SDA intact.*

If the I<sup>2</sup>C master peripheral detects it has lost the arbitration, it stops generating SCL; sets *I2Cn\_INTFLO.areri*; sets *I2Cn\_INTFLO.tx\_lockout*, flushing any remaining data in the transmit FIFO; and clears *I2Cn\_MSTCTRL.start*, *I2Cn\_MSTCTRL.restart*, and *I2Cn\_MSTCTRL.stop* to 0. So long as the peripheral is not itself addressed by the winning master, the I<sup>2</sup>C peripheral stays in master mode (*I2Cn\_CTRL.mst\_mode* = 1). If at any time another master addresses this peripheral using the address programmed in *I2Cn\_SLAVE.addr*, then the I<sup>2</sup>C peripheral clears *I2Cn\_CTRL.mst\_mode* to 0 and begins responding as a slave. This can even occur during the same address transmission during which the peripheral lost arbitration.

*Note: Arbitration loss is considered an error condition, and like the other error conditions, it sets *I2Cn\_INTFLO.tx\_lockout* to 1. Therefore, after an arbitration loss, the software needs to clear *I2Cn\_INTFLO.tx\_lockout* and reload the transmit FIFO.*

Also, in a multi-master 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 will:

1. Wait for the other master to complete the transaction(s) by sending a STOP,
2. Count out the bus free time using  $t_{BUF} = t_{SCL\_LO}$  (see *Equation 13-3*), and then
3. Send a START condition and begin transmitting the slave address byte(s) in the transmit FIFO, followed by the rest of the transfer.

The I<sup>2</sup>C master peripheral is compliant with all bus arbitration and clock synchronization requirements of the I<sup>2</sup>C specification; this operation is automatic, and no additional programming is required.

#### 13.4.7 Slave Mode Operation

When in slave mode, the I<sup>2</sup>Cn peripheral operates as a slave device on the I<sup>2</sup>C bus and responds to an external master's requests to transmit or receive data. To configure the I<sup>2</sup>Cn peripheral as a slave, write the *I2Cn\_CTRL.mst\_mode* bit to zero. The I<sup>2</sup>C clock is driven by the master on the bus so that the SCL device pin is driven by the external master, and *I2Cn\_STATUS.mst\_busy* remains a zero. The desired slave address must be set by writing to the *I2Cn\_SLAVE.addr* register.

For slave mode operation, the following register fields should be configured with the I<sup>2</sup>Cn peripheral disabled:

- *I2Cn\_CTRL.mst\_mode* = 0 for slave operation.
- *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 slave 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*
  - ◆ Recommended value is 0 for applications that can tolerate slave clock stretching (*I2Cn\_CTRL.clkstr\_dis* = 0).
  - ◆ Recommend value is 1 for applications that do not allow slave clock stretching (*I2Cn\_CTRL.clkstr\_dis* = 1).
- *I2Cn\_CLKHI.hi*
  - ◆ Applies to slave 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 13-2* is  $\geq t_{SU;DAT(\min)}$
- *I2Cn\_HSCLK.hsclk\_hi*
  - ◆ Applies to slave 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 13-6* is  $\geq t_{SU;DAT(\min)}$
- *I2Cn\_SLAVE.addr*
- *I2Cn\_SLAVE.ext\_addr\_en*

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\_lv*
  - ◆ Transmit and receive FIFO Threshold Levels
- *I2Cn\_TXCTRL1.tx\_rdy*
  - ◆ Transmit Ready (Can only be cleared by the hardware)
- *I2Cn\_TIMEOUT.scl\_to\_val*
  - ◆ Time Out Control
- *I2Cn\_DMA.rx\_en* and *I2Cn\_DMA.tx\_en*
  - ◆ Transmit and receive DMA Enables
- *I2Cn\_FIFO.data*
  - ◆ FIFO access register

#### 13.4.7.1 Slave Transmitter

The device will operate as a slave transmitter when the received address matches the device slave address with the R/W bit set to 1. The master is then reading from the device slave. There two main modes of slave transmitter operation: just-in-time mode and preload mode.

In just-in-time mode, the software waits to write the transmit data to the transmit FIFO until after the master addresses it for a READ transaction, "just in time" for the data to be sent to the master. This allows the software to defer the determination of what data should be sent until the time of the address match. As an example, the transmit data could be based on an immediately preceding I<sup>2</sup>C WRITE transaction that requests a certain block of data to be sent, or the data could represent the latest, most up-to-date value of a sensor reading. Clock stretching *must* be enabled (*I2Cn\_CTRL.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 specifically for this mode `I2Cn_CTRL.clkstr_dis` = 0, `I2Cn_TXCTRL0[5:2]` = 0x8 and `I2Cn_TXCTRL0.preload_mode` = 0. Don't forget to program `I2Cn_CLKHI.hi` and `I2Cn_HSCLK.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 will respond to its address with an ACK.
  - b. When the address match occurs, the hardware will set `I2Cn_INTFLO.addr_match` and `I2Cn_INTFLO.tx_lockout`.
3. The software waits for `I2Cn_INTFLO.addr_match` = 1, either through polling the interrupt flag or setting `I2Cn_INTENO.addr_match` to interrupt the CPU.
4. After reading `I2Cn_INTFLO.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. At this point, the hardware will hold SCL low until the software clears `I2Cn_INTFLO.tx_lockout` and loads data into the FIFO.
5. The software clears `I2Cn_INTFLO.addr_match` and `I2Cn_INTFLO.tx_lockout`. Now that `I2Cn_INTFLO.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 will release SCL (after counting out `I2Cn_CLKHI.hi`) and send out the data on the bus.
7. While the master keeps requesting data and sending ACKs, `I2Cn_INTFLO.ddone` will remain 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_INTENO.tx_thd` interrupt.
  - b. If the transmit FIFO ever empties during the transaction, the hardware will start clock stretching and wait for it to be refilled.
8. The master ends the transaction by sending a NACK. Once this happens, the `I2Cn_INTFLO.done` interrupt flag is set, and the software can stop monitoring the transmit FIFO.
9. The transaction is complete. The software should "clean up", including clearing `I2Cn_INTFLO.done` and clearing `I2Cn_INTENO.tx_thd` interrupt. Return to step 3, waiting on an address match.
10. If the software needs to know how many data bytes were transmitted to the master, it should check the transmit FIFO level as soon as the software sees `I2Cn_INTFLO.done` = 1 and use that to determine how many data bytes were successfully sent.
  - a. *Note that any data remaining in the transmit FIFO is discarded prior to the next transmit operation; it is NOT necessary for the software to manually flush the transmit FIFO for this to occur.*

The other mode of operation for slave transmit is preload mode. In this mode, it is assumed that the software knows prior to the transmit operation what data it should send to the master. 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 slave transmit preload mode:

1. With `I2Cn_CTRL.en` = 0, initialize all relevant registers, including specifically for this mode `I2Cn_CTRL.cl_clk_stretch_dis` = 1, `I2Cn_TXCTRL0[5:2]` = 0xF and `I2Cn_TXCTRL0.preload_mode` = 1.
2. The software sets `I2Cn_CTRL.en` = 1.
  - a. Even though the controller is enabled, at this point, it will not ACK an address match with R/W=1 until the software sets `I2Cn_TXCTRL1.preload_rdy` = 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_INTENO.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 *prior* to setting `I2Cn_TXCTRL1.preload_rdy` = 1.
  - b. If clock stretching is enabled, then an empty transmit FIFO will 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 will let the transaction complete as quickly as possible.
4. Once the software has prepared for the transmit operation; the software should set `I2Cn_TXCTRL1.preload_rdy` = 1.
  - a. The controller is now fully enabled and will respond with an ACK to an address match.
  - b. The hardware will set `I2Cn_INTFLO.addr_match` once an address match has occurred. `I2Cn_INTFLO.tx_lockout` will NOT be set and will remain 0.
5. The software waits for `I2Cn_INTFLO.addr_match` = 1, either through polling the interrupt flag or setting `I2Cn_INTENO.amie` to interrupt the CPU.
6. After seeing `I2Cn_INTFLO.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 `I2Cn_CTRL.read`, indicating transmit:
  - a. At this point, the hardware will begin sending out the data that was preloaded into the transmit FIFO.
  - b. Once the first data byte is sent, the hardware will also automatically clear `I2Cn_TXCTRL1.preload_rdy` to 0.
7. While the master keeps requesting data and sending ACKs, `I2Cn_INTFLO.done` will remain 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_INTENO.tx_thd` interrupt.
  - b. If clock stretching is disabled and the transmit FIFO ever empties during the transaction, the hardware will set `I2Cn_INTF1.tx_un` = 1 and send 0xFF for all following data bytes requested by the master.
8. The master ends the transaction by sending a NACK. Once this happens, the `I2Cn_INTFLO.done` interrupt flag is set.
  - a. If the transmit FIFO goes empty at the same time that the master indicates the transaction is complete by sending a NACK, this is not considered an underrun event, and the `I2Cn_INTF1.tx_un` flag will remain 0.
9. The transaction is complete, the software should "clean up", which should include clearing `I2Cn_INTFLO.done`. Return to step 3 and prepare for the next transaction.
  - a. If the software needs to know how many data bytes were transmitted to the master, it should check the transmit FIFO level as soon as the software sees `I2Cn_INTFLO.done` = 1 and use that to determine how many data bytes were successfully sent.
  - b. By default, any data remaining in the transmit FIFO is NOT discarded and instead is reused for the next transmit operation.
  - c. If this is not desired, the software can flush the transmit FIFO. The safest way to do this is by clearing and then resetting `I2Cn_CTRL.en`. This will flush both the transmit and receive FIFOs.

Once a slave starts transmitting out of its *I2Cn\_FIFO*, detection of an out of sequence STOP, START, or RESTART condition terminates the current transaction. When a transaction is terminated in such a manner, *I2Cn\_INTFLO.start\_err* or *I2Cn\_INTFLO.stop\_err* is set to 1.

If the transmit FIFO is not ready (*I2Cn\_TXCTRL1.preload\_rdy* = 0) and the I<sup>2</sup>C controller receives a data read request from the master, the hardware automatically sends a NACK at the end of the first address byte. The setting of the do not respond field is ignored by the hardware in this case because the only opportunity to send a NACK for an I<sup>2</sup>C read transaction is after the address byte.

### 13.4.7.2 Slave Receivers

The device operates as a slave receiver when the received address matches the device slave address with the R/W bit set to 0. The external master is writing to the slave.

Program flow for a receive operation is as follows:

1. With *I2Cn\_CTRL.en* = 0, initialize all relevant registers.
2. Set *I2Cn\_CTRL.en* = 1.
  - a. If an address match with R/W=0 occurs, and the receive FIFO is empty, the peripheral responds with an ACK, and the *I2Cn\_INTFLO.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\_INTFLO.addr\_match* = 1, either by polling or by enabling the *wr\_addr\_match* interrupt to the CPU. Once a successful address match occurs, the hardware sets *I2Cn\_INTFLO.addr\_match* =1.
4. Read *I2Cn\_CTRL.read* to determine whether the transaction is a transmit (*I2Cn\_CTRL.read* = 1) or receive (*I2Cn\_CTRL.read* = 0) operation. In this case, assume *I2Cn\_CTRL.read* = 0, indicating receive. At this point, the device begins receiving data into the receive FIFO.
5. Clear *I2Cn\_INTFLO.addr\_match*, and while the master keeps sending data, *I2Cn\_INTFLO.done* remains 0, and 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\_INTFLO.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 for the software to read from the receive FIFO, **or**
    - ii. If *I2Cn\_CTRL.clkstr\_dis* = 1, respond to the master with a NACK, and the last byte is discarded.
6. The master ends the transaction by sending a RESTART or STOP. Once this happens, the *I2Cn\_INTFLO.done* interrupt flag is set, and software can stop monitoring the receive FIFO.
7. Once a slave starts receiving into its receive FIFO, detection of an out of sequence STOP, START, or RESTART condition releases the I<sup>2</sup>C bus to the Idle state, and the hardware sets the *I2Cn\_INTFLO.start\_err* field or *I2Cn\_INTFLO.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 that a master 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 master, the software can control whether the NACK is sent with the initial address match or if instead, it is sent at the end of the first data byte. Setting *I2Cn\_RXCTRL0.dnr* to 1 chooses the former while setting *I2Cn\_RXCTRL0.dnr* to 0 chooses the latter.

### 13.4.8 Interrupt Sources

The I<sup>2</sup>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<sup>2</sup>C interrupt, the software determines the cause of the interrupt by reading the I<sup>2</sup>C Interrupt Flags registers [I2Cn\\_INTFL0](#) and [I2Cn\\_INTFL1](#). Interrupts can be generated for the following events:

- In either master or slave mode:
  - ◆ Transaction complete
  - ◆ Transaction timeout
  - ◆ FIFO is empty, not empty, and full to a configurable threshold level
  - ◆ Transmit FIFO lockout during a FIFO flush
  - ◆ Out of sequence START and STOP conditions
- In master mode:
  - ◆ Address ACK or NACK received from the slave
  - ◆ Data NACK received from the slave
  - ◆ Lost arbitration
- In slave mode:
  - ◆ Sent a NACK to an external master because the transmit or receive FIFO was not ready
  - ◆ Incoming address match
  - ◆ Transmit FIFO underflow or receive FIFO overflow

Interrupts for each event can be enabled or disabled by setting or clearing the corresponding bit in the [I2Cn\\_INTENO](#) or [I2Cn\\_INTEN1](#) interrupt enable registers.

*Note: Disabling the interrupt does not prevent the corresponding flag from being set by the hardware but does prevent an IRQ when the interrupt flag is set.*

*Note: Prior to enabling an interrupt, the status of the corresponding interrupt flag should be checked and, if necessary, serviced or cleared. This prevents a previous interrupt event from interfering with a new I<sup>2</sup>C communications session.*

### 13.4.9 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. Writing to a full transmit FIFO has no effect. Reads from [I2Cn\\_FIFO](#) dequeue data from the receive FIFO. Writing to a full transmit FIFO has no effect, and reading from an empty receive FIFO returns 0xFF.

The transmit and receive FIFO only reads or writes one byte at a time. Transactions larger 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 will have the valid data in the lowest 8 bits and 0's in the upper bits. In any case, the transmit and receive FIFOs only accepts 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 section [DMA Control](#) for more information on configuring the DMA.

During a receive transaction (which during master operation is a READ, and during slave operation is a WRITE), received bytes are automatically written to the 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 master mode transaction, then the controller sets the [I2Cn\\_INTFL1.rx\\_ov](#) or the [I2Cn\\_INTFL1.tx\\_ov](#) bit, and one of two things happen, depending on the value of [I2Cn\\_CTRL.clkstr\\_dis](#):

- If clock stretching is enabled ([I2Cn\\_CTRL.clkstr\\_dis = 0](#)), then the controller stretches the clock until the software makes space available in the receive FIFO by reading from [I2Cn\\_FIFO](#). Once space is available, the peripheral moves the data byte from the shift register into the receive FIFO, the SCL device pin is released, and the master is free to continue the transaction.

- If clock stretching is disabled (*I2Cn\_CTRL.clkstr\_dis* = 1), then the controller responds to the master with a NACK, and the data byte is lost. The master can return the bus to idle with a STOP condition or start a new transaction with a RESTART condition.

During a transmit transaction (which during master operation is a WRITE, and during slave operation is a READ), either 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 and transmit FIFO are flushed when the I<sup>2</sup>Cn port is disabled by clearing *I2Cn\_CTRL.en*=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 software under the following conditions:

- General call address match - Automatic flushing and lockout can be disabled by setting *I2Cn\_TXCTRL0.gc\_addr\_flush\_dis*.
- Slave address match write - Automatic flushing and lockout can be disabled by setting *I2Cn\_TXCTRL0.wr\_addr\_flush\_dis*.
- Slave address match read - Automatic flushing and lockout can be disabled by setting *I2Cn\_TXCTRL0.rd\_addr\_flush\_dis*.
- During operation as a slave transmitter, a NACK is received. Automatic flushing and lockout can be disabled by setting *I2Cn\_TXCTRL0.nack\_flush\_dis*.
- Any of the following interrupts (Automatic flushing cannot be disabled for these conditions):
  - ◆ Arbitration error
  - ◆ Timeout error
  - ◆ Master mode address NACK error
  - ◆ Master mode data NACK error
  - ◆ Start error
  - ◆ Stop error

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\_INTF0.tx\_lockout* = 1) and writes to the transmit FIFO are ignored until the software acknowledges the external event by clearing *I2Cn\_INTF0.tx\_lockout*.

#### 13.4.10 Transmit FIFO Preloading

There may be situations during slave mode operation where software wants to preload the transmit FIFO prior to transmission, such as when clock stretching is disabled. In this scenario, rather than responding to an external master requesting data with an ACK and clock stretching while software writes the data to the transmit FIFO, the controller instead 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 master 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 master. Setting *I2Cn\_TXCTRL1.preload\_rdy* to 1 sends an ACK to the master on the next read

transaction and transmits the data in the transmit FIFO. Preloading the transmit FIFO should be complete prior to setting the `I2Cn_TXCTRL1.preload_rdy` field to 1.

The required steps for implementing transmit FIFO Preloading in an application are as follow:

1. Enable transmit FIFO preloading by setting `I2Cn_TXCTRL0.preload_mode` to 1.  
This automatically clears the `I2Cn_TXCTRL1.preload_rdy` to 0.
2. If the transmit FIFO lockout flag (`I2Cn_INTFLO.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 master 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 master.
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 may repeat the preloading process or disable transmit FIFO preloading.

*Note: To prevent the preloaded data from being cleared when the master 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 whether the other auto flush disable bits should be set. For example, if a master uses I<sup>2</sup>C WRITE transactions to determine what data the slave should send in the following READ transactions, then the software can clear `I2Cn_TXCTRL0.wr_addr_flush_dis` to 0. Then when a WRITE occurs, the transmit FIFO is flushed, giving the software time to load the new data. For the READ transaction, the external master can poll the slave address until the new data has been loaded and `I2Cn_TXCTRL1.preload_rdy` is set, at which point the peripheral responds with an ACK.*

### 13.4.11 Interactive Receive Mode (IRXM)

In some situations, the I2Cn might want to inspect and respond to each byte of received data. In this case, IRXM can be used. IRXM is enabled by setting `I2Cn_CTRL.irxm_en` = 1. If IRXM is enabled, it must occur before any I<sup>2</sup>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_INTFLO.irxm` = 1). The 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_INTFLO.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_INTFLO.irxm` flag, the software can disable IRXM and, if operating as a master, 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 does not apply to address bytes, only to data bytes.*

*Note: IRXM does not apply to general call address responses or START byte responses.*

*Note: When enabling IRXM and operating as a slave, clock stretching must remain enabled (`I2Cn_CTRL.clkstr_dis` = 0).*

### 13.4.12 Clock Stretching

When the I<sup>2</sup>Cn peripheral requires some response or intervention from the software in order 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<sup>2</sup>C Bus Specification defines the term 'clock stretching' to only apply to a slave device holding the SCL line low, this section describes situations where the I<sup>2</sup>Cn peripheral holds the SCL line low in either slave *or* master mode and refers to *both* as clock stretching.

When the I<sup>2</sup>Cn 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 IRX (I<sup>2</sup>Cn\_CTRL.irxm\_en = 1), the peripheral can also clock stretch *before* the ACK bit, allowing the software to decide whether to send an ACK or NACK.

For a transmit operation (as either master or slave), 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 I<sup>2</sup>Cn\_FIFO.data to stop clock stretching and continue the transaction. If operating in master mode, however, instead of sending more data, the software may also set either I<sup>2</sup>Cn\_MSTCTRL.stop or I<sup>2</sup>Cn\_MSTCTRL.restart to send a STOP or RESTART condition, respectively.

For a receive operation (as either master or slave), 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 I<sup>2</sup>Cn\_FIFO.data to stop clock stretching and continue the transaction. If operating in master mode and this is the final byte of the transaction, as determined by I<sup>2</sup>Cn\_RXCTRL1.cnt, then the software must also set either I<sup>2</sup>Cn\_MSTCTRL.stop or I<sup>2</sup>Cn\_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 masters do not support other devices stretching the clock, it is possible to completely disable all clock stretching during slave mode by setting I<sup>2</sup>Cn\_CTRL.clkstr\_dis to 1 and clearing I<sup>2</sup>Cn\_CTRL.irxm\_en to 0. In this case, instead of clock stretching the I<sup>2</sup>Cn peripheral sends a NACK if receiving data or sends 0xFF if transmitting data.*

*Note: The clock synchronization required to support other I<sup>2</sup>C master or slave devices stretching the clock is built into the peripheral and requires no intervention from software to operate correctly.*

### 13.4.13 Bus Timeout

The timeout register, I<sup>2</sup>Cn\_TIMEOUT.scl\_to\_val, is used to detect bus errors. [Equation 13-8](#) and [Equation 13-9](#) show equations for calculating the maximum and minimum timeout values based on the value loaded into the I<sup>2</sup>Cn\_TIMEOUT.scl\_to\_val field.

*Equation 13-8: I<sup>2</sup>C Timeout Maximum*

$$t_{TIMEOUT} \leq \left( \frac{1}{f_{I2C\_CLK}} \right) \times ((I2Cn\_TIMEOUT.scl\_to\_val \times 32) + 3)$$

Due to clock synchronization, the timeout is guaranteed to meet the following minimum time calculation shown in [Equation 13-9](#).

*Equation 13-9: I<sup>2</sup>C Timeout Minimum*

$$t_{TIMEOUT} \leq \left( \frac{1}{f_{I2C\_CLK}} \right) \times ((I2Cn\_TIMEOUT.scl\_to\_val \times 32) + 2)$$

The timeout feature is disabled when I<sup>2</sup>Cn\_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 I<sup>2</sup>Cn peripheral hardware drives SCL low and is reset by the I<sup>2</sup>Cn peripheral hardware when the SCL line is released.

The timeout counter only monitors if the I<sup>2</sup>Cn peripheral hardware is driving the SCL line low. It does not monitor if an external I<sup>2</sup>Cn 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 I<sup>2</sup>Cn 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 may hold the SCL line low longer than the maximum timeout supported, the timeout can be disabled by setting the timeout field to 0 (*I2Cn\_TIMEOUT.scl\_to\_val* = 0).

#### **13.4.14 DMA Control**

There are independent DMA channels for each transmit FIFO and 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 13-10: DMA Burst Size Calculation for I<sup>2</sup>C Transmit*

$$\text{DMA Burst Size} \leq \text{TX FIFO Depth} - \text{I2Cn\_TXCTRL0.tx\_thresh} = 8 - \text{I2Cn\_TXCTRL0.tx\_thresh}$$

where  $0 \leq \text{I2Cn\_TXCTRL0.tx\_thresh} \leq 7$

Applications trying to avoid transmit underflow 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 13-11: DMA Burst Size Calculation for I<sup>2</sup>C Receive*

$$\text{DMA Burst Size} \leq \text{I2Cn\_RXCTRL0.rx\_thresh}$$

where  $1 \leq \text{I2Cn\_RXCTRL0.rx\_thresh} \leq 8$

Applications trying to avoid receive overflow 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).*

To enable DMA transfers, enable the transmit DMA channel (*I2Cn\_DMA.tx\_en*) and the receive DMA channel (*I2Cn\_DMA.rx\_en*) if receiving data.

### **13.5 Registers**

See *Table 2-4* for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in *Table 13-5*. Register names for a specific instance are

defined by replacing "n" with the instance number. As an example, a register PERIPHERALn\_CTRL resolves to PERIPHERAL0\_CTRL and PERIPHERAL1\_CTRL for instances 0 and 1, respectively.

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

*Table 13-5: I<sup>2</sup>C Register Summary*

Offset	Register	Description
[0x0000]	I2Cn_CTRL	I <sup>2</sup> C Control Register
[0x0004]	I2Cn_STATUS	I <sup>2</sup> C Status Register
[0x0008]	I2Cn_INTFLO	I <sup>2</sup> C Interrupt Flags 0 Register
[0x000C]	I2Cn_INTENO	I <sup>2</sup> C Interrupt Enable 0 Register
[0x0010]	I2Cn_INFL1	I <sup>2</sup> C Interrupt Flags 1 Register
[0x0014]	I2Cn_INTEN1	I <sup>2</sup> C Interrupt Enable 1 Register
[0x0018]	I2Cn_FIFOLEN	I <sup>2</sup> C FIFO Length Register
[0x001C]	I2Cn_RXCTRL0	I <sup>2</sup> C Receive Control 0 Register
[0x0020]	I2Cn_RXCTRL1	I <sup>2</sup> C Receive Control 1 Register
[0x0024]	I2Cn_TXCTRL0	I <sup>2</sup> C Transmit Control 0 Register
[0x0028]	I2Cn_TXCTRL1	I <sup>2</sup> C Transmit Control 1 Register
[0x002C]	I2Cn_FIFO	I <sup>2</sup> C Transmit and Receive FIFO Register
[0x0030]	I2Cn_MSTCTRL	I <sup>2</sup> C Master Control Register
[0x0034]	I2Cn_CLKLO	I <sup>2</sup> C Clock Low Time Register
[0x0038]	I2Cn_CLKHI	I <sup>2</sup> C Clock High Time Register
(0x003C)	I2Cn_HSCLK	I <sup>2</sup> C Hs-Mode Clock Control Register
[0x0040]	I2Cn_TIMEOUT	I <sup>2</sup> C Timeout Register
[0x0048]	I2Cn_DMA	I <sup>2</sup> C DMA Enable Register
[0x004C]	I2Cn_SLAVE	I <sup>2</sup> C Slave Address Register

### 13.5.1 Register Details

*Table 13-6: I<sup>2</sup>C Control Register*

I <sup>2</sup> C Control			I2Cn_CTRL		[0x0000]
Bits	Field	Access	Reset	Description	
31:16	-	RO	0	<b>Reserved</b>	
15	hs_en	R/W	0	<b>Hs-Mode Enable</b> Set this field to 1 for I <sup>2</sup> C Hs-Mode operation. 0: Disabled 1: Enabled	
14	-	RO	0	<b>Reserved</b>	
13	one_mst_mode	R/W	0	<b>Single Master Only</b> When set to 1, the device MUST ONLY be used in a single master application with slave devices that are NOT going to hold SCL low (i.e., the slave devices never clock stretch)	
12	clkstr_dis	R/W	0	<b>Slave Mode Clock Stretching</b> 0: Enabled 1: Disabled	

I <sup>2</sup> C Control			I2Cn_CTRL		[0x0000]
Bits	Field	Access	Reset	Description	
11	read	R	0	<b>Slave Read/Write Bit Status</b> Returns the logic level of the R/W bit on a received address match ( <i>I2Cn_INTFLO.addr_match = 1</i> ) or general call match ( <i>I2Cn_INTFLO.gc_addr_match = 1</i> ). This bit is valid for three system clock cycles after the address match status flag is set.	
10	bb_mode	R/W	0	<b>Software Output Control Enabled</b> Setting this field to 1 enables software bit-bang control of the I2Cn Bus. 0: The I <sup>2</sup> C controller manages the SDA and SCL pins in the hardware. 1: SDA and SCL are controlled by the software using the <i>I2Cn_CTRL.sda_out</i> and <i>I2Cn_CTRL.scl_out</i> fields.	
9	sda	R	-	<b>SDA Status</b> 0: SDA pin is logic low. 1: SDA pin is logic high.	
8	scl	R	-	<b>SCL Status</b> 0: SCL pin is logic low. 1: SCL pin is logic high.	
7	sda_out	R/W	0	<b>SDA Pin Output Control</b> Set the state of the SDA hardware pin (actively pull low or float). 0: Pull SDA Low 1: Release SDA <i>Note: Only valid when I2Cn_CTRL.bb_mode=1</i>	
6	scl_out	R/W	0	<b>SCL Pin Output Control</b> Set the state of the SCL hardware pin (actively pull low or float). 0: Pull SCL low 1: Release SCL <i>Note: Only valid when I2Cn_CTRL.bb_mode =1</i>	
5	-	RO	0	<b>Reserved</b>	
4	irxm_ack	R/W	0	<b>IRXM Acknowledge</b> If IRXM is enabled ( <i>I2Cn_CTRL.rx_mode = 1</i> ), this field determines if the hardware sends an ACK or a NACK to an IRM transaction. 0: Respond to IRXM with ACK 1: Respond to IRXM with NACK	
3	irxm_en	R/W	0	<b>IRXM Enable</b> When receiving data, this field allows for an interactive receive mode (IRM) interrupt event after each received byte of data. The I2Cn peripheral hardware can be enabled to send either an ACK or NACK for IRXM. See the <i>Interactive Receive Mode</i> section for detailed information. 0: Disable 1: Enable <i>Note: Only set this field when the I<sup>2</sup>C bus is inactive.</i>	
2	gc_addr_en	R/W	0	<b>General Call Address Enable</b> 0: Ignore General Call Address 1: Acknowledge General Call Address	

I <sup>2</sup> C Control			I2Cn_CTRL		[0x0000]
Bits	Field	Access	Reset	Description	
1	mst_mode	R/W	0	<b>Master Mode Enable</b> 0: Slave mode enabled. 1: Master mode enabled.	
0	en	R/W	0	<b>I<sup>2</sup>C Peripheral Enable</b> 0: Disabled 1: Enabled	

 Table 13-7: I<sup>2</sup>C Status Register

I <sup>2</sup> C Status			I2Cn_STATUS		[0x0004]
Bits	Field	Access	Reset	Description	
31:6	-	RO	0	<b>Reserved</b>	
5	mst_busy	RO	0	<b>Master Mode I<sup>2</sup>C Bus Transaction Active</b> The peripheral is operating in master mode, and a valid transaction beginning with a START command is in progress on the I <sup>2</sup> C bus. This bit reads 1 until the master ends the transaction with a STOP command. This bit continues to read 1 while a slave performs clock stretching. 0: Device not actively driving SCL clock cycles. 1: Device operating as master and actively driving SCL clock cycles.	
4	tx_full	RO	0	<b>Transmit FIFO Full</b> 0: Not full 1: Full	
3	tx_em	RO	1	<b>Transmit FIFO Empty</b> 0: Not empty 1: Empty	
2	rx_full	RO	0	<b>Receive FIFO Full</b> 0: Not full 1: Full	
1	rx_em	RO	1	<b>Receive FIFO Empty</b> 0: Not empty 1: Empty	
0	busy	RO	0	<b>Master or Slave Mode I<sup>2</sup>C Busy Transaction Active</b> The peripheral is operating in master or slave mode, and a valid transaction beginning with a START command is in progress on the I <sup>2</sup> C bus. This bit reads 1 until the peripheral acting as a master or an external master ends the transaction with a STOP command. This bit continues to read 1 while a slave performs clock stretching. 0: I <sup>2</sup> C bus is idle. 1: I <sup>2</sup> C bus transaction in progress.	

 Table 13-8: I<sup>2</sup>C Interrupt Flag 0 Register

I <sup>2</sup> C Interrupt Flag 0			I2Cn_INTF0		[0x0008]
Bits	Field	Access	Reset	Description	
31:24	-	RO	0	<b>Reserved</b>	

I <sup>2</sup> C Interrupt Flag 0				I2Cn_INTFL0	[0x0008]
Bits	Field	Access	Reset	Description	
23	wr_addr_match	R/W1C	0	<b>Slave Write Address Match Interrupt Flag</b> If set, the device has been accessed for a write (i.e., receive) transaction in slave mode, and the address received matches the device slave address. 0: No address match. 1: Address match.	
22	rd_addr_match	R/W1C	0	<b>Slave Read Address Match Interrupt Flag</b> If set, the device has been accessed for a read (i.e., transmit) transaction in slave mode, and the address received matches the device slave address. 0: No address match. 1: Address match.	
21:17	-	RO	0	<b>Reserved</b>	
16	-	R/W1C	0	<b>MAMI Interrupt Flag</b>	
15	tx_lockout	R/W1C	0	<b>Transmit FIFO Locked Interrupt Flag</b> 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	<b>Out of Sequence STOP Interrupt Flag</b> This flag is set if a STOP condition occurs out of 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	<b>Out of Sequence START Interrupt Flag</b> This flag is set if a START condition occurs out of 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	<b>Slave Mode Do Not Respond Interrupt Flag</b> 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 <sup>2</sup> C address match has occurred, and either the transmit or receive FIFO is not configured.	
11	data_err	R/W1C	0	<b>Master Mode Data NACK from External Slave Interrupt Flag</b> This flag is set by hardware if a NACK is received from a slave. This flag is only valid if the I2Cn peripheral is configured for master mode operation. Write 1 to clear. Write 0 has no effect. 0: Error condition has not occurred. 1: Data NACK received from a slave.	
10	addr_nack_err	R/W1C	0	<b>Master Mode Address NACK from Slave Error Flag</b> This flag is set by the hardware if an Address NACK is received from a slave bus. This flag is only valid if the I2Cn peripheral is configured for master mode operation. Write 1 to clear. Write 0 has no effect. 0: Error condition has not occurred. 1: Address NACK received from a slave.	

I <sup>2</sup> C Interrupt Flag 0				I2Cn_INTFL0	[0x0008]
Bits	Field	Access	Reset	Description	
9	to_err	R/W1C	0	<b>Timeout Error Interrupt Flag</b> This field is set to 1 when this device holds the SCL low longer than the programmed timeout value, in either master or slave 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	<b>Master Mode Arbitration Lost Interrupt Flag</b> Write 1 to clear. Write 0 has no effect. 0: Condition has not occurred. 1: Condition occurred.	
7	addr_ack	R/W1C	0	<b>Master Mode Address ACK from External Slave Interrupt Flag</b> This field is set when a slave address ACK is received. Write 1 to clear. Write 0 has no effect. 0: Condition has not occurred. 1: The slave device ACK for the address was received.	
6	stop	R/W1C	0	<b>Slave Mode STOP Condition Interrupt Flag</b> This flag is set by the 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	<b>Transmit FIFO Threshold Level Interrupt Flag</b> This field is set by the hardware if the number of bytes in the transmit FIFO is less than or equal to the transmit FIFO threshold level. Write 1 to clear. This field is automatically cleared by 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 fewer than or equal to the transmit threshold level.	
4	rx_thd	R/W1C	1	<b>Receive FIFO Threshold Level Interrupt Flag</b> This field is set by the hardware if the number of bytes in the receive FIFO is greater than or equal to the receive FIFO threshold level. This field is automatically cleared when the receive FIFO contains fewer bytes than the receive threshold setting. 0: Receive FIFO contains fewer bytes than the receive threshold level. 1: Receive FIFO contains at least receive threshold level of bytes.	
3	addr_match	R/W1C	0	<b>Slave Mode Incoming Address Match Status Interrupt Flag</b> Write 1 to clear. Writing 0 has no effect. 0: Slave address match has not occurred. 1: Slave address match occurred.	
2	gc_addr_match	R/W1C	0	<b>Slave Mode General Call Address Match Received Interrupt Flag</b> Write 1 to clear. Writing 0 has no effect. 0: General call address match has not occurred. 1: General call address match occurred.	
1	irxm	R/W1C	0	<b>IRXM Interrupt Flag</b> Write 1 to clear. Writing 0 is ignored. 0: Interrupt condition has not occurred. 1: Interrupt condition occurred.	

I <sup>2</sup> C Interrupt Flag 0				I2Cn_INTFL0	[0x0008]
Bits	Field	Access	Reset	Description	
0	done	R/W1C	0	<b>Transfer Complete Interrupt Flag</b> This flag is set for both master and slave mode once a transaction completes. Write 1 to clear. Writing 0 has no effect. 0: Transfer is not complete. 1: Transfer complete.	

Table 13-9: I<sup>2</sup>C Interrupt Enable 0 Register

I <sup>2</sup> C Interrupt Enable 0				I2Cn_INTCEN0	[0x000C]
Bits	Field	Access	Reset	Description	
31:24	-	RO	0	<b>Reserved</b>	
23	wr_addr_match	R/W	0	<b>Slave Write Address Match Interrupt Enable</b> This bit is set to enable interrupts when the device is accessed in slave mode, and the address received matches the device slave addressed for a write transaction. 0: Disabled 1: Enabled	
22	rd_addr_match	R/W	0	<b>Slave Read Address Match Interrupt Enable</b> This bit is set to enable interrupts when the device is accessed in slave mode, and the address received matches the device slave addressed for a read transaction. 0: Disabled 1: Enabled	
21:17	-	RO	0	<b>Reserved</b>	
16	mami	R/W	0	<b>MAMI Interrupt Enable</b>	
15	tx_lockout	R/W	0	<b>transmit FIFO Lock Out Interrupt Enable</b> 0: Disabled 1: Enabled	
14	stop_err	R/W	0	<b>Out of Sequence STOP Condition Detected Interrupt Enable</b> 0: Disabled 1: Enabled	
13	start_err	R/W	0	<b>Out of Sequence START Condition Detected Interrupt Enable</b> 0: Disabled 1: Enabled	
12	dnr_err	R/W	0	<b>Slave Mode Do Not Respond Interrupt Enable</b> Set this field to enable interrupts in slave mode when the "Do Not Respond" condition occurs. 0: Disabled 1: Enabled	
11	data_err	R/W	0	<b>Master Mode Received Data NACK from Slave Interrupt Enable</b> 0: Disabled 1: Enabled	
10	addr_nack_err	R/W	0	<b>Master Mode Received Address NACK from Slave Interrupt Enable</b> 0: Disabled 1: Enabled	
9	to_err	R/W	0	<b>Timeout Error Interrupt Enable</b> 0: Disabled 1: Enabled	

I <sup>2</sup> C Interrupt Enable 0				I2Cn_INTENO	[0x000C]
Bits	Field	Access	Reset	Description	
8	arb_err	R/W	0	<b>Master Mode Arbitration Lost Interrupt Enable</b> 0: Disabled 1: Enabled	
7	addr_ack	R/W	0	<b>Received Address ACK from Slave Interrupt Enable</b> Set this field to enable interrupts for master mode slave device address ACK events. 0: Disabled 1: Enabled	
6	stop	R/W	0	<b>STOP Condition Detected Interrupt Enable</b> 0: Disabled 1: Enabled	
5	tx_thd	R/W	0	<b>Transmit FIFO Threshold Level Interrupt Enable</b> 0: Disabled 1: Enabled	
4	rx_thd	R/W	0	<b>Receive FIFO Threshold Level Interrupt Enable</b> 0: Disabled 1: Enabled	
3	addr_match	R/W	0	<b>Slave Mode Incoming Address Match Interrupt Enable</b> 0: Disabled 1: Enabled	
2	gc_addr_match	R/W	0	<b>Slave Mode General Call Address Match Received Interrupt Enable</b> 0: Disabled 1: Enabled	
1	irxm	R/W	0	<b>Interactive Receive Interrupt Enable</b> 0: Disabled 1: Enabled	
0	done	R/W	0	<b>Transfer Complete Interrupt Enable</b> 0: Disabled 1: Enabled	

Table 13-10: I<sup>2</sup>C Interrupt Flag 1 Register

I <sup>2</sup> C Interrupt Status Flags 1				I2Cn_INTFL1	[0x0010]
Bits	Field	Access	Reset	Description	
31:3	-	RO	0	<b>Reserved</b>	
2	start	R/W1C	0	<b>START Condition Status Flag</b> If set, a device START condition has been detected. 0: START condition not detected. 1: START condition detected.	
1	tx_un	R/W1C	0	<b>Slave Mode Transmit FIFO Underflow Status Flag</b> In slave mode operation, the hardware sets this flag automatically if the transmit FIFO is empty and the master requests more data by sending an ACK after the previous byte is transferred. 0: Slave mode transmit FIFO underflow condition has not occurred. 1: Slave mode transmit FIFO underflow condition occurred.	

I <sup>2</sup> C Interrupt Status Flags 1				I2Cn_INTFL1	[0x0010]
Bits	Field	Access	Reset	Description	
0	rx_ov	R/W1C	0	<b>Slave Mode Receive FIFO Overflow Status Flag</b> In slave mode operation, the hardware sets this flag automatically when a receive FIFO overflow occurs. Write 1 to clear. Writing 0 has no effect. 0: Slave mode receive FIFO overflow event has not occurred. 1: Slave mode receive FIFO overflow condition occurred (data lost).	

Table 13-11: I<sup>2</sup>C Interrupt Enable 1 Register

I <sup>2</sup> C Interrupt Enable 1				I2Cn_INTEN1	[0x0014]
Bits	Field	Access	Reset	Description	
31:3	-	RO	0	<b>Reserved</b>	
2	start	R/W	0	<b>START Condition Interrupt Enable</b> 0: Disabled. 1: Enabled.	
1	tx_un	R/W	0	<b>Slave Mode Transmit FIFO Underflow Interrupt Enable</b> 0: Disabled. 1: Enabled.	
0	rx_ov	R/W	0	<b>Slave Mode Receive FIFO Overflow Interrupt Enable</b> 0: Disabled. 1: Enabled.	

Table 13-12: I<sup>2</sup>C FIFO Length Register

I <sup>2</sup> C FIFO Length				I2Cn_FIFOLEN	[0x0018]
Bits	Field	Access	Reset	Description	
31:16	-	RO	0	<b>Reserved</b>	
15:8	tx_depth	RO	8	<b>Transmit FIFO Length</b> Reading this field returns the depth of the transmit FIFO. 8: 8-bytes	
7:0	rx_depth	RO	8	<b>Receive FIFO Length</b> Reading this field returns the depth of the receive FIFO. 8: 8-bytes	

Table 13-13: I<sup>2</sup>C Receive Control 0 Register

I <sup>2</sup> C Receive Control 0				I2Cn_RXCTRL0	[0x001C]
Bits	Field	Access	Reset	Description	
31:12	-	RO	0	<b>Reserved</b>	

I <sup>2</sup> C Receive Control 0			I2Cn_RXCTRL0		[0x001C]
Bits	Field	Access	Reset	Description	
11:8	thd_lvl	R/W	0	<b>Receive FIFO Threshold Level</b> 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 <i>I2Cn_INTFL0.rx_thd</i> 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	<b>Flush Receive FIFO</b> 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	<b>Reserved</b>	
0	dnr	R/W	0	<b>Slave Mode Do Not Respond</b> Slave 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 then always respond to data bytes with a NACK. 1: NACK the address.	

 Table 13-14: I<sup>2</sup>C Receive Control 1 Register

I <sup>2</sup> C Receive Control 1			I2Cn_RXCTRL1		[0x0020]
Bits	Field	Access	Reset	Description	
31:12	-	RO	0	<b>Reserved</b>	
11:8	lvl	R	0	<b>Receive FIFO Byte Count Status</b> 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	

I <sup>2</sup> C Receive Control 1			I2Cn_RXCTRL1		[0x0020]
Bits	Field	Access	Reset	Description	
7:0	cnt	R/W	1	<b>Receive FIFO Transaction Byte Count Configuration</b> When in master mode, write the number of bytes to be received in a transaction from 1 to 256. 0x00 represents 256. 0: 256 byte receive transaction. 1: 1 byte receive transaction. 2: 2 byte receive transaction. ... 255: 255 byte receive transaction. <i>This field is ignored when I2Cn_CTRL.irxm_en = 1. To receive more than 256 bytes, use I2Cn_CTRL.irxm_en = 1</i>	

Table 13-15: I<sup>2</sup>C Transmit Control 0 Register

I <sup>2</sup> C Transmit Control 0			I2Cn_TXCTRL0		[0x0024]
Bits	Field	Access	Reset	Description	
31:12	-	RO	0	<b>Reserved</b>	
11:8	thd_val	R/W	0	<b>Transmit FIFO Threshold Level</b> 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 <i>I2Cn_INTFLO.thd_val</i> 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	
7	flush	R/W1O	0	<b>Transmit FIFO Flush</b> 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 <i>Note: The hardware automatically clears this bit to 0 after it is written to 1 when the flush is completed.</i> <i>If I2Cn_INTFLO.tx_lockout = 1, then I2Cn_TXCTRL0.flush = 1.</i>	
6	-	RO	0	<b>Reserved</b>	
5	nack_flush_dis	R/W	0	<b>Transmit FIFO received NACK Auto Flush Disable</b> Various situations or conditions are described in this user guide that leads to the transmit FIFO being flushed and locked out ( <i>I2Cn_INTFLO.tx_lockout</i> = 1). 0: Received NACK at the end of a slave transmit operation enabled 1: Received NACK at the end of a slave transmit operation disabled. <i>Note: upon entering transmit preload mode, the hardware automatically sets this bit to 0</i> <i>The software can subsequently set to any value desired (i.e., The hardware does not continuously force the bitfield to this value).</i>	

I <sup>2</sup> C Transmit Control 0				I2Cn_TXCTRL0	[0x0024]
Bits	Field	Access	Reset	Description	
4	rd_addr_flush_dis	R/W	0	<b>Transmit FIFO Slave Address Match Read Auto Flush Disable</b> Various situations or conditions are described in this user guide that leads to the transmit FIFO being flushed and locked out ( <i>I2Cn_INTFLO.tx_lockout</i> = 1). 0: Enabled. 1: Disabled. <i>Note: upon entering transmit preload mode, the hardware automatically sets this bit to 1</i> <i>The software can subsequently set to any value desired (i.e., The hardware does not continuously force the bitfield to this value).</i>	
3	wr_addr_flush_dis	R/W	0	<b>Transmit FIFO Slave Address Match Write Auto Flush Disable</b> Various situations or conditions are described in this user guide that leads to the transmit FIFO being flushed and locked out ( <i>I2Cn_INTFLO.tx_lockout</i> = 1). 0: Enabled. 1: Disabled. <i>Note: upon entering transmit preload mode, the hardware automatically sets this bit to 1</i> <i>The software can subsequently set to any value desired (i.e., The hardware does not continuously force the bitfield to this value).</i>	
2	gc_addr_flush_dis	R/W	0	<b>Transmit FIFO General Call Address Match Auto Flush Disable</b> Various situations or conditions are described in this user guide that leads to the transmit FIFO being flushed and locked out ( <i>I2Cn_INTFLO.tx_lockout</i> = 1). 0: Enabled. 1: Disabled. <i>Note: upon entering transmit preload mode, the hardware automatically sets this bit to 1</i> <i>The software can subsequently set to any value desired (i.e., The hardware does not continuously force the bitfield to this value).</i>	
1	tx_ready_mode	R/W	0	<b>Transmit FIFO Ready Manual Mode</b> 0: The hardware controls the <i>I2Cn_TXCTRL1.preload_rdy</i> field. 1: The software control of the <i>I2Cn_TXCTRL1.preload_rdy</i> field.	
0	preload_mode	R/W	0	<b>Transmit FIFO Preload Mode Enable</b> 0: Normal operation. An address match in slave mode, or a general call address match, flushes and locks the transmit FIFO so it cannot be written and sets the <i>I2Cn_INTFLO.tx_lockout</i> field to 1. 1: transmit FIFO preload mode. An address match in slave mode, or a general call address match, does not lock the transmit FIFO and does not set <i>I2Cn_INTFLO.tx_lockout</i> . This allows the software to preload data into the transmit FIFO. The status of the I <sup>2</sup> C is controllable at <i>I2Cn_TXCTRL1.preload_rdy</i> .	

 Table 13-16: I<sup>2</sup>C Transmit Control 1 Register

I <sup>2</sup> C Transmit Control Register 1				I2Cn_TXCTRL1	[0x0028]
Bits	Field	Access	Reset	Description	
31:12	-	RO	0	Reserved	

I <sup>2</sup> C Transmit Control Register 1			I2Cn_TXCTRL1		[0x0028]
Bits	Field	Access	Reset	Description	
11:8	lvl	R	0	<b>Transmit FIFO Byte Count Status</b> 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	<b>Reserved</b>	
0	preload_rdy	R/W1O	1	<b>Transmit FIFO Preload Ready Status</b> When transmit FIFO preload mode is enabled, <i>I2Cn_TXCTRL0.preload_mode</i> = 1, this bit is automatically cleared to 0. While this bit is 0, if the hardware receives a slave address match, a NACK is sent. Once the hardware is ready (software preloaded the transmit FIFO, configured the DMA, etc.), the software must set this bit to 1, so the hardware sends an ACK on a slave address match.  When transmit FIFO preload mode is disabled, <i>I2Cn_TXCTRL0.preload_mode</i> = 1, this bit is forced to 1, and the hardware behaves normally.	

Table 13-17: I<sup>2</sup>C Data Register

I <sup>2</sup> C Data			I2Cn_FIFO		[0x002C]
Bits	Field	Access	Reset	Description	
31:8	-	RO	0	<b>Reserved</b>	
7:0	data	R/W	0xFF	<b>FIFO Data</b> 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 13-18: I<sup>2</sup>C Master Control Register

I <sup>2</sup> C Master Control			I2Cn_MSTCTRL		[0x0030]
Bits	Field	Access	Reset	Description	
31:11	-	RO	0	<b>Reserved</b>	
10:8	mcode	R/W	0	<b>MCODE</b> This field sets the master code used in Hs-mode operation	
7	ex_addr_en	R/W	0	<b>Slave Extended Addressing Enable</b> 0: Send a 7-bit address to the slave. 1: Send a 10-bit address to the slave.	
6:3	-	RO	0	<b>Reserved</b>	
2	stop	R/W1O	0	<b>Send STOP Condition</b> 1: Send a STOP Condition at the end of the current transaction <i>Note: This bit is automatically cleared by the hardware when the STOP condition begins.</i>	

I <sup>2</sup> C Master Control			I2Cn_MSTCTRL		[0x0030]
Bits	Field	Access	Reset	Description	
1	restart	R/W1O	0	<b>Send Repeated START Condition</b> After sending data to a slave, the master may send another START to retain control of the bus. 1: Send a repeated START condition to the slave instead of sending a STOP condition at the end of the current transaction. <i>Note: This bit is automatically cleared by the hardware when the repeated START condition begins.</i>	
0	start	R/W1O	0	<b>Start Master Mode Transfer</b> 1: Start master mode transfer <i>Note: This bit is automatically cleared by the hardware when the transfer is completed or aborted.</i>	

 Table 13-19: I<sup>2</sup>C SCL Low Control Register

I <sup>2</sup> C Clock Low Control			I2Cn_CLKLO		[0x0034]
Bits	Field	Access	Reset	Description	
31:9	-	RO	0	<b>Reserved</b>	
8:0	lo	R/W	0x001	<b>Clock Low Time</b> In master mode, this configures the SCL low time. $t_{SCL\_LO} = f_{I2C\_CLK} \times (lo + 1)$ <i>Note: 0 is not a valid setting for this field.</i>	

 Table 13-20: I<sup>2</sup>C SCL High Control Register

I <sup>2</sup> C Clock High Control			I2Cn_CLKHI		[0x0038]
Bits	Field	Access	Reset	Description	
31:9	-	RO	0	<b>Reserved</b>	
8:0	hi	R/W	0x001	<b>Clock High Time</b> In master mode, this configures the SCL high time. $t_{SCL\_HI} = 1/f_{I2C\_CLK} \times (hi + 1)$ In both master and slave mode, this also configures the time SCL is held low after new data is loaded from the transmit FIFO or after the software clears <i>I2Cn_INTFLO.irxm</i> during IRXM. <i>Note: 0 is not a valid setting for this field.</i>	

 Table 13-21: I<sup>2</sup>C Hs-Mode Clock Control Register

I <sup>2</sup> C Hs-Mode Clock Control			I2Cn_HSCLK		[0x003C]
Bits	Field	Access	Reset	Description	
31:16	-	R/W	0	<b>Reserved</b>	
15:8	hsclk_hi	R/W	0	<b>Hs-Mode Clock High Time</b> This field sets the Hs-Mode clock high count. In slave mode, this is the time SCL is held high after data is output on SDA. <i>Note: See section 13.4.4 SCL Clock Generation for Hs-mode for details on the requirements for the Hs-mode clock high and low times.</i>	

I <sup>2</sup> C Hs-Mode Clock Control			I2Cn_HSCLK		[0x003C]
Bits	Field	Access	Reset	Description	
7:0	hsclk_lo	R/W	0	<b>Hs-Mode Clock Low Time</b> This field sets the Hs-Mode clock low count. In slave mode, this is the time SCL is held low after data is output on SDA. <i>Note: See section <a href="#">13.4.4 SCL Clock Generation for Hs-mode</a> for details on the requirements for the Hs-mode clock high and low times.</i>	

 Table 13-22: I<sup>2</sup>C Timeout Register

I <sup>2</sup> C Timeout			I2Cn_TIMEOUT		[0x0040]
Bits	Field	Access	Reset	Description	
31:16	-	RO	0	<b>Reserved</b>	
15:0	scl_to_val	R/W	0	<b>Bus Error SCL Timeout Period</b> Set this value to the number of I <sup>2</sup> C clock cycles desired to cause a bus timeout error. The peripheral timeout timer starts when it pulls SCL low. After the peripheral releases the line, if the line is not pulled high prior to the timeout number of I <sup>2</sup> C clock cycles, a bus error condition is set ( <i>I2Cn_INTFL0.to_err</i> = 1), and the peripheral releases the SCL and SDA lines 0: Timeout disabled. All other values result in a timeout calculation of: $t_{BUS\_TIMEOUT} = \frac{1}{f_{I2C\_CLK}} \times scl\_to\_val$ <i>Note: The timeout counter monitors the I2Cn peripheral's driving of the SCL pin, not an external I<sup>2</sup>C device driving the SCL pin.</i>	

 Table 13-23: I<sup>2</sup>C DMA Register

I <sup>2</sup> C DMA			I2Cn_DMA		[0x0048]
Bits	Field	Access	Reset	Description	
31:2	-	RO	0	<b>Reserved</b>	
1	rx_en	R/W	0	<b>Receive DMA Channel Enable</b> 0: Disable 1: Enable	
0	tx_en	R/W	0	<b>Transmit DMA Channel Enable</b> 0: Disable 1: Enable	

 Table 13-24: I<sup>2</sup>C Slave Address Register

I <sup>2</sup> C Slave Address			I2Cn_SLAVE		[0x004C]
Bits	Field	Access	Reset	Description	
31:16	-	RO	0	<b>Reserved</b>	
15	ext_addr_en	R/W	0	<b>Slave Mode Extended Address Length Select</b> 0: 7-bit addressing 1: 10-bit addressing	
14:10	-	RO	0	<b>Reserved</b>	

I <sup>2</sup> C Slave Address			I2Cn_SLAVE		[0x004C]
Bits	Field	Access	Reset	Description	
9:0	addr	R/W	0	<p><b>Slave Mode Slave Address</b>            In slave mode operation, (<i>I2Cn_CTRL.mstr</i> = 0), set this field to the slave address for the I2Cn port. For 7-bit addressing, the address occupies the least significant 7 bits. For 10-bit addressing, the 9-bits of address occupies the most significant 9 bit, and the R/W bit occupies the least significant bit.</p> <p><i>Note:</i> <i>I2Cn_SLAVE.ext_addr_en</i> controls if this field is a 7-bit or 10-bit address.</p>	

## 14. Inter-Integrated Sound Interface (I<sup>2</sup>S)

I<sup>2</sup>S is a serial audio interface for communicating pulse-code modulation (PCM) encoded streams between devices. The peripheral supports both master and slave 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<sup>2</sup>S input clock.
- Flexible data format:
  - ◆ 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<sup>2</sup>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<sup>2</sup>S serial data input and serial data output pins.

### 14.1 Instances

*Table 14-1: MAX78000 I<sup>2</sup>S Instances*

Instance	Supported Channels	I <sup>2</sup> S_CLK Clock Options		Receive FIFO Depth	Transmit FIFO Depth
I2S0	Stereo	PCLK	I2S_EXTCLK	8 × 32-bits	8 × 32-bits

#### 14.1.1 I<sup>2</sup>S Bus Lines and Definitions

The I<sup>2</sup>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 In (SDI).
4. Serial Data Out (SDO).
5. External Clock Input (I2S\_EXTCLK) required for operation in master mode.

Detailed pin and alternate function mapping is shown in *Table 14-2*.

*Table 14-2. MAX78000 I<sup>2</sup>S Pin Mapping*

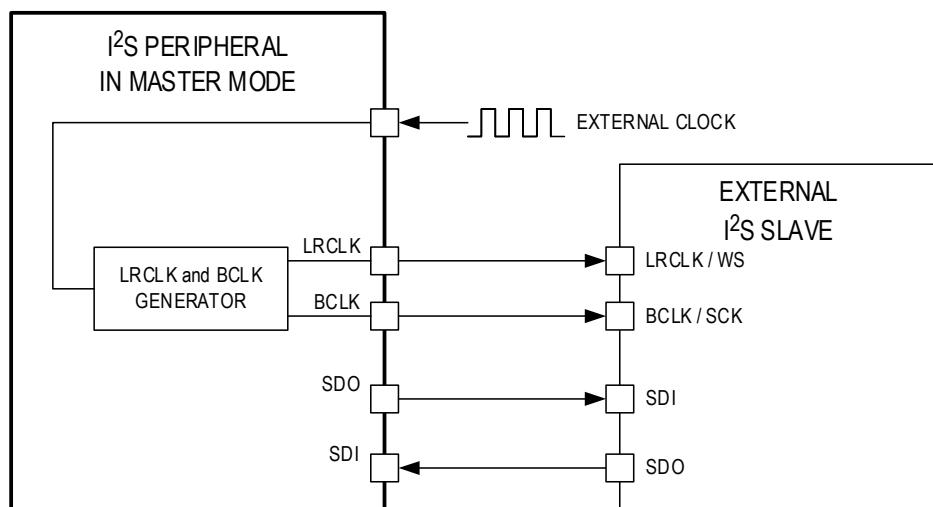
Instance	I <sup>2</sup> S Signal	Pin Description	81 CTBGA Pin Number	Alternate Function Number	Notes
I2S0	BCLK (SCK)	I <sup>2</sup> S bit clock	P1.2	AF1	Also referred to as serial clock
	LRCLK (WS)	I <sup>2</sup> S left/right clock (word select)	P1.3	AF1	Also referred to as word select
	SDI	I <sup>2</sup> S serial data input	P1.4	AF1	
	SDO	I <sup>2</sup> S serial data output	P1.5	AF1	
	I2S_CLKEXT	I <sup>2</sup> S external clock	P0.14	AF2	This input is required to use the I2S peripheral as a Master.

## 14.2 Details

The I<sup>2</sup>S supports full-duplex serial communication with separate SDI and SDO pins. *Figure 14-1* shows an interconnect between a peripheral configured in host mode, communicating with an external I<sup>2</sup>S slave receiver and an external I<sup>2</sup>S transmitter. In master mode, the peripheral hardware generates the BCLK and LRCLK, and both are output to each slave device.

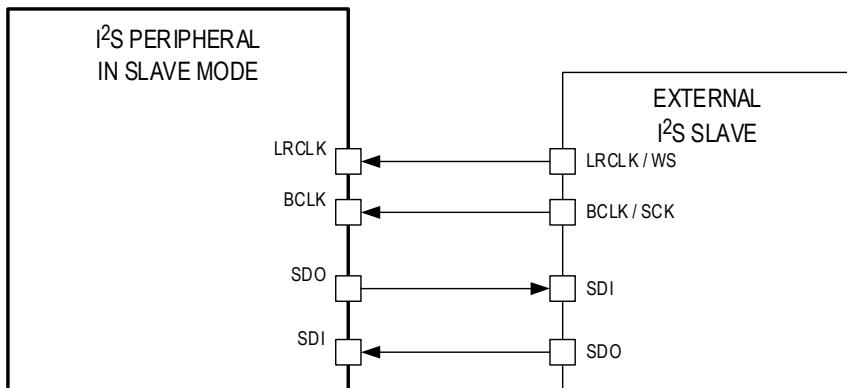
*Note: Master operation requires the use of the I2S\_EXTCLK signal to generate the LRCLK and BCLK signals.*

*Figure 14-1: I<sup>2</sup>S Master Mode, Full Duplex Connection*



*Figure 14-2* shows the I<sup>2</sup>S peripheral configured for slave operation. The LRCLK and BCLK signals are generated externally and are inputs to the I<sup>2</sup>S peripheral.

*Figure 14-2: I<sup>2</sup>S Slave Mode*



## 14.3 Master and Slave Mode Configuration

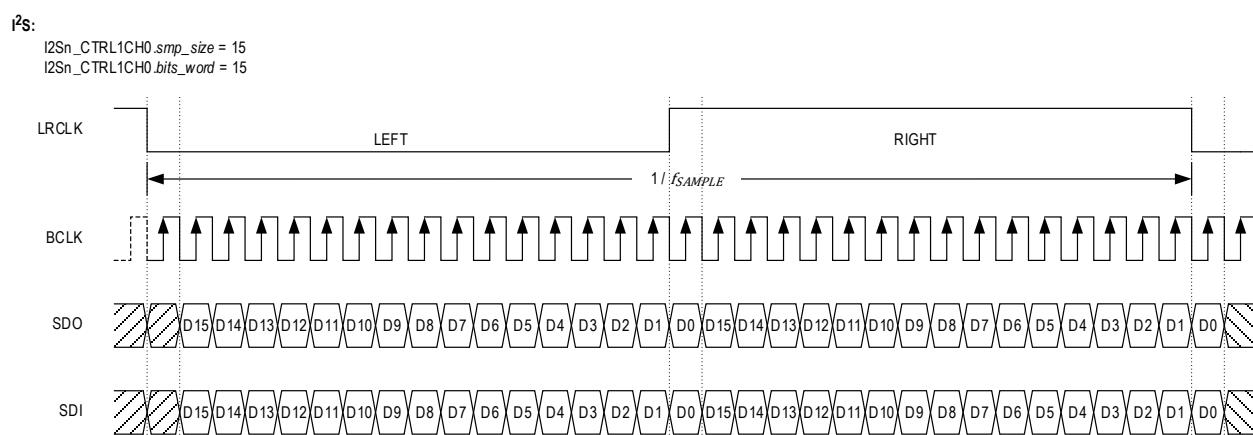
The hardware supports master and slave mode. In master mode, the BCLK and LRCK signals are generated internally, and output on the BCLK and LRCK pins. In slave mode, the BCLK and LRCK pins are configured as inputs, and the peripheral timing is controlled by the external clock source.

*Table 14-3: I<sup>2</sup>S Mode Configuration*

Device Mode	<i>I2Sn_CTRL1CH0.ch_mode</i>	LRCLK	BCLK
Master	0	Output to Slave	Output to Slave
Slave	3	Input from Master	Input from Master

## 14.4 Clocking

*Figure 14-3: Audio Interface I<sup>2</sup>S Signal Diagram*



I<sup>2</sup>S communication is synchronized using two signals, the LRCLK and the BCLK. When the I<sup>2</sup>S peripheral is configured as a master, the BCLK and LRCLK signals are generated internally by the peripheral using the I<sup>2</sup>S external clock signal. See [Table 14-2](#) for details of the I<sup>2</sup>S pin mapping and alternate function selection. If using the I<sup>2</sup>S peripheral in master mode, the I<sup>2</sup>S external clock must be enabled to generate the BCLK and LRCLK signals.

When the I<sup>2</sup>S peripheral is configured in slave mode, the BCLK and LRCLK pins must be configured as inputs. An external master generates the BCLK and LRCLK signals, which the peripheral uses to synchronize itself to the I<sup>2</sup>S bus. [Figure 14-3](#) shows the default signals and timing for I<sup>2</sup>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 14-1](#).

*Equation 14-1: CD Audio Bit Frequency Calculation*

$$f_{BCLK} = 44.1 \text{ kHz} \times 16 \times 2 = 1.4112 \text{ MHz}$$

#### 14.4.1 BCLK Generation for Master Mode

As indicated by [Equation 14-1](#), the requirements for determining the BCLK frequency are:

1. Audio sample frequency.
2. Number of bits per sample, also referred to as the sample width.

Using the above requirements, [Equation 14-2](#) shows the formula to calculate the bit clock frequency for a given audio file.

*Equation 14-2: Calculating the Bit Clock Frequency for Audio*

$$f_{BCLK} = f_{SAMPLE} \times \text{Sample Width} \times 2$$

In master mode, the I<sup>2</sup>S external clock input is used to generate the BCLK frequency. The I<sup>2</sup>S external clock is divided by the [\*I2Sn\\_CTRL1CH0.clkdiv\*](#) field to achieve the target BCLK frequency as shown in [Equation 14-3](#).

*Equation 14-3: Master Mode BCLK Generation Using the I<sup>2</sup>S External Clock*

$$f_{BCLK} = \frac{f_{ERFO}}{(I2Sn\_CTRL1CH0.\text{clkdiv} + 1) \times 2}$$

Use [Equation 14-4](#) to determine the I<sup>2</sup>S clock divider for a target BCLK frequency.

*Equation 14-4: Master Mode Clock Divisor Calculation for a Target Bit Clock Frequency*

$$I2Sn\_CTRL1CH0.\text{clkdiv} = \frac{f_{ERFO}}{2 \times f_{BCLK}} - 1$$

#### 14.4.2 LRCLK Period Calculation

An I<sup>2</sup>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 14-3](#). The LRCLK is a 50% duty cycle signal and is the same frequency as the audio sampling frequency,  $f_{SAMPLE}$ .

The I<sup>2</sup>S peripheral uses the Bits Per Word field, [\*I2Sn\\_CTRL1CH0.bits\\_word\*](#), to define the sample width of the audio, 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 [\*I2Sn\\_CTRL1CH0.bits\\_word\*](#) field to 15 for audio sampled using a 16-bit width.

*Equation 14-5: Bits Per Word Calculation*

$$I2Sn\_CTRL1CH0.\text{bits\_word} = \text{Sample Width} - 1$$

The LRCLK frequency, or word select frequency, is automatically generated by the I<sup>2</sup>S peripheral hardware when it is set to operate as a master. The LRCLK frequency calculation is shown in [Equation 14-6: LRCLK Frequency Calculation](#).

#### *Equation 14-6: LRCLK Frequency Calculation*

$$f_{LRCLK} = f_{BCLK} \times (I2Sn\_CTRL1CH0.bits\_word + 1)$$

## 14.5 Data Formatting

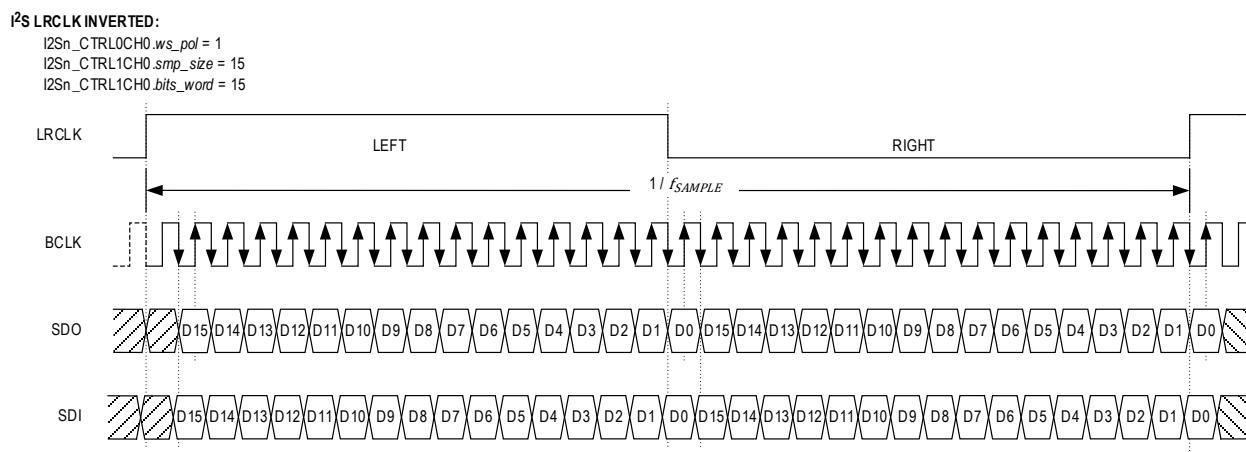
### 14.5.1 Sample Size

The sample size field, *I2Sn\_CTRL1CH0.smp\_word*, 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 *I2Sn\_CTRL1CH0.bits\_word* field. For example, for 16-bit sample width audio, the *I2Sn\_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 field to 0 is the equivalent of 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 *I2Sn\_CTRL1CH0.smp\_word* to 15 collects 16 samples. See [Figure 14-6](#) for an example of the bits per word field's setting compared to the sample size field's setting.

### 14.5.2 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, *I2Sn\_CTRL0CH0.ws\_pol*. By default, LRCLK low is for the left channel, high is for the right channel as shown in [Figure 14-3](#). Setting *I2Sn\_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 14-4](#).

*Figure 14-4: Audio Mode with Inverted Word Select Polarity*



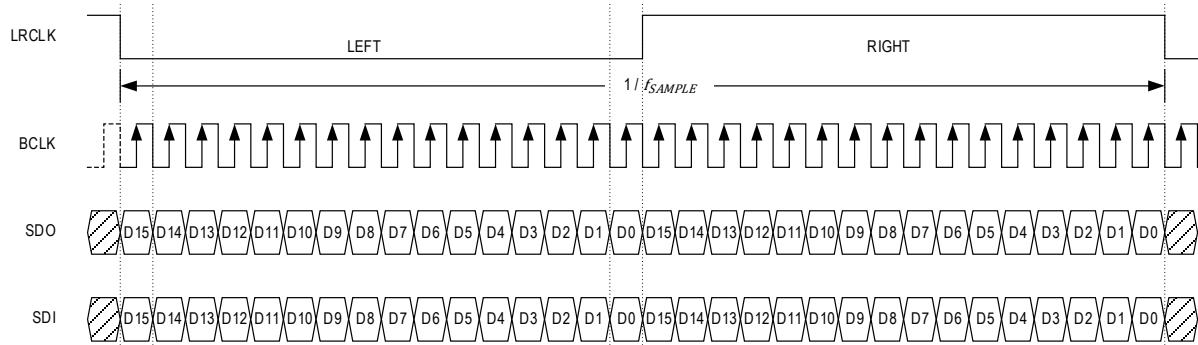
### 14.5.3 First Bit Location Control

The default setting is for the first bit of I<sup>2</sup>S data to be located at the second complete BCLK cycle after the LRCLK transition as required by the I<sup>2</sup>S specification. See [Figure 14-3](#) 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 14-5](#). Set *I2Sn\_CTRL0CH0.msb\_loc* to 1 to left justify the data with respect to the LRCLK.

*Figure 14-5: Audio Master Mode Left-Justified First Bit Location*

**Audio Mode:**

*I2Sn\_CTRL0CH0.msb\_bc = 1*  
*I2Sn\_CTRL0CH0.wsize = 1 (Half-Word)*  
*I2Sn\_CTRL1CH0.smp\_size = 15*  
*I2Sn\_CTRL1CH0.bits\_word = 15*



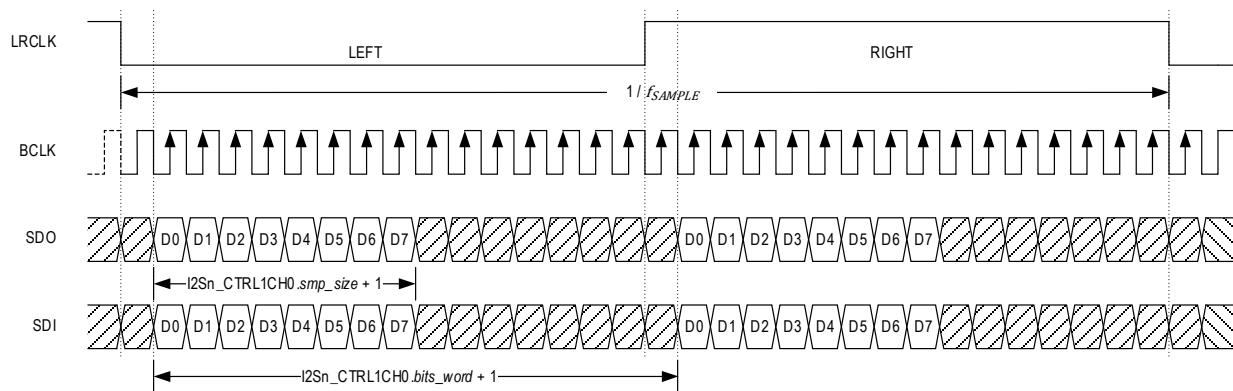
#### 14.5.4 Sample Adjustment

When the sample size field, *I2Sn\_CTRL1CH0.smp\_size*, is less than the bits per word field, *I2Sn\_CTRL1CH0.bits\_word*, use the *I2Sn\_CTRL1CH0.adrst* 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 14-6* shows an example of the default adjustment, MSB, where *I2Sn\_CTRL1CH0.smp\_size = 7* and *I2Sn\_CTRL1CH0.bits\_word = 15*. *Figure 14-7* shows the adjustment set to the LSB of the SDI/SDO data.

*Figure 14-6: MSB Adjustment when Sample Size is Less Than Bits Per Word*

**I<sup>2</sup>S with Left Adjustment:**

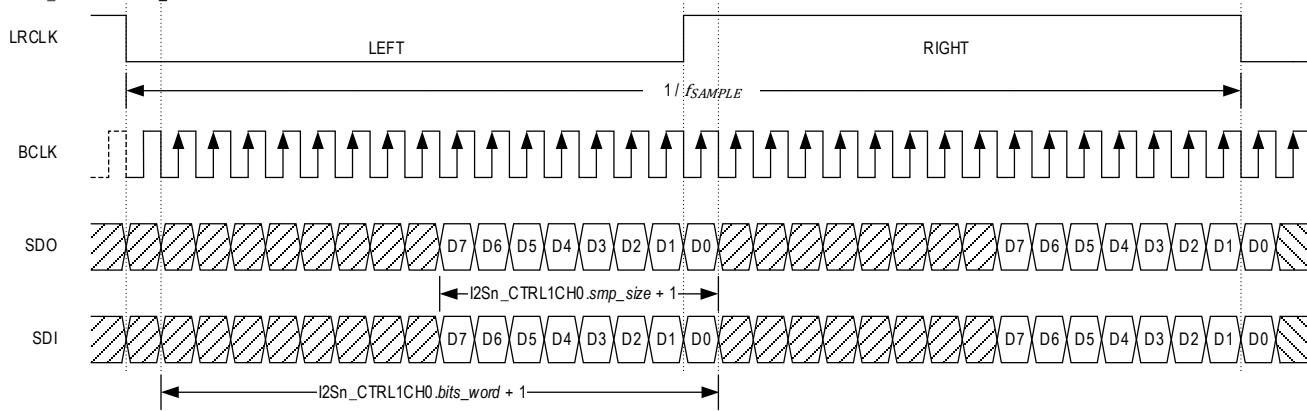
*I2Sn\_CTRL1CH0.lsb\_first = 1*  
*I2Sn\_CTRL1CH0.smp\_size = 7*  
*I2Sn\_CTRL1CH0.bits\_word = 15*



*Figure 14-7: LSB Adjustment when Sample Size is Less Than Bits Per Word*

**I<sup>2</sup>S with Right Adjustment**

`I2Sn_CTRL1CH0.adjust = 1`  
`I2Sn_CTRL0CH0.wsize = 1 (Half-Word)`  
`I2Sn_CTRL1CH0.smp_size = 7`  
`I2Sn_CTRL1CH0.bits_word = 15`



#### 14.5.5 Stereo/Mono Configuration

The I<sup>2</sup>S can transfer stereo or mono data based on the `I2Sn_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 `I2Sn_CTRL0CH0.stereo` to 0. Set `I2Sn_CTRL0CH0.stereo` field to 2 for left channel mono. Set `I2Sn_CTRL0CH0.stereo` field to 3 for right channel mono.

*Figure 14-8: I<sup>2</sup>S Mono Left Mode*

**I<sup>2</sup>S MONO LEFT:**

`I2Sn_CTRL0CH0.stereo = 2`  
`I2Sn_CTRL1CH0.smp_size = 15`  
`I2Sn_CTRL1CH0.bits_word = 15`

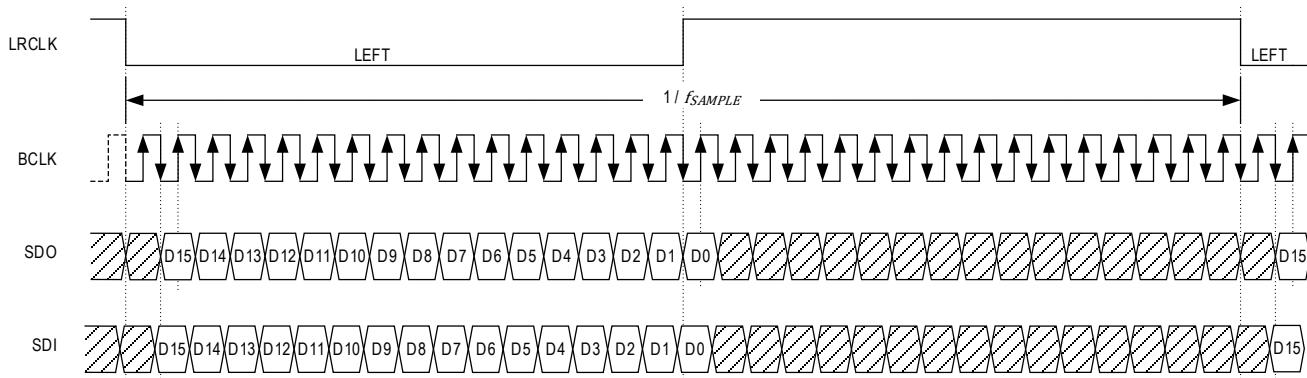
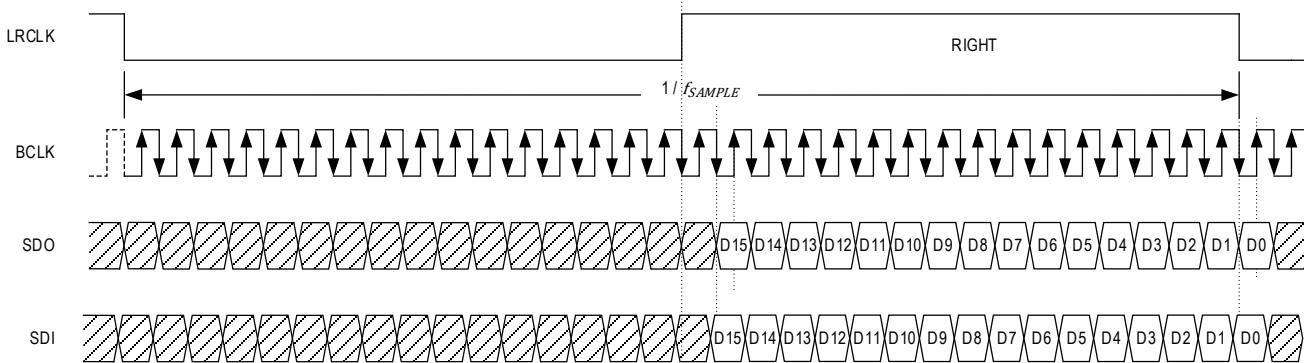


Figure 14-9: I<sup>2</sup>S Mono Right Mode

#### I<sup>2</sup>S MONO RIGHT:

```
I2Sn_CTRL0CHO.stereo = 3
I2Sn_CTRL1CHO.smp_size = 15
I2Sn_CTRL1CHO.bits_word = 15
```



## 14.6 Transmit and Receive FIFOs

### 14.6.1 FIFO Data Width

I<sup>2</sup>S audio data is programmable from 1 to 32 bits using the `I2Sn_CTRL1CHO.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 `I2Sn_CTRL0CHO.wsize` field. For FIFO word sizes less than 32-bits, the data frame, comprising of a full LRCLK cycle, may still be 64 bits; the unused bits are transmitted as zero by the hardware.

### 14.6.2 Transmit FIFO

An I<sup>2</sup>S transaction is started by writing data to the transmit FIFO using the `I2Sn_FIFOCHO.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 `I2Sn_CTRL0CHO.wsize` field, at a time, in the order it was written to the transmit FIFO. Use the I<sup>2</sup>S interrupt flags to monitor the transmit FIFO status and determine when the transfer cycle(s) have completed.

If the transmit FIFO becomes empty, an error condition occurs, and results in undefined behavior.

### 14.6.3 Receive FIFO

The received data is loaded into the receive FIFO, and it can then be unloaded by reading from the `I2Sn_FIFOCHO.data` register. An overrun event occurs if the receive FIFO is full and another word is shifted into the FIFO.

### 14.6.4 FIFO Word Control

The data width of the transmit and receive FIFOs can be configured using the `I2Sn_CTRL0CHO.wsize` field. *Table 14-4*, *Table 14-5*, and *Table 14-6*, describe the data ordering based on the `I2Sn_CTRL0CHO.wsize` setting.

The transmit and receive FIFOs must be flushed, and the peripheral reset by the software, prior to reconfiguration. The software resets the peripheral by setting the `I2Sn_CTRL0CHO.rst` field to 1.

Table 14-4: Data Ordering for Byte Data Size (Stereo Mode)

Byte Data Width ( <i>I2Sn_CTRLCHO.wsize</i> = 0)				
FIFO Entry	MSByte			LSByte
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 14-5: Data Ordering for Half-Word Data Size (Stereo Mode)

Half-Word Data Width ( <i>I2Sn_CTRLCHO.wsize</i> = 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 14-6: Data Ordering for Word Data Size (Stereo Mode)

Word Data Width ( <i>I2Sn_CTRLCHO.wsize</i> = 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

#### 14.6.5 FIFO Data Alignment

The I<sup>2</sup>S data can be left aligned (reset default), or right aligned, using the *I2Sn\_CTRLCHO.align* field. The following conditions apply to each setting:

Left aligned: *I2Sn\_CTRLCHO.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 data in the transmit FIFO is sent from the LSB to the number of bits plus 1.

Right aligned: `I2Sn_CTRL0CHO.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 and then 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 data in the transmit FIFO is sent from the LSB to the number of bits plus 1.

#### 14.6.6 Typical Audio Configurations

*Table 14-7:* shows the relationship between the bits per word field and the sample size field. *Equation 14-7* shows the required relationship between the sample size field and the bits per word field.

*Equation 14-7: Sample Size Relationship Bits per Word*

$$I2Sn_CTRL1CH0.smp\_size \leq I2Sn_CTRL1CH0.bits\_word$$

The `I2Sn_CTRL1CH0.bits_word` column in *Table 14-7* is set by the equation  $\frac{\# BCLK}{\text{Channel}} - 1$ . The `I2Sn_CTRL1CH0.smp_size` column is the number of samples per word captured from the I<sup>2</sup>S bus, and is calculated by the equation  $\frac{\# Samples}{\text{Channel}} - 1$ . Channel refers to the left and right channels of audio.

*Table 14-7: Configuration for Typical Audio Width and Samples per WS Clock Cycle*

Audio Sample Width/ Samples per WS Cycle	# BCLK Channel	# Samples Channel	I2Sn_CTRL1CH0			Sign extension (align = 1) <sup>†</sup>
			bits_word	smp_size	wsizer	
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	

<sup>†</sup>Sign Extension applies only when `I2Sn_CTRL0CHO.align` is set to 1 and `I2Sn_CTRL0CHO.smp_size` is less than the FIFO width size setting.

#### 14.7 Interrupt Events

The I<sup>2</sup>S peripheral generates interrupts for the events shown in *Table 14-8*. 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 14-8: I<sup>2</sup>S Interrupt Events*

Event	Interrupt Flag	Interrupt Enable
Receive FIFO overrun	<code>I2Sn_INTFL.rx_ov_ch0</code>	<code>I2Sn_INTEN.rx_ov_ch0</code>
Receive threshold	<code>I2Sn_INTFL.rx_thd_ch0</code>	<code>I2Sn_INTEN.rx_thd_ch0</code>
Transmit FIFO half-empty	<code>I2Sn_INTFL.tx_he_ch0</code>	<code>I2Sn_INTEN.tx_he_ch0</code>

Event	Interrupt Flag	Interrupt Enable
Transmit FIFO one byte remaining	<i>I2Sn_INFL.tx_ob_ch0</i>	<i>I2Sn_INEN.tx_ob_ch0</i>

#### 14.7.1 Receive FIFO Overrun

A receive FIFO overrun event occurs if the number of data words in the receive FIFO, *I2Sn\_DMACH0.rx\_lvl* is equal to the RX\_FIFO\_DEPTH and another word has been shifted into the FIFO. The hardware automatically sets the *I2Sn\_INFL.rx\_ov\_ch0* field to 1 when this event occurs.

#### 14.7.2 Receive FIFO Threshold

A receive FIFO threshold event occurs when a word is shifted in and the number of words in the receive FIFO, *I2Sn\_DMACH0.rx\_lvl*, exceeds the *I2Sn\_CTRLOCHO.rx\_thd\_val*. The event does not occur if the opposite transition occurs. When this event occurs, the hardware automatically sets the *I2Sn\_INFL.rx\_thd\_ch0* field to 1.

#### 14.7.3 Transmit FIFO Half-Empty

A transmit FIFO half-empty event occurs when the number of words in the transmit FIFO, *I2Sn\_DMACH0.tx\_lvl*, is less than  $\frac{1}{2}$  of the TX\_FIFO\_DEPTH as shown in *Equation 14-8*. When this event occurs, the *I2Sn\_INFL.tx\_he\_ch0* flag is set to 1 by the hardware.

*Note: The transmit FIFO half empty interrupt flag is set by the hardware one BCLK cycle prior to the actual condition occurring. If the BCLK is much slower than the I<sup>2</sup>S peripheral clock, the software may receive the interrupt while the actual transmit FIFO level is still equal to  $\frac{1}{2}$  of the TX\_FIFO\_DEPTH. The software should always read the transmit FIFO level prior to filling it to determine the correct number of words to write to the transmit FIFO. Read the level of the transmit FIFO using the *I2Sn\_DMACH0.tx\_lvl* field.*

*Equation 14-8: Transmit FIFO Half-Empty Condition*

$$I2Sn_DMACH0.tx_lvl < \left( \frac{\text{TX FIFO DEPTH}}{2} \right)$$

#### 14.7.4 Transmit FIFO One Entry Remaining

A transmit FIFO one entry remaining event occurs when the number of entries in the transmit FIFO is 1, *I2Sn\_DMACH0.tx\_lvl* = 1. When this event occurs, the *I2Sn\_INFL.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 prior to the actual condition occurring. If the BCLK is much slower than the I<sup>2</sup>S peripheral clock, the software may receive the interrupt while the actual transmit FIFO level is still equal to 2. The software should always read the transmit FIFO level prior to filling it to determine the correct number of words to write to the transmit FIFO. Read the level of the transmit FIFO using the *I2Sn\_DMACH0.tx\_lvl* field.*

### 14.8 Direct Memory Access

The I<sup>2</sup>S supports DMA for both transmit and receive; separate DMA channels can be connected to the receive and transmit FIFOs. The following describe 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  $\frac{1}{2}$  of the transmit FIFO's depth.

## 14.9 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<sup>2</sup>S serial communication. A typical software sequence is shown below.

1. Set *GCR\_PCLKDIS1.i2s0* to 0 to enable the I<sup>2</sup>S peripheral clock source shown in *Table 14-1*.
2. Disable the I<sup>2</sup>S clock by setting *I2Sn\_CTRL1CHO.en* to 0.
3. Set *I2Sn\_CTRL0CHO.rst* to 1 to reset the I<sup>2</sup>S configuration.
4. Set *I2Sn\_CTRL1CHO.flush* to 1 to flush the FIFO buffers.
5. Configure the *I2Sn\_CTRL0CHO.ch\_mode* to select the master or slave configuration.
  - a. For master mode, configure the baud rate by programming the *I2Sn\_CTRL1CHO.clkdiv* field to achieve the required bit rate, set the *I2Sn\_CTRL1CHO.smp\_size* field to the desired sample size of the data, and the *I2Sn\_CTRL1CHO.adjsf* 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 *I2Sn\_CTRL1CHO.rx\_thd*. The threshold of the transmit FIFO 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 *I2Sn\_INTEN* register if desired.
9. Program the *clkdiv* bits in *I2Sn\_CTRL1CHO* register for the new bit clock frequency.
10. For master operation, load data in the transmit FIFO for transmit.
11. Re-enable the bit clock by setting *I2Sn\_CTRL1CHO.en* to 1.

## 14.10 Registers

See *Table 2-4* for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in *Table 14-9*. Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register *PERIPHERALn\_CTRL* resolves to *PERIPHERAL0\_CTRL* and *PERIPHERAL1\_CTRL* for instances 0 and 1, respectively.

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

*Table 14-9: I<sup>2</sup>S Register Summary*

Offset	Register Name	Description
[0x0000]	<i>I2Sn_CTRL0CHO</i>	I <sup>2</sup> S Global Mode Control 0 Register
[0x0010]	<i>I2Sn_CTRL1CHO</i>	I <sup>2</sup> S Master Mode Configuration Register
[0x0030]	<i>I2Sn_DMACHO</i>	I <sup>2</sup> S DMA Control Channel Register
[0x0040]	<i>I2Sn_FIFOCHO</i>	I <sup>2</sup> S FIFO Register
[0x0050]	<i>I2Sn_INTFL</i>	I <sup>2</sup> S Interrupt Status Register
[0x0054]	<i>I2Sn_INTEN</i>	I <sup>2</sup> S Interrupt Enable Register

### 14.10.1 Register Details

*Table 14-10: I<sup>2</sup>S Control 0 Register*

I <sup>2</sup> S Control 0 Register			I2Sn_CTRL0CHO		0x0000
Bits	Field	Access	Reset	Description	
31:24	rx_thd_val	R/W	0	<b>RX FIFO Interrupt Threshold</b> 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.	

I <sup>2</sup> S Control 0 Register			I2Sn_CTRL0CHO		0x0000
Bits	Field	Access	Reset	Description	
23:21	-	RO	0	<b>Reserved</b>	
20	fifo_lsb	R/W	0	<b>FIFO Bit Field Control</b> Only used if the FIFO size is larger than the sample size and <i>I2Sn_CTRL0CHO.align</i> = 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	<b>Reset</b> Write 1 to reset the I <sup>2</sup> 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	<b>FIFO Flush</b> Write 1 to start a flush of the receive and transmit FIFOs. 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	<b>Receive Enable</b> Enable receive mode for the I <sup>2</sup> S peripheral. 0: Disabled 1: Enabled	
16	tx_en	R/W	0	<b>Transmit Enable</b> Enable transmit mode for the I <sup>2</sup> S peripheral. 0: Disabled 1: Enabled	
15:14	wsize	R/W	0x3	<b>Data Size When Reading/Writing FIFO</b> 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).	
13:12	stereo	R/W	0	<b>I<sup>2</sup>S Mode</b> Select the mode for the I <sup>2</sup> 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	<b>Reserved</b>	
10	align	R/W	0	<b>FIFO Data Alignment</b> Set this field to control the alignment of the data in the FIFOs. This field is only used if the FIFO data width, <i>I2Sn_CTRL0CHO.wsize</i> , is not equal to the bits per word field. 0: MSB 1: LSB	

I <sup>2</sup> S Control 0 Register				I2Sn_CTRL0CHO	0x0000
Bits	Field	Access	Reset	Description	
9	msb_loc	R/W	0	<b>First Bit Location Sampling</b> This field controls when the first bit is transmitted/received in relation to the LRCLK. By default, the first bit is transmitted/received on SDO/SDI on the second complete LRCLK cycle. 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	<b>LRCLK Polarity Select</b> 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 <sup>2</sup> S association. 0: LRCLK low for left channel. 1: LRCLK high for left channel.	
7:6	ch_mode	R/W	0	<b>Mode</b> Set this field to indicate master or slave I <sup>2</sup> S operation. When using master mode, the I <sup>2</sup> S external clock must be used to generate the LRCLK/BCLK signals. 0: Master mode, internal generation of LRCLK/BCLK using the I <sup>2</sup> S external clock input. 1-2: Reserved 3: Slave mode, external generation of LRCLK/BCLK.	
5:2	-	DNM	0	<b>Reserved, Do Not Modify</b>	
1	lsb_first	R/W	0	<b>LSB First</b> 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	<b>Reserved</b>	

Table 14-11: I<sup>2</sup>S Master Mode Configuration Register

I <sup>2</sup> S Master Mode Configuration				I2Sn_CTRL1CHO	0x0010
Bits	Field	Access	Reset	Description	
31:16	clkdiv	R/W	0	<b>I<sup>2</sup>S Frequency Divisor</b> Set this field to the required divisor to achieve the desired frequency for the I <sup>2</sup> S BCLK. See <a href="#">BCLK Generation for Master Mode</a> for detailed information. <i>Note: This field only applies when the I<sup>2</sup>S peripheral is set to master mode, I2Sn_CTRL0CHO.ch_mode = 0.</i>	
15	adjst	R/W	0	<b>Data Justification When Sample Size is Less than Bits Per Word</b> 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	<b>Reserved</b>	

I <sup>2</sup> S Master Mode Configuration			I2Sn_CTRL1CHO		0x0010
Bits	Field	Access	Reset	Description	
13:9	smp_size	R/W	0	<b>Sample Size</b> 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 number of bits are required by the application and this field allows flexibility. An example use case would be for 16-bit audio being received and the application only needs 8-bits of resolution. See <a href="#">Sample Size</a> for additional details. <i>Note: The sample size is equal to I2Sn_CTRL1CHO.bits_word when I2Sn_CTRL1CHO.smp_size = 0 or I2Sn_CTRL1CHO.smp_size &gt; I2Sn_CTRL1CHO.bits_word.</i>	
8	en	R/W	0	<b>I<sup>2</sup>S Enable</b> For master mode operation, this field is used to start the generation of the I <sup>2</sup> S LRCLK and BCLK outputs. In slave mode, this field enables the peripheral to begin receiving signals on the I <sup>2</sup> S interface. 0: Disabled 1: Enabled	
7:5	-	RO	0	<b>Reserved</b>	
4:0	bits_word	R/W	0	<b>I<sup>2</sup>S Word Length</b> This field is defined as the I <sup>2</sup> S data bits per left and right channel. <i>Example: If the bit clocks is 16 per half frame, bits_word is 15.</i>	

Table 14-12: I<sup>2</sup>S DMA Control Register

I <sup>2</sup> S DMA Control			I2Sn_DMACHO		0x0030
Bits	Field	Access	Reset	Description	
31:24	rx_lvl	RO	0	<b>Receive FIFO Level</b> This field is the number of data words in the receive FIFO.	
23:16	tx_lvl	RO	0	<b>Transmit FIFO Level</b> This field is the number of data words in the transmit FIFO.	
15	dma_rx_en	RW	0	<b>DMA Receive Channel Enable</b> 0: Disabled 1: Enabled	
14:8	dma_rx_thd_val	RW	0	<b>DMA Receive FIFO Event Threshold</b> 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	RW	0	<b>DMA Transmit Channel Enable</b> 0: Disabled 1: Enabled	
6:0	dma_tx_thd_val	RO	0	<b>DMA Transmit FIFO Event Threshold</b> 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 14-13: I<sup>2</sup>S FIFO Register

I <sup>2</sup> S FIFO Register			I2Sn_FIFOCH0		0x0040
Bits	Field	Access	Reset	Description	
31:0	data	R/W	0	<b>I<sup>2</sup>S FIFO</b> Writing to this field loads the next character into the transmit FIFO and increments the <i>I2Sn_DMACH0.tx_lvl</i> . Writes are ignored if the transmit FIFO is full. Reads of this field return the next character available from the receive FIFO and decrements the <i>I2Sn_DMACH0.rx_lvl</i> . The value 0 is returned if <i>I2Sn_DMACH0.rx_lvl</i> = 0.	

 Table 14-14: I<sup>2</sup>S Interrupt Flag Register

I <sup>2</sup> S Interrupt Flag			I2Sn_INTFL		0x0050
Bits	Field	Access	Reset	Description	
31:4	-	DNM	0	<b>Reserved, Do Not Modify</b>	
3	tx_he_ch0	W1C	0	<b>Transmit FIFO Half-Empty Event Interrupt Flag</b> 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	<b>Transmit FIFO One Entry Remaining Event Interrupt Flag</b> 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	<b>Receive FIFO Threshold Event Interrupt Flag</b> 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	<b>Receive FIFO Overrun Event Interrupt Flag</b> If this field is set to 1, the event has occurred. Write 1 to clear. 0: No event 1: Event occurred	

 Table 14-15: I<sup>2</sup>S Interrupt Enable Register

I <sup>2</sup> S Interrupt Enable			I2Sn_INTEN		0x0054
Bits	Field	Access	Reset	Description	
31:4	-	DNM	0	<b>Reserved, Do Not Modify</b>	
3	tx_he_ch0	R/W	0	<b>Transmit FIFO Half-Empty Event Interrupt Enable</b> Set this field to 1 to enable interrupts for this event. 0: Disabled 1: Enabled	
2	tx_ob_ch0	R/W	0	<b>Transmit FIFO One Entry Remaining Event Interrupt Enable</b> Set this field to 1 to enable interrupts for this event. 0: Disabled 1: Enabled	
1	rx_thd_ch0	R/W	0	<b>Receive FIFO Threshold Event Interrupt Enable</b> Set this field to 1 to enable interrupts for this event. 0: Disabled 1: Enabled	

I <sup>2</sup> S Interrupt Enable				I2Sn_INTEN	0x0054
Bits	Field	Access	Reset	Description	
0	rx_ov_ch0	R/W	0	<b>Receive FIFO Overrun Event Interrupt Enable</b> Set this field to 1 to enable interrupts for this event. 0: Disabled 1: Enabled	

## 15. 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.

### 15.1 Instances

There is one instance of the CAMERAIF, shown in [Table 15-1](#). The pins and alternate functions for the CAMERAIF are shown in [Table 15-2](#).

*Table 15-1: MAX78000 CAMERAIF Instances*

Instance	CAMERAIF Peripheral Clock Options	RX FIFO Depth
PCIF	PCLK	8

*Table 15-2: MAX78000 CAMERAIF Signals*

Signal Name	81-CTBGA Pin	Alternate Function	Signal Direction	Description
PCIF_PCLK	P1.9	AF1	Input	Pixel Clock Input
PCIF_HSYNC	P1.8	AF1	Input	Horizontal Synchronization Input
PCIF_VSYNC	P0.15	AF2	Input	Vertical Synchronization Input
PCIF_D0	P0.20	AF2	Input	Pixel Data Input 0
PCIF_D1	P0.21	AF2	Input	Pixel Data Input 1
PCIF_D2	P0.22	AF2	Input	Pixel Data Input 2
PCIF_D3	P0.23	AF2	Input	Pixel Data Input 3
PCIF_D4	P0.24	AF2	Input	Pixel Data Input 4
PCIF_D5	P0.25	AF2	Input	Pixel Data Input 5
PCIF_D6	P0.26	AF2	Input	Pixel Data Input 6
PCIF_D7	P0.27	AF2	Input	Pixel Data Input 7
PCIF_D8	P0.30	AF2	Input	Pixel Data Input 8
PCIF_D9	P0.31	AF2	Input	Pixel Data Input 9
PCIF_D10	P1.6	AF2	Input	Pixel Data Input 10
PCIF_D11	P1.7	AF2	Input	Pixel Data Input 11

## 15.2 Capture Modes

The CAMERAIF supports either single image capture mode or continuous capture mode. Each mode and the CAMERAIF configuration is described in the following sections.

### 15.2.1 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 `CAMERAIFn_CTRL.read_mode` field to 1. The `CAMERAIFn_CTRL.read_mode` field remains set prior to and while receiving image data from the camera. Once the image is complete, the hardware automatically sets the `CAMERAIFn_CTRL.read_mode` field to 0 and sets the `CAMERAIFn_INTFL.img_done` status to 1.

### 15.2.2 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 `CAMERAIFn_CTRL.read_mode` field to 2. Disable continuous mode capture by setting the `CAMERAIFn_CTRL.read_mode` field to 0.

## 15.3 Timing Modes

There are two different timing modes. Horizontal and Vertical Synchronization Mode and Data Streaming Mode. Both timing modes can be combined with single image capture or continuous capture read modes.

### 15.3.1 Horizontal and Vertical Synchronization Timing Mode

In this timing mode the 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 for detection of the start of video signal. The PCIF\_HSYNC signal is used to frame a complete set of pixel data. Re-assertion of the PCIF\_VSYNC signal indicates to the CAMERAIF that the image is complete.

Set the bit `CAMERAIFn_CTRL.ds_timing_sel` to 0 to configure the CAMERAIF for Horizontal and Vertical Synchronization mode.

### 15.3.2 Data Stream Timing Mode

In this timing mode the PCIF\_HSYNC and PCIF\_VSYNC input pins are ignored. The 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 Start Active Video (SAV) code (`CAMERAIFn_DS_TIMING_CODES.sav`) and the End Active Video (EAV) code (`CAMERAIFn_DS_TIMING_CODES.eav`). These two codes must match the codes sent by the connected camera respectively and cannot be identical. Set `CAMERAIFn_CTRL.ds_timing_sel` to 1 to configure the CAMERAIF for embedded timing codes mode.

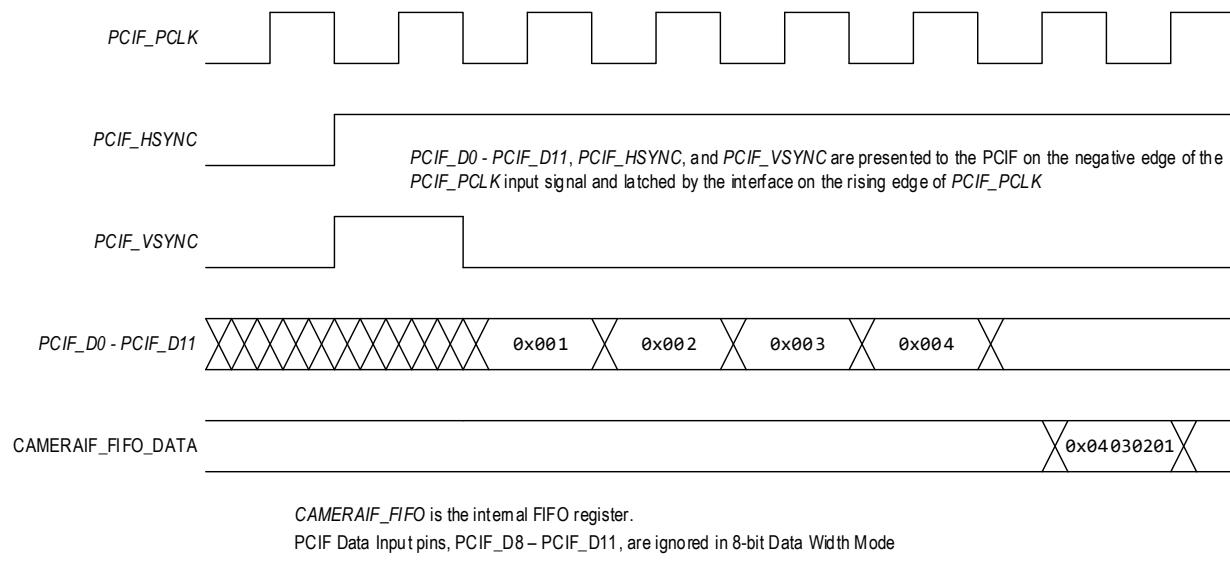
## 15.4 Data Width

The width of the pixel data can be configured as 8-bit, 10-bit or 12-bit. Pixel data is read from the PCIF\_Dx input pins on the rising edge of the PCIF\_PCLK input pixel clock. It is assumed that PCIF\_Dx changes on the negative edge of PCIF\_PCLK.

### 15.4.1 8-bit Width

Setting `CAMERAIFn_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 as having the most significant byte the most recently received 8-bit PCIF\_Dx data. See [Figure 15-1](#) and [Figure 15-2](#) examples.

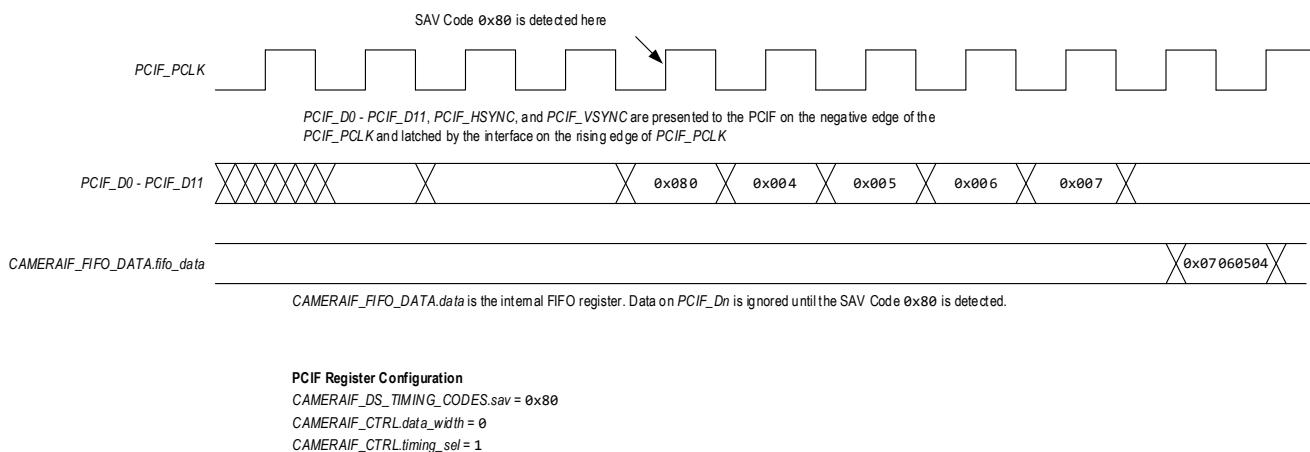
*Figure 15-1: Horizontal and Vertical Synchronization Timing Mode with 8-Bit Data Width*



#### PCIF Register Configuration

`CAMERAIF_CTRL.data_width = 0`  
`CAMERAIF_CTRL.timing_sel = 0`

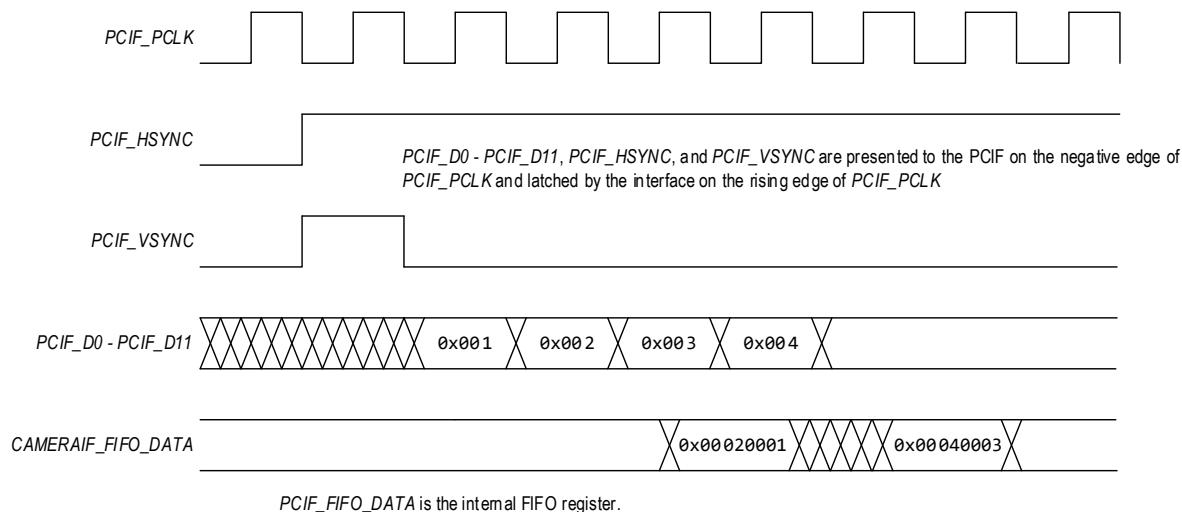
*Figure 15-2: Data Stream Timing Mode with 8-Bit Data Width*



#### 15.4.2 10 and 12-bit Width

Setting `CAMERAIFn_CTRL.data_width` to 1 sets the recognized pixel width on the PCIF\_Dx bus to 10-bits. Setting `CAMERAIFn_CTRL.data_width` to 2 sets the recognized pixel width on the PCIF\_Dx bus to 12-bits. As with the 8-bit width setting, the pixel data is framed as 32-bit words before these words are transferred to the 32-bit wide data FIFO `CAMERAIFn_FIFO_DATA` and made ready to be read. These pixel widths are MSB zero padded to 16-bits and then two 16-bit pixels are concatenated to form the 32-bit word, the most recently received PCIF\_Dx data is the most significant 16-bits of the FIFO data. See [Figure 15-3](#) for PCIF\_VSYNC/PCIF\_HSYNC Timing example.

Figure 15-3: 10 or 12-bit PCIF\_VSYNC/PCIF\_HSYNC



#### PCIF Register Configuration

`CAMERAIF_CTRL.data_width = 1 or 2`  
`CAMERAIF_CTRL.timing_sel = 0`

## 15.5 Data FIFO

The data FIFO `CAMERAIFn_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 `CAMERAIFn_CTRL fifo_thresh`. The `CAMERAIFn_INTFL fifo_thresh` is set if the data FIFO depth becomes greater than or equal to `CAMERAIFn_CTRL fifo_thresh`. An interrupt can be generated when this condition happens if `CAMERAIFn_INTEN fifo_thresh` is set. The data FIFO also provides status flags for FIFO Full (`CAMERAIFn_INTFL fifo_full`) and FIFO Not Empty (`CAMERAIFn_INTFL fifo_not_empty`). Both status flags have associated interrupts (`CAMERAIFn_INTEN fifo_full` and `CAMERAIFn_INTEN fifo_not_empty`) that can be enabled and triggered when the status flags become set.

## 15.6 Usage

### 15.6.1 DMA

1. Set `CAMERAIFn_CTRL.data_width` and `CAMERAIFn_CTRL.ds_timing_sel` as required by the camera sensor attached.
2. Enable the `CAMERAIFn_INTEN.img_done` to generate an interrupt once the image is complete.
3. Set `CAMERAIFn_CTRL.read_mode` for single image or continuous capture. Triggering the camera sensor to output an image starts the PCI automatically.
4. Set the `CAMERAIFn_CTRL.rx_dma_thresh` field to the desired FIFO level required to trigger a DMA threshold event.
5. Enable the receive DMA by setting the `CAMERAIFn_CTRL.rx_dma_en` field to 1.
6. Enable the CAMERAIF by setting the `CAMERAIFn_CTRL.en` 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 `CAMERAIFn_INTEN.img_done` interrupt.
8. The interrupt handler can then reset the interrupt flag by writing 1 to `CAMERAIFn_INTFL.img_done`.

### 15.6.2 Interrupts

1. Set `CAMERAIFn_CTRL.data_width` and `CAMERAIFn_CTRL.ds_timing_sel` as required by the camera sensor attached.
2. Set `CAMERAIFn_CTRL fifo_thresh` to the desired level to allow the interrupt to service the FIFO before it fills.
3. Enable the `CAMERAIFn_INTEN.img_done` and the `CAMERAIFn_INTEN fifo_thresh` interrupts, generating an interrupt when the image is complete or the FIFO has been filled to the threshold level set in the `CAMERAIFn_CTRL fifo_thresh` field.
4. Set `CAMERAIFn_CTRL.read_mode` for 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 `CAMERAIFn_CTRL.en` field to 1.
6. As data is read from the camera sensor by the PCIF, the hardware triggers an ISQ whenever the FIFO threshold `CAMERAIFn_CTRL fifo_thresh` has been met. The interrupt handler should perform a burst read from the FIFO (`CAMERAIFn_FIFO_DATA fifo_data`). When the camera sensor finishes transmitting image data, signaled either by a rising edge on PCIF\_VSYNC or a Data Stream EAV code, the hardware generates an `CAMERAIFn_INTEN.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 `CAMERAIFn_INTFL.img_done`.
8. Software should check `CAMERAIFn_INTFL fifo_not_empty` and perform a read of `CAMERAIFn_FIFO_DATA.data` to receive the remainder of the words of data that occupy the FIFO less than `CAMERAIFn_CTRL fifo_thresh`. When all of the data is read from the FIFO, hardware clears the `CAMERAIFn_INTFL fifo_not_empty` flag automatically.

## 15.7 Camera Registers

See [Table 2-4](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own independent set of the registers shown in [Table 15-3](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register `PERIPHERALn_CTRL` resolves to `PERIPHERAL0_CTRL` and `PERIPHERAL1_CTRL` for instances 0 and 1, respectively.

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

*Table 15-3: Parallel Camera Interface Register Summary*

Offset	Register	Name
[0x0000]	<code>CAMERAIFn_VER</code>	CAMERAIF Revision Register
[0x0004]	<code>CAMERAIFn_FIFO_SIZE</code>	CAMERAIF FIFO Size Register
[0x0008]	<code>CAMERAIFn_CTRL</code>	CAMERAIF Configuration Register
[0x000C]	<code>CAMERAIFn_INTEN</code>	CAMERAIF Interrupt Enable Register
[0x0010]	<code>CAMERAIFn_INTFL</code>	CAMERAIF Status Flag Register
[0x0014]	<code>CAMERAIFn_DS_TIMING_CODES</code>	CAMERAIF Timing Code Register
[0x0030]	<code>CAMERAIFn_FIFO_DATA</code>	CAMERAIF FIFO Data Register

### 15.7.1 Parallel Camera Register Details

*Table 15-4: CAMERAIF Version Register*

CAMERAIF Version		CAMERAIFn_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.	

CAMERAIF Version		CAMERAIFn_VER			[0x0000]
Bits	Field	Access	Reset	Description	
7:0	minor	RO	*	<b>Minor Revision</b> This field returns the minor revision number of the CAMERAIF.	

Table 15-5: CAMERAIF FIFO Size Register

CAMERAIF FIFO Size		CAMERAIFn_FIFO_SIZE			[0x0004]
Bits	Field	Access	Reset	Description	
31:8	-	RO	0	<b>Reserved</b>	
7:0	fifo_size	RO	8	<b>FIFO Size</b> This field returns the size of the CAMERAIF FIFO in words. 8: FIFO size is 8 words	

Table 15-6: CAMERAIF Configuration Register

CAMERAIF Configuration		CAMERAIFn_CTRL			[0x0008]
Bits	Field	Access	Reset	Description	
31	en	R/W	0	<b>Camera Interface Enable</b> Set this field to 1 to enable the Camera interface. 0: Camera interface disabled 1: Camera interface enabled	
30	cnnmode	R/W	0	<b>CNN Mode Enable</b> Set this field to enable CNN mode operation. 0: CNN mode disabled 1: CNN mode enabled	
29:15	-	RO	0	<b>Reserved</b>	
14:11	rx_dma_thresh	R/W	1	<b>DMA Threshold</b> 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. <i>Note: This field is only used if the <a href="#">CAMERAIFn_CTRL.rx_dma_en</a> is set to 1.</i> 0: Invalid, do not set this field to 0 1: Receive DMA threshold event occurs when the FIFO level is greater than or equal to 1 ... ... 8: Receive DMA threshold event occurs when the FIFO level is equal to 8	
10	rx_dma_en	R/W	0	<b>Receive DMA Enable</b> 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	

CAMERAIF Configuration		CAMERAIFn_CTRL			[0x0008]
Bits	Field	Access	Reset	Description	
9:5	fifo_thresh	R/W	1	<b>Data FIFO Threshold Setting</b> If the number of words in the FIFO is greater than or equal to this value the <a href="#">CAMERAIFn_INTFL fifo_thresh</a> 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_sel	R/W	0	<b>Camera Timing Select</b> 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 start of video (SAV) and end of video (EAV) codes.	
3:2	data_width	R/W	0	<b>Camera Data Width</b> 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 <i>Note: Unused PCIF_Dx pins are ignored.</i>	
1:0	read_mode	R/W	0	<b>Camera Read Mode</b> Set this field to the required camera read mode or set to 0 to disable the CAMERAIF. 0: Disabled 1: Single image capture 2: Continuous capture 3: Reserved	

Table 15-7: CAMERAIF Interrupt Enable Register

CAMERAIF Interrupt Enable		CAMERAIFn_INTEN			[0x000C]
Bits	Field	Access	Reset	Description	
31:4	-	RO	0	<b>Reserved</b>	
3	fifo_not_empty	R/W	0	<b>FIFO Not Empty Interrupt Enable</b> Set this field to 1 to generate an interrupt when the FIFO is not empty ( <a href="#">CAMERAIFn_INTFL fifo_not_empty = 1</a> ), indicating data is available to read from the FIFO. 0: Interrupt disabled 1: Interrupt enabled	
2	fifo_thresh	R/W	0	<b>FIFO Threshold Interrupt Enable</b> Set this field to 1 to generate an interrupt when the FIFO threshold is reached ( <a href="#">CAMERAIFn_INTFL fifo_thresh = 1</a> ). 0: Interrupt Disabled 1: Interrupt Enabled	

CAMERAIF Interrupt Enable			CAMERAIFn_INTEN		[0x000C]
Bits	Field	Access	Reset	Description	
1	fifo_full	R/W	0	<b>FIFO Full Interrupt Enable</b> Set this bit to 1 to generate an interrupt when the FIFO is full ( <i>CAMERAIFn_INTFL fifo_full</i> = 1). 0: Interrupt Disabled 1: Interrupt Enabled	
0	img_done	R/W	0	<b>Image Complete Interrupt Enable</b> Set this bit to 1 to generate an interrupt when the image is done ( <i>CAMERAIFn_INTFL img_done</i> = 1). 0: Interrupt Disabled 1: Interrupt Enabled	

Table 15-8: CAMERAIF Status Flags Register

CAMERAIF Status Flags			CAMERAIFn_INTFL		[0x0010]
Bits	Field	Access	Reset	Description	
31:4	-	RO	0	<b>Reserved</b>	
7	fifo_not_empty_raw	RO	0	<b>FIFO Not Empty Status Flag Raw</b> This field is the raw status bit as set by hardware prior to being passed through the APB interface. 0: The FIFO is empty 1: The FIFO is not empty	
6	fifo_thresh_raw	RO	0	<b>FIFO Threshold Status Flag</b> This status is set by hardware when the FIFO level is greater than or equal to the <i>CAMERAIFn_CTRL fifo_thresh</i> 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	
5	fifo_full	RO		<b>FIFO Full Status Flag</b> 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	
4	img_done	RO		<b>Image Complete Status Flag</b> This status is set by hardware when either the PCIF_VSYNC device pin has transitioned logic level during a triggered camera sensor read, or the EAV code, <i>CAMERAIFn_DS_TIMING_CODES.eav</i> , is detected. 0: End of image not detected 1: End of image detected	
3	fifo_not_empty	RO	0	<b>FIFO Not Empty Status Flag</b> 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	

CAMERAIF Status Flags			CAMERAIFn_INTFL		[0x0010]
Bits	Field	Access	Reset	Description	
2	fifo_thresh	RO	0	<b>FIFO Threshold Status Flag</b> This status is set by hardware when the FIFO level is greater than or equal to the <a href="#">CAMERAIFn_CTRL.fifo_thresh</a> 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	<b>FIFO Full Status Flag</b> 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	<b>Image Complete Status Flag</b> 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, <a href="#">CAMERAIFn_DS_TIMING_CODES.eav</a> , is detected. 0: End of image not detected 1: End of image detected	

Table 15-9: CAMERAIF Timing Codes Register

CAMERAIF Camera Timing Codes			CAMERAIFn_DS_TIMING_CODES		[0x0014]
Bits	Field	Access	Reset	Description	
31:16	-	RO	0	<b>Reserved</b>	
15:8	eav	R/W	0x9D	<b>End Active Video</b> The end active video field is an 8-bit code that is camera-dependent. This value cannot be equal to <a href="#">CAMERAIFn_DS_TIMING_CODES.sav</a> . 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	<b>Start Active Video</b> The start active video field is an 8-bit code that is camera-dependent. This value cannot be equal to <a href="#">CAMERAIFn_DS_TIMING_CODES.eav</a> . Set this field to the camera's start active video field, which may differ from the reset default of 0x80.	

Table 15-10: CAMERAIF FIFO Data Register

CAMERAIF FIFO Data			CAMERAIFn_FIFO_DATA		[0x0030]
Bits	Field	Access	Reset	Description	
31:0	fifo_data	R	0	<b>Data</b> Data from the FIFO to be read. Once read, the next value in the FIFO becomes immediately available to read.	

## 16. 1-Wire Master (OWM)

The device provides a 1-Wire master (OWM) that 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.

### 16.1 1-Wire Master Features

The OWM provides the following features:

- Flexible, 1-Wire timing generation (required 1MHz timing base) using the OWM module clock frequency, which is in turn derived from the current system clock source. 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 firmware drive) modes.
- 1-Wire reset generation and presence-pulse detection.
- Generation of 1-Wire read and write time slots for single-bit and eight-bit byte transmissions.
- Search ROM Accelerator (SRA) mode, which simplifies the generation of multiple-bit time slots and discrepancy resolution required when completing the Search ROM function to determine the IDs of multiple, unknown 1-Wire slaves on the bus.
- Transmit data completion, received data available, presence pulse detection, and 1-Wire line-error condition interrupts.

For more information about the Maxim 1-Wire protocol and supporting devices, refer to the following resources:

- *AN937: The Book of iButton Standards*
  - ◆ [www.maximintegrated.com/AN937](http://www.maximintegrated.com/AN937)
- *AN1796: Overview of 1-Wire Technology and its Use*
  - ◆ [www.maximintegrated.com/AN1796](http://www.maximintegrated.com/AN1796)
- *AN187: 1-Wire Search Algorithm*
  - ◆ [www.maximintegrated.com/AN187](http://www.maximintegrated.com/AN187)

*iButton* is a registered trademark of Maxim Integrated Products, Inc.

### 16.2 1-Wire Pins and Configuration

The one instance of the peripheral shown in *Table 16-1* lists the location of the OWM\_IO and OWM\_PE signals.

*Table 16-1: MAX78000 1-Wire Master Peripheral Pins*

OWMn	ALTERNATE FUNCTION	ALT FUNCTION #	81-CTBGA
OWM0	OWM_IO	AF1	P0.6
	OWM_PE	AF1	P0.7

### 16.2.1 1-Wire I/O (OWM\_IO)

The OWM\_IO pin is a bidirectional I/O that is used to directly drive the external 1-Wire bus. As described in the *Book of iButton Standards*, this I/O is generally driven as an open-drain output. The 1-Wire bus requires a common pullup to return the 1-Wire bus line to an idle high state when no master or slave device is actively driving the line low. This pullup can consist of a fixed resistor pullup (connected to the 1-Wire bus outside the microcontroller), an internal pullup enabled by setting *OWM\_CFG.int\_pullup\_enable* to 1, or an OWM module controlled external pullup enabled by setting *OWM\_CFG.ext\_pullup\_mode* to 1.

### 16.2.2 Pullup Enable (OWM\_PE)

The 1-Wire pullup enable (PE) signal is an active high output used to enable an optional external pullup on the 1-Wire bus. This pullup is intended to provide a stronger (lower impedance) pullup on the 1-Wire bus under certain circumstances, such as during overdrive mode.

### 16.2.3 Clock Configuration

To correctly generate the timing required by the 1-Wire protocol in Standard or Overdrive timing modes, the OWM clock must be set to achieve  $f_{owmclk} = 1\text{MHz}$ . This clock generates both the Standard and Overdrive timing, so it does not need adjustment when transitioning from Standard to Overdrive mode or vice versa.

The OWM peripheral uses the system peripheral clock, PCLK, divided by the value in the *OWM\_CLK\_DIV\_1US.divisor* field as shown in *Equation 16-1* where  $f_{PCLK} = f_{SYSCLK}/2$ .

*Equation 16-1: OWM 1MHz Clock Frequency*

$$f_{owmclk} = 1\text{MHz} = \frac{f_{PCLK}}{\text{OWM\_CLK\_DIV\_1US}.divisor}$$

## 16.3 1-Wire Protocol

The general timing and communication protocols used by the OWM interface are those standardized for the 1-Wire network.

Because the 1-Wire interface is a master interface, it initiates and times all communication on the 1-Wire bus. Except for the present pulse generation when a device first connects to the 1-Wire bus, 1-Wire slave devices complete 1-Wire bus communication only as directed by the 1-Wire bus master. From a firmware perspective, the lowest-level timing and electrical details of how the 1-Wire network operates are unimportant. The application can configure the OWM module properly and direct it to complete low-level operations such as reset, read, and write bit/byte operations. Thus, the OWM module on the microcontroller is designed to interface to the 1-Wire bus at a low level.

### 16.3.1 Networking Layers

In the *Book of iButton Standards*, the 1-Wire communication protocol is described in terms of the ISO-OSI model (International Organization of Standardization (ISO) Open System Interconnection (OSI) network layer model). Network layers that apply to this description are the Physical, Link, Network, and Transport layers. The Transport layer consists of the software that transfers memory data other than ROM ID contents to and from the individual 1-Wire network nodes. The Presentation layer 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

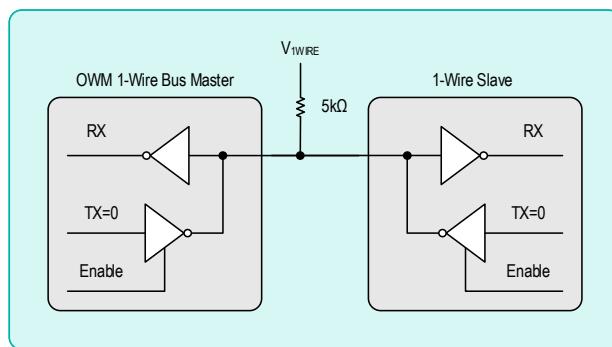
and Network layers with regards to the OSI network layer model. The Transport and Presentation layers are beyond the scope of this document.

### **16.3.1.1 Physical Layer**

The 1-Wire communication bus consists of a single data/power line plus ground. Devices (either master or slave) that interface to the 1-Wire communication bus using an open-drain (active low) connection, which means that the 1-Wire bus normally idles in a high state.

An external pullup resistor is used to pull the 1-Wire line high when no master or slave device is driving the line. This means that 1-Wire devices do not actively drive the 1-Wire line high. Instead, they either drive the line low or release it (set their output to high impedance) to allow the external resistor to pull the line high. This allows the 1-Wire bus to operate in a wired-AND manner as shown in *Figure 16-1* and avoids bus contention if more than one device attempts to drive the 1-Wire bus at the same time.

*Figure 16-1: 1-Wire Signal Interface*



### **16.3.1.2 Link Layer**

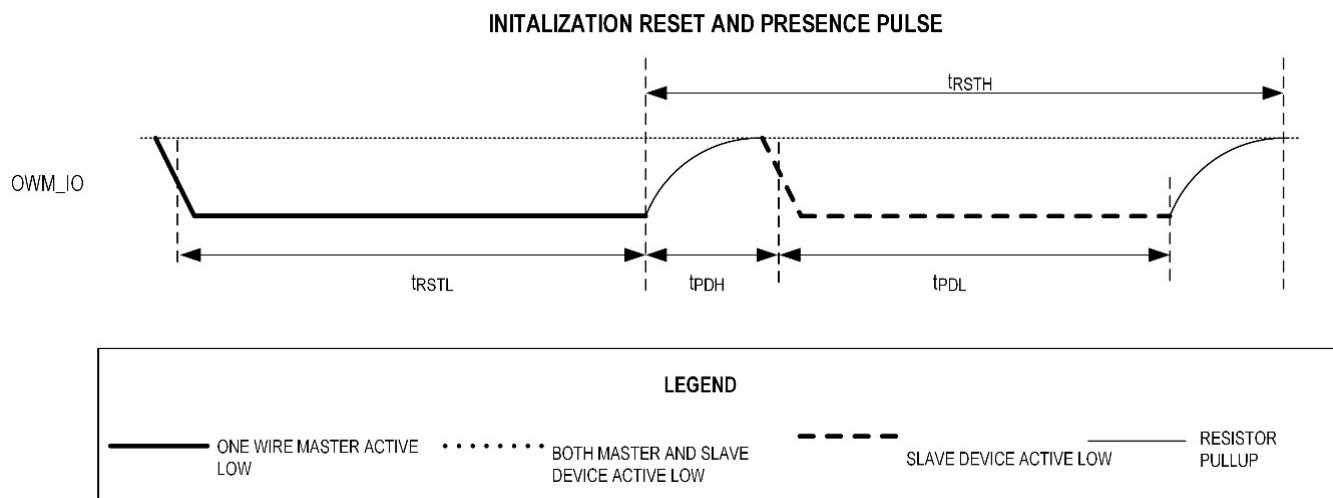
The 1-Wire Bus supports a single master and one or more slave devices (multidrop). Slave devices can connect to and disconnect from the 1-Wire Bus dynamically (as is typically the case with iButton devices that operate using an intermittent touch contact interface), which means that it is the master's responsibility to poll the bus as needed to determine the number and types of 1-Wire devices that are connected to the bus.

All communication sequences on the 1-Wire Bus are initiated by the OWM. The OWM determines when 1-Wire data transmissions begin, as well as the overall communication speed that is used. There are three different communication speeds supported by the 1-Wire specification: standard speed, overdrive speed, and hyperdrive speed. However, only standard speed and overdrive speed are supported by the OWM peripheral in the devices.

#### **16.3.1.2.1 OWM Reset and Presence Detect**

The OWM begins each communication sequence by sending a reset pulse as shown in *Figure 16-2*. This pulse resets all 1-Wire slave devices on the line to their initial states and causes them all to begin monitoring the line for a command from the OWM. Each 1-Wire slave device on the line responds to the reset pulse by sending out a presence pulse. These pulses from multiple 1-Wire slave devices are combined in wired-AND fashion, resulting in a pulse whose length is determined by the slowest 1-Wire slave device on the bus.

*Figure 16-2: 1-Wire Reset Pulse*



In general, the 1-Wire line must idle in a high state when communication is not taking place. It is possible for the master to pause communication in between time slots. There is not an overall "timeout" period that causes a slave to revert to the reset state if the master takes too long between one time slot and the next time slot.

The 1-Wire communication protocol relies on the fact that the maximum allowable length for a bit transfer (write 0/1 or read bit) time slot is less than the minimum length for a 1-Wire reset. At any time, if the 1-Wire line is held low (by the master or by any slave device) for more than the minimum reset pulse time, all slave devices on the line interpret this as a 1-Wire reset pulse.

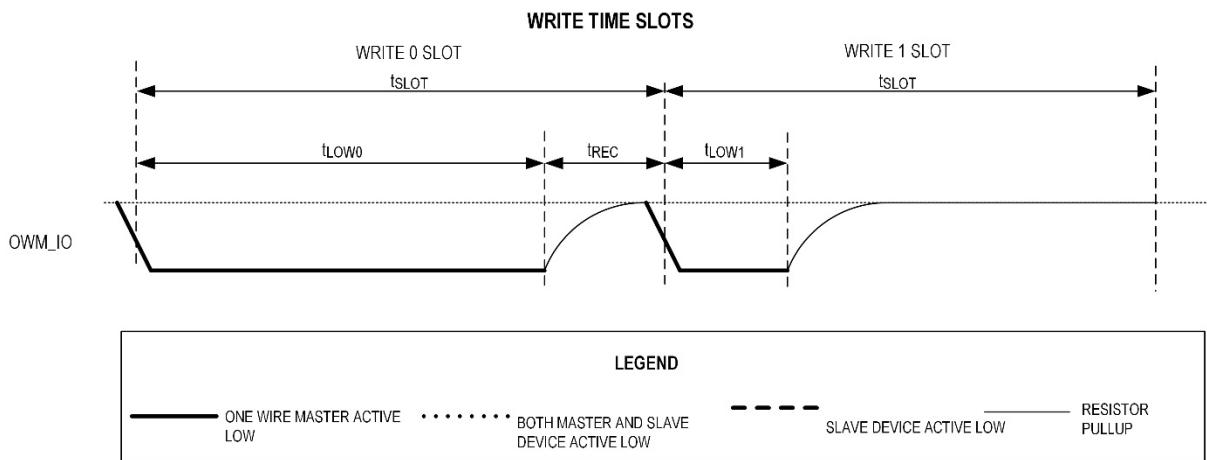
#### 16.3.1.2.2 OWM Write Time Slot

All 1-Wire bit time slots are initiated by the 1-Wire bus master and begin with a single falling edge. There is no indication given by the beginning of a time slot 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 16-3*, the time slots for writing a 0 bit and writing a 1 bit begin identically, with the falling edge and a minimum-width low pulse sent by the master. To write a one bit, the master releases the line after the minimum low pulse, allowing it to be pulled high. To write a zero bit, the master continues to hold the line low until the end of the time slot.

*Figure 16-3: 1-Wire Write Time Slot*



From the slave's perspective, the initial falling edge of the time slot triggers the start of an internal timer, and when the proper amount of time has passed, the slave samples the 1-Wire line that is driven by the master. This sampling point is in between the end of the minimum-width low pulse and the end of the time slot.

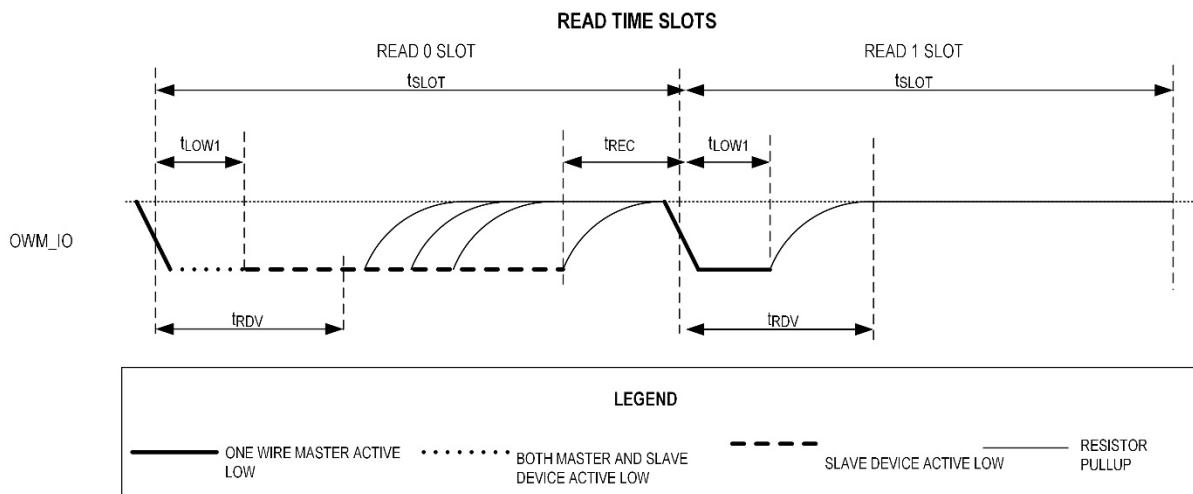
#### 16.3.1.2.3 OWM Read Time Slot

As with all 1-Wire transactions, the master initiates all bit read time slots. Like the bit write time slots, the bit read time slot begins with a falling edge. From the master's perspective, this time slot is transmitted identically to the "Write 1 Bit" time slot shown in *Figure 16-3*. The master begins by transmitting a falling edge, holds the line low for a minimum-width period, and then releases the line.

The difference here is that instead of the slave sampling the line, the slave begins transmitting either a 0 (by holding the line low) or a 1 (by leaving the line to float high) after the initial falling edge. The master then samples the line to read the bit value that is transmitted by the slave device.

As an example, *Figure 16-4* shows a sequence in which the slave device transmits data back to the 1-Wire bus master upon request. Note that to transmit a 1 bit, the slave device does not need to do anything. It simply leaves the line alone (to float high) and waits for the next time slot. To transmit a 0 bit, the slave device holds the line low until the end of the time slot.

*Figure 16-4: 1-Wire Read Time Slot*



#### 16.3.1.2.4 Standard Speed and Overdrive Speed

By default, all 1-Wire communications following reset begin at the lowest rate of speed (that is, standard speed). For 1-Wire devices that support it, it is possible for the OWM to increase the rate of communication from standard speed to overdrive speed by sending the appropriate command.

The protocols and time slots operate identically for standard and overdrive speeds. The difference comes in the widths of the time slots and pulses. The OWM automatically adjusts the timings based on the setting of the *OWM\_CFG.overdrive* field.

If a 1-Wire slave device receives a standard speed reset pulse, it resets and reverts to standard speed communication. If the device is already communicating in overdrive mode, and it receives a reset pulse at the overdrive speed, it resets but remains in overdrive mode.

#### 16.3.1.3 Network Layer

##### 16.3.1.3.1 ROM Commands

Following the initial 1-Wire reset pulse on the bus, all slave 1-Wire devices are active, which means 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 16-2*.

*Table 16-2: 1-Wire ROM Commands*

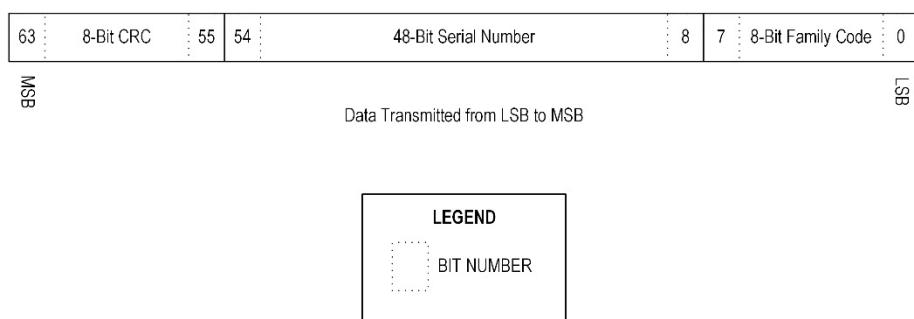
ROM Command	Hex Value
Read ROM	0x33
Match ROM	0x55
Search ROM	0xF0
Skip ROM	0xCC
Overdrive Skip ROM	0x3C
Overdrive Match ROM	0x69
Resume Communication	0xA5

The ROM command layer relies on the fact that all 1-Wire slave devices are assigned a globally unique, 64-bit ROM ID. This ROM ID value is factory programmed to ensure that no two 1-Wire slave devices have the same value.

##### 16.3.1.3.2 ROM ID

*Figure 16-5* is a visual representation of the 1-Wire ROM ID fields and shows the organization of the fields within the 64-bit ROM ID for a device.

*Figure 16-5: 1-Wire ROM ID Fields*



*Table 16-3* provides a detailed description of each of the ROM ID fields.

*Table 16-3: 1-Wire Slave Device ROM ID Field*

Field	Bit Number	Description
Family code	0-7	This 8-bit value is used to identify the type of a 1-Wire slave device.
Unique ID	8-55	This 48-bit value is factory-programmed to give each 1-Wire slave device (within a given family code group) a globally unique identifier.
CRC	56-63	This is the 8-bit, 1-Wire CRC as defined in the <i>Book of iButton Standards</i> . The CRC is generated using the polynomial $(x^8 + x^5 + x^4 + 1)$ .

*Note: For certain operations that consist only of writing from the OWM to the slave, it is technically possible for the master to communicate with more than one slave at a time on the same 1-Wire bus. For this to work, the exact same data must be transmitted to all slave devices, and any values read back from the slaves must either be identical as well or must be disregarded by the master device (because different slaves can attempt to transmit different values). The 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.

### 16.3.2 Read ROM Command

The Read ROM command allows the OWM to obtain the 8-byte ROM ID of any slave device connected to the 1-Wire bus. Each slave device on the bus responds to this command by transmitting all eight bytes of its ROM ID value starting with the least significant byte (Family Code) and ending with the most significant byte (CRC).

Because this command is addressed to all 1-Wire devices on the bus, if more than one slave is present on the bus there is a data collision as multiple slaves attempt to transmit their ROM IDs at once. This condition is detectable by the OWM because the CRC value does not match the ROM ID value received. In this case, the OWM should reset the 1-Wire bus and select a single slave device on the bus to continue either by using the Match ROM command (if the ROM ID values are already known) or the Search ROM command (if the master has not yet identified some or all devices on the bus).

After the Read ROM command is complete, all slave devices on the 1-Wire bus are selected or active, and communication proceeds to the Transport layer.

### 16.3.3 Skip ROM and Overdrive Skip ROM Commands

The Skip ROM command is used to activate all slave devices present on the 1-Wire bus regardless of their ROM ID. Normally, this command is used when only a single 1-Wire slave device is connected to the bus. After the Skip ROM command is complete, all slave devices on the 1-Wire bus are selected or active and communication proceeds to the Transport layer.

The Overdrive Skip ROM command operates in an identical manner except that running it also causes the receiving slave devices to shift communication speed from standard speed to overdrive speed. The Overdrive Skip ROM command byte itself (0x3C) is transmitted at standard speed. All subsequent communication is sent at overdrive speed.

### 16.3.4 Match ROM and Overdrive Match ROM Commands

The Match ROM command is used by the OWM to select one and only one slave 1-Wire device when the ROM ID of the device 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.

### 16.3.5 Search ROM Command

The Search ROM command allows the OWM to determine the ROM ID values of all 1-Wire slave devices connected to the bus using an iterative search process. Each execution of the Search ROM command reveals the ROM ID of one slave device on the bus.

The operation of the Search ROM command resembles a combination of the Read ROM and Match ROM commands. First, all slaves on the bus transmit the least significant bit (bit 0) of their ROM IDs. Next, all slaves on the bus transmit a complement of the same bit. By analyzing the two bits received, the master can determine 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.

### 16.3.6 Search ROM Accelerator Operation

To allow the Search ROM command to process more quickly, the OWM module provides a special accelerator mode for use with the Search ROM command. This mode is activated by setting `OWM_CTRL_STAT.sra_mode` to 1.

When this mode is active, ROM IDs being processed by the Search ROM command are broken into 4-bit nibbles where the current 64-bit ROM ID varies with each pass through the search algorithm. Each 4-bit processing step is initiated by writing the 4-bit value to `OWM_DATA.tx_rx`. This causes the generation of twelve 1-Wire time slots by the OWM as each bit in the 4-bit value (starting with the LSB) results in a read of two bits (all active slaves transmitting bit N of their ROM IDs, then all active slaves transmitting the complement of bit N of their ROM ID), and then a write of a single bit by the OWM.

After the 4-bit processing stage is complete, the return value loaded into `OWM_DATA.tx_rx` consists of 8 bits. The low nibble (bits 0 through 3) contains the four discrepancy flags: one for each ID bit processed. If the discrepancy bit is set to 1, it means that either two slaves with differing ID bits in that position both responded (the 2 bits read were both zero), or 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.

### 16.3.7 Resume Communication Command

If more than one 1-Wire slave device is on the bus, then the master must specify which one it wishes to communicate with each time a new 1-Wire command (starting with a reset pulse) begins. Using the commands discussed previously, this would normally involve sending the Match ROM command each time, which means the master must explicitly specify the full 64-bit ROM ID of the part it communicates with for each command.

The Resume Communication command provides a shortcut for this process by allowing the master to repeatedly select the same device for multiple commands without having to transmit the full ROM ID each time.

When the OWM selects a single device (using the Match ROM or Search ROM commands), an internal flag called the RC (for Resume Communication) flag is set in the slave device. (Only one device on the bus has this flag set at any one time; the Skip ROM command selects multiple devices, but the RC flag is not set by the Skip ROM command.)

When the master resets the 1-Wire bus, the RC flag remains set. At this point, it is possible for the master to send the Resume Communication command. This command does not have a ROM ID attached to it, but the device that has the RC flag set responds to this command by going to the active state while all other devices deactivate and drop off the 1-Wire bus.

Issuing any other ROM command clears the RC flag on all devices. So, for example, if a Match ROM command is issued for device A, its RC flag is set. The Resume Communication command can then be used repeatedly to send commands to device A. If a Match ROM command is then sent with the ROM ID of device B, the RC flag on device A will clear to 0, and the RC flag on device B is set.

## 16.4 1-Wire Operation

Once the OWM peripheral is correctly configured, then using the OWM peripheral to communicate with the 1-Wire network involves directing the OWM to generate the proper reset, read, and write operations to communicate with the 1-Wire slave devices used in a specific application.

The OWM manages the following 1-Wire protocol primitives directly in either Standard or Overdrive mode:

- 1-Wire bus reset (including detection of presence pulse from responding slave devices).
- Write single bit (a single write time slot).
- Write 8-bit byte, least significant bit first (eight write time slots).
- Read single bit (a single write-1 time slot).
- Read 8-bit byte, least significant bit first (eight write-1 time slots).
- Search ROM Acceleration Mode allowing the generation of four groups of three time slots (read, read, and write) from a single 4-bit register write to support the Search ROM command.

### 16.4.1 Resetting the OWM

The first step in any 1-Wire communication sequence is to reset the 1-Wire bus. To direct the OWM module to complete a 1-Wire reset, write `OWM_CTRL_STAT.start_ow_reset` to 1. This generates a reset pulse and checks for a replying presence pulse from any connected slave devices.

Once the reset time slot is complete, the `OWM_CTRL_STAT.start_ow_reset` field is automatically cleared to zero. Then, the interrupt flag `OWM_INTFL.ow_reset_done` is set to 1 by 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.

## 16.5 1-Wire Data Reads

### 16.5.1 Reading a Single Bit Value from the 1-Wire Bus

The procedure for reading a single bit is like the procedure for writing a single bit because the operation is completed by writing a 1 bit that the slave device either leaves unchanged (to transmit a 1 bit) or overrides by forcing the line low (to transmit a 0 bit).

To read a single bit value from the 1-Wire Bus, complete the following steps:

1. Set `OWM_CFG.single_bit_mode` to 1. This setting causes the OWM to transmit/receive a single bit of data at a time instead of the default 8 bits.
2. Write `OWM_DATA.tx_rx` to 1. Only bit 0 of this field is used in this instance; the other bits in the field are ignored. Writing to the `OWM_DATA` register initiates the read of the bit on the 1-Wire bus.
3. Once the single-bit transmission is complete, 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_ready` is set to 1, an OWM module interrupt occurs.
5. Read `OWM_DATA.tx_rx` (only bit 0 is used) to determine the value returned by the slave device. Note that if no slave devices are present or the slaves are not communicating with the master, bit 0 remains set to 1.

### 16.5.2 Reading an 8-Bit Value from the 1-Wire Bus

The procedure for reading an 8-bit byte is like the procedure for writing an 8-bit byte because the operation is completed by writing eight 1 bits that the slave device either leaves unchanged (to transmit 1 bits) or overrides by forcing the line low (to transmit 0 bits).

1. Set *OWM\_CFG.single\_bit\_mode* to 0. This setting causes the OWM to transmit/receive in the default 8-bit mode.
2. Write *OWM\_DATA.tx\_rx* to 0xFFh.
3. Once the 8-bit transmission completes, 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\_ready* is set to 1, an OWM module interrupt occurs.
5. Read *OWM\_DATA.tx\_rx* to determine the 8-bit value returned by the slave device. *Note that if no slave devices are present or the slave devices are not communicating with the master, the return value 0xFF is the same as the transmitted value.*

## 16.6 Registers

See *Table 2-4* for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in *Table 16-4*. Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn\_CTRL resolves to PERIPHERAL0\_CTRL and PERIPHERAL1\_CTRL for instances 0 and 1, respectively.

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

*Table 16-4: OWM Register Summary*

Offset	Register	Description
[0x0000]	<i>OWM_CFG</i>	OWM Configuration Register
[0x0004]	<i>OWM_CLK_DIV_1US</i>	OWM Clock Divisor Register
[0x0008]	<i>OWM_CTRL_STAT</i>	OWM Control/Status Register
[0x000C]	<i>OWM_DATA</i>	OWM Data Buffer Register
[0x0010]	<i>OWM_INTFL</i>	OWM Interrupt Flag Register
[0x0014]	<i>OWM_INTEN</i>	OWM Interrupt Enable Register

### 16.6.1 Register Details

*Table 16-5: OWM Configuration Register*

OWM Configuration Register		OWM_CFG			[0x0000]
Bits	Field	Access	Reset	Description	
31:8	-	R/W	0	<b>Reserved for Future Use</b> Do not modify this field.	
7	int_pullup_enable	R/W	0	<b>Internal Pullup Enable</b> Set this field to enable the internal pullup resistor. 0: Internal pullup disabled. 1: Internal pullup enabled.	

OWM Configuration Register			OWM_CFG		[0x0000]
Bits	Field	Access	Reset	Description	
6	overdrive	R/W	0	<b>Overdrive Enable</b> Set this field to 1 to enable overdrive mode for 1-Wire communications. Clearing this field sets 1-Wire communications to standard speed. 0: Overdrive mode disabled, standard speed mode. 1: Overdrive mode enabled.	
5	single_bit_mode	R/W	0	<b>Bit Mode Enable</b> When set to 1, only a single bit at a time is transmitted and received (LSB of <i>OWM_DATA</i> ) 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	<b>External Pullup Enable</b> 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	<b>External Pullup Mode</b> 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	<b>Bit-Bang Mode Enable</b> Enable bit-bang control of the I/O pin. If this bit is set to 1, <i>OWM_CTRL_STAT.bit_bang_oe</i> 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	<b>Presence Detect Force</b> 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 <i>OWM_CTRL_STAT.presence_detect</i> 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	<b>Long Line Mode Enable</b> Selects alternate timings for 1-Wire communication. The recommended setting depends on the length of the wire. For lines less than 40 meters, 0 should be used. Setting this bit to 0 leaves the write one release, the data sampling, and the time-slot recovery times at approximately 5µs, 15µs, and 7µs, respectively. Setting this bit to 1 enables long line mode timings during standard mode communications. This mode moves the write one release, the data sampling, and the time-slot recovery times out to approximately 8µs, 22µs, and 14µs, respectively. 0: Standard operation for lines less than 40 meters. 1: Long Line mode enabled.	

Table 16-6: OWM Clock Divisor Register

OWM Clock Divisor		OWM_CLK_DIV_1US			[0x0004]
Bits	Field	Access	Reset	Description	
31:8	-	R	0	<b>Reserved for Future Use</b> Do not modify this field.	
7:0	divisor	R/W	0	<b>OWM Clock Divisor</b> Divisor for the OWM peripheral clock. The target is to achieve a 1MHz clock. See the <a href="#">Clock Configuration</a> section for details. 0x00: OWM clock disabled. $0x01: f_{owmclk} = \frac{f_{PCLK}}{1}$ $0x02: f_{owmclk} = \frac{f_{PCLK}}{2}$ ... $0xFF: f_{owmclk} = \frac{f_{PCLK}}{255}$	

Table 16-7: OWM Control Status Register

OWM Control Status		OWM_CTRL_STAT			[0x0008]
Bits	Field	Access	Reset	Description	
31:8	-	RO	0	<b>Reserved for Future Use</b> Do not modify this field.	
7	presence_detect	RO	0	<b>Presence Detect Flag</b> 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	<b>Reserved for Future Use</b> Do not modify this field.	
4	od_spec_mode	RO	0	<b>Overdrive Spec Mode</b> Returns the version of the overdrive spec.	
3	ow_input	RO	-	<b>OWM_IN State</b> 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	<b>OWM Bit-Bang Output</b> When bit-bang mode is enabled ( <a href="#">OWM_CFG.bit_bang_en = 1</a> ), 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	<b>Search ROM Accelerator Enable</b> 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.	

OWM Control Status			OWM_CTRL_STAT		[0x0008]
Bits	Field	Access	Reset	Description	
0	start_ow_reset	R/W	0	<b>Start 1-Wire Reset Pulse</b> Write 1 to start a 1-Wire reset sequence. Automatically cleared by the OWM hardware when the reset sequence is complete. 0: 1-Wire reset sequence complete or inactive. 1: Start a 1-Wire reset sequence.	

Table 16-8: OWM Data Buffer Register

OWM Data			OWM_DATA		[0x000C]
Bits	Field	Access	Reset	Description	
31:8	-	R/W	0	<b>Reserved for Future Use</b> Do not modify this field.	
7:0	tx_rx	R/W	0	<b>OWM Data Field</b> Writing to this field sets the transmit data and initiates a 1-Wire data transmit cycle. Reading from this field returns the data received by the master during the last 1-Wire data transmit cycle.	

Table 16-9: OWM Interrupt Flag Register

OWM Interrupt Flag			OWM_INTFL		[0x0010]
Bits	Field	Access	Reset	Description	
31:5	-	R/W	0	<b>Reserved for Future Use</b> Do not modify this field.	
4	line_low	R/W1C	0	<b>Line Low Flag</b> 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	<b>Line Short Flag</b> The OWM hardware detected a short on the OWM_IO pin. Write 1 to clear this flag.	
2	rx_data_ready	R/W1C	0	<b>RX Data Ready</b> Data received from the 1-Wire bus and is available in the <i>OWM_DATA.tx_rx</i> field. Write 1 to clear this flag. 0: RX data not available. 1: Data received and is available in the <i>OWM_DATA.tx_rx</i> field.	
1	tx_data_empty	R/W1C	0	<b>TX Empty</b> The OWM hardware automatically sets this interrupt flag when the data transmit is complete. Write 1 to clear this flag. 0: Either no data was sent or the data in the <i>OWM_DATA.tx_rx</i> field has not completed transmission. 1: Data in the <i>OWM_DATA.tx_rx</i> field was transmitted.	
0	ow_reset_done	R/W1C	0	<b>Reset Complete</b> This flag is set when a 1-Wire reset sequence completes. To start a 1-Wire reset sequence, see <i>OWM_CTRL_STAT.start_ow_reset</i> . Write 1 to clear this flag. 0: 1-Wire reset sequence not complete or bus idle. 1: 1-Wire reset sequence complete.	

*Table 16-10: OWM Interrupt Enable Register*

OWM Interrupt Enable		OWM_INTEN			[0x0014]
Bits	Field	Access	Reset	Description	
31:5	-	RO	0	<b>Reserved for Future Use</b> Do not modify this field.	
4	line_low	R/W	0	<b>Line Low Interrupt Enable</b> I/O pin low detected interrupt enable. 0: Interrupt disabled. 1: Interrupt enabled.	
3	line_short	R/W	0	<b>Line Short Interrupt Enable</b> I/O pin short detected interrupt enable. 0: Interrupt disabled. 1: Interrupt enabled.	
2	rx_data_ready	R/W	0	<b>Receive Data Ready Interrupt Enable</b> RX data ready interrupt enable. 0: Interrupt disabled. 1: Interrupt enabled.	
1	tx_data_empty	R/W	0	<b>Transmit Data Empty Interrupt Enable</b> TX data empty interrupt enable. 0: Interrupt disabled. 1: Interrupt enabled.	
0	ow_reset_done	R/W	0	<b>1-Wire Reset Sequence Complete Interrupt Enable</b> 1-Wire reset sequence completed. 0: Interrupt disabled. 1: Interrupt enabled.	

## 17. Real-Time Clock (RTC)

### 17.1 Overview

The real-time clock (RTC) is a 32-bit binary timer that keeps the time of day up to 136 years. It provides time-of-day and sub-second alarm functionality in the form of system interrupts.

The RTC operates on an external 32.768kHz time base. It can be generated from the internal crystal oscillator driving an external 32.768kHz crystal between the 32KIN and 32KOUT pins (ERTCO), 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 register, *RTC\_SEC*, is incremented every time there is a rollover of the *RTC\_SSEC.ssec* sub-seconds field.

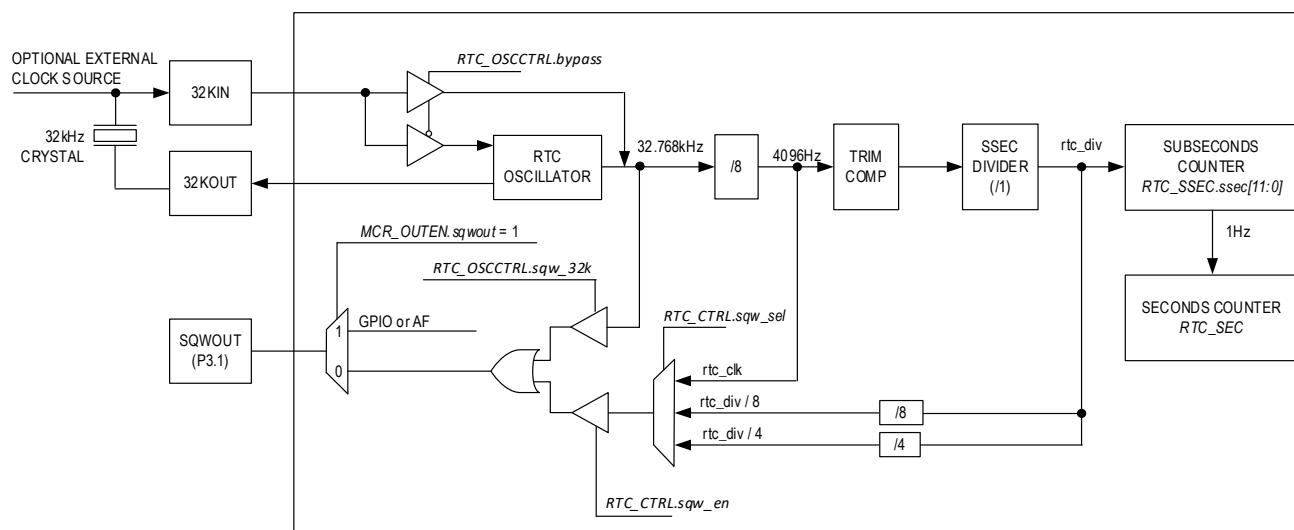
Two alarm functions are provided:

1. A programmable time-of-day alarm provides a single event, alarm timer using the *RTC\_TODA* alarm register, *RTC\_CTRL*.*tod\_alarm\_ie* field.
2. A programmable sub-second provides a recurring alarm using the RTC sub-second alarm register, *RTC\_SSECA*, and is *RTC\_CTRL*.*ssec\_alarm* field

The RTC is powered in the AoD or if applicable, while there is a valid voltage on the V<sub>COREA</sub> device pin.

Disabling the RTC, *RTC\_CTRL.en* cleared to 0, stops incrementing the *RTC\_SSEC*, *RTC\_SEC*, and the internal RTC sub-second counter, 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* set to 1, the *RTC\_TRIM.vrtc\_tmr* field is also incremented every 32 seconds.

*Figure 17-1: MAX78000 RTC Block Diagram (12-bit Sub-Second Counter)*



## 17.2 Instances

One instance of the RTC peripheral is provided. The RTC counter and alarm register details and description are shown in *Table 17-1*.

*Table 17-1: RTC Seconds, Sub-Seconds, Time-of-Day Alarm and Sub-Seconds Alarm Register Details*

Register	Width (bits)	Counter Increment	Minimum	Maximum	Description
<i>RTC_SEC</i>	32	1 second	1 second	136 years	Seconds Counter Register
<i>RTC_SSEC</i>	12	$244 \mu\text{s} (\frac{1}{4096\text{Hz}})$	244 $\mu\text{s}$	1 second	Sub-Seconds Counter Register
<i>RTC_TODA</i>	20	1 second	1 second	12 days	Time-of-Day Alarm Register
<i>RTC_SSECA</i>	32	$244 \mu\text{s} (\frac{1}{4096\text{Hz}})$	244 $\mu\text{s}$	12 days	Sub-Second Alarm Register

## 17.3 Register Access Control

Access protection mechanisms prevent the software from accessing critical registers and fields 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 17-2: RTC Register Access*

Register	Field	Read Access	Write Access	<i>RTC_CTRL.busy = 1</i> during writes	Description
<i>RTC_SEC</i>	All	<i>RTC_CTRL.busy = 0</i> <i>RTC_CTRL.rdy = 1<sup>†</sup></i>	<i>RTC_CTRL.busy = 0</i> <i>RTC_CTRL.rdy = 1<sup>†</sup></i>	Y	Seconds Counter
<i>RTC_SSEC</i>	<i>ssec</i>	<i>RTC_CTRL.busy = 0</i> <i>RTC_CTRL.rdy = 1<sup>†</sup></i>	<i>RTC_CTRL.busy = 0</i> <i>RTC_CTRL.rdy = 1<sup>†</sup></i>	Y	Sub-Seconds Counter
<i>RTC_TODA</i>	All	Always	<i>RTC_CTRL.busy = 0</i> <i>RTC_CTRL.tod_alarm_ie = 0</i> <i>RTC_CTRL.en = 0</i>	Y	Time-of-Day Alarm
<i>RTC_SSECA</i>	All	Always	<i>RTC_CTRL.busy = 0</i> <i>RTC_CTRL.ssec_alarm_ie = 0</i> <i>RTC_CTRL.en = 0</i>	Y	Sub-Second Alarm
<i>RTC_TRIM</i>	All	Always	<i>RTC_CTRL.busy = 0</i> <i>RTC_CTRL.wr_en = 1</i>	Y	Trim
<i>RTC_OSCCTRL</i>	All	Always	<i>RTC_CTRL.wr_en = 1</i>	N	Oscillator Control
<i>RTC_CTRL</i>	<i>en</i>	Always	<i>RTC_CTRL.busy = 0</i> <i>RTC_CTRL.wr_en = 1</i>	Y	RTC Enable field
	All other bits	Always	++	Y	

<sup>†</sup> See the *RTC\_SEC* and *RTC\_SSEC* Read Access Control section for details.

<sup>++</sup> See the *RTC\_CTRL.busy* field description for limitations on specific bits.

### 17.3.1 *RTC\_SEC* and *RTC\_SSEC* Read Access Control

Software reads of the *RTC\_SEC* and *RTC\_SSEC* registers 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*). To avoid this, the hardware sets the *RTC\_CTRL.rdy* field to 1 for 120  $\mu\text{s}$  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 the *RTC\_SEC* and *RTC\_SSEC* registers.

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* register and/or the *RTC\_SSEC* register while the *RTC\_CTRL.rdy* field is 1 to ensure valid data. This avoids 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.

### **17.3.2 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.

## **17.4 RTC Alarm Functions**

The RTC provides time-of-day and sub-second interval alarm functions. The time-of-day alarm is implemented by matching the count values in the counter register with the value stored in the alarm register. The sub-second interval alarm provides an auto-reload timer driven by the trimmed RTC clock source.

### **17.4.1 Time-of-Day Alarm**

Program the RTC time-of-day alarm register (*RTC\_TODA*) to configure the time-of-day-alarm. The alarm triggers when the value stored in *RTC\_TODA.tod\_alarm* matches the lower 20 bits of the *RTC\_SEC* seconds count register. This allows programming the time-of-day-alarm to any future value between 1 second and 12 days relative to the current time with a resolution of 1 second. Disable the time-of-day alarm before changing the *RTC\_TODA.tod* field.

When the alarm occurs, a single event sets the time-of-day alarm interrupt flag (*RTC\_CTRL.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.

### **17.4.2 Sub-Second Alarm**

The *RTC\_SSECA* and *RTC\_CTRL.ssec\_alarm\_ie* field control the sub-second alarm. Writing *RTC\_SSECA* sets the starting value for the sub-second alarm counter. Writing the sub-second alarm enable (*RTC\_CTRL.ssec\_alarm\_ie*) bit to 1 enables the sub-second alarm. Once enabled, an internal alarm counter begins incrementing from the *RTC\_SSECA* value. When the counter rolls over from 0xFFFF FFFF to 0x0000 0000, 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.rssa*.

Disable the sub-second interval alarm, *RTC\_CTRL.ssec\_alarm\_ie*, prior to changing the interval alarm value, *RTC\_SSECA*.

The delay (uncertainty) associated with enabling the sub-second alarm is up to one period of the sub-second clock. This uncertainty is propagated to the first interval alarm. Thereafter, if the interval alarm remains enabled, the alarm triggers after each sub-second interval as defined without the first alarm uncertainty because the sub-second alarm is an auto-

reload timer. Enabling the sub-second alarm with the sub-second alarm register set to 0 (*RTC\_SSECA* = 0) results in the maximum sub-second alarm interval.

### 17.4.3 RTC Interrupt and Wakeup Configuration

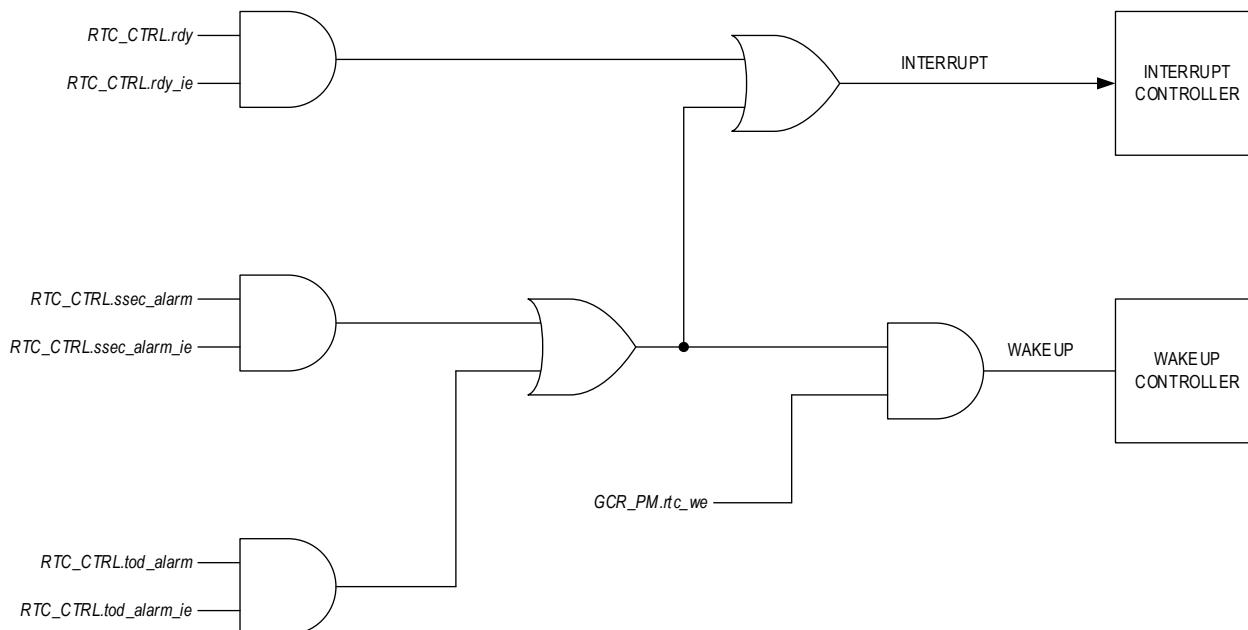
The following are a list of conditions that, when enabled, generate an RTC interrupt.

1. Time-of-day alarm
2. Sub-second alarm
3. *RTC\_CTRL.rdy* field asserted high, signaling read access permitted

The RTC can be configured so the time-of-day and sub-second alarms are a wakeup source for exiting the following low-power modes:

1. *SLEEP*
2. *DEEPSLEEP*
3. *UPM*
4. *BACKUP*

*Figure 17-2: RTC Interrupt/Wakeup Diagram Wakeup Function*



Use the following procedure to enable the RTC as a wakeup source:

1. Configure the RTC interrupt enable bits, so one or more interrupt conditions generate an RTC interrupt.
2. Create a 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 wakeup by the RTC.
4. Enter the desired low-power mode. See *Operating Modes* for details.

#### 17.4.4 Square Wave Output

The RTC can output a 50% duty cycle square wave signal derived from the 32kHz oscillator on a selected device pin. See [Table 17-3](#) for the device pins, frequency options, and control fields specific to this device. Frequencies noted as compensated are used during the RTC frequency calibration procedure because they incorporate the frequency adjustments provided by the digital trim function.

*Table 17-3: MAX78000 RTC Square Wave Output Configuration*

Function	Option	Control Field
Output Pin	P3.1: SQWOUT	<a href="#"><i>MCR_OUTEN.sqwout_en = 1</i></a>
Frequency Selection	1Hz (Compensated)	<a href="#"><i>RTC_CTRL.sqw_sel = 0</i></a>
	512Hz (Compensated)	<a href="#"><i>RTC_CTRL.sqw_sel = 1</i></a>
	4kHz	<a href="#"><i>RTC_CTRL.sqw_sel = 2</i></a>
	32kHz	<a href="#"><i>RTC_OSCCTRL.32k_out = 1</i></a>
Enable Frequency Output	1Hz (Compensated)	<a href="#"><i>RTC_CTRL.sqw_en = 1</i></a> <a href="#"><i>RTC_OSCCTRL.32k_out = 0</i></a>
	512Hz (Compensated)	<a href="#"><i>RTC_CTRL.sqw_en = 1</i></a> <a href="#"><i>RTC_OSCCTRL.32k_out = 0</i></a>
	4kHz	<a href="#"><i>RTC_CTRL.sqw_en = 1</i></a> <a href="#"><i>RTC_OSCCTRL.32k_out = 0</i></a>
	32kHz	<a href="#"><i>RTC_OSCCTRL.32k_out = 1</i></a>

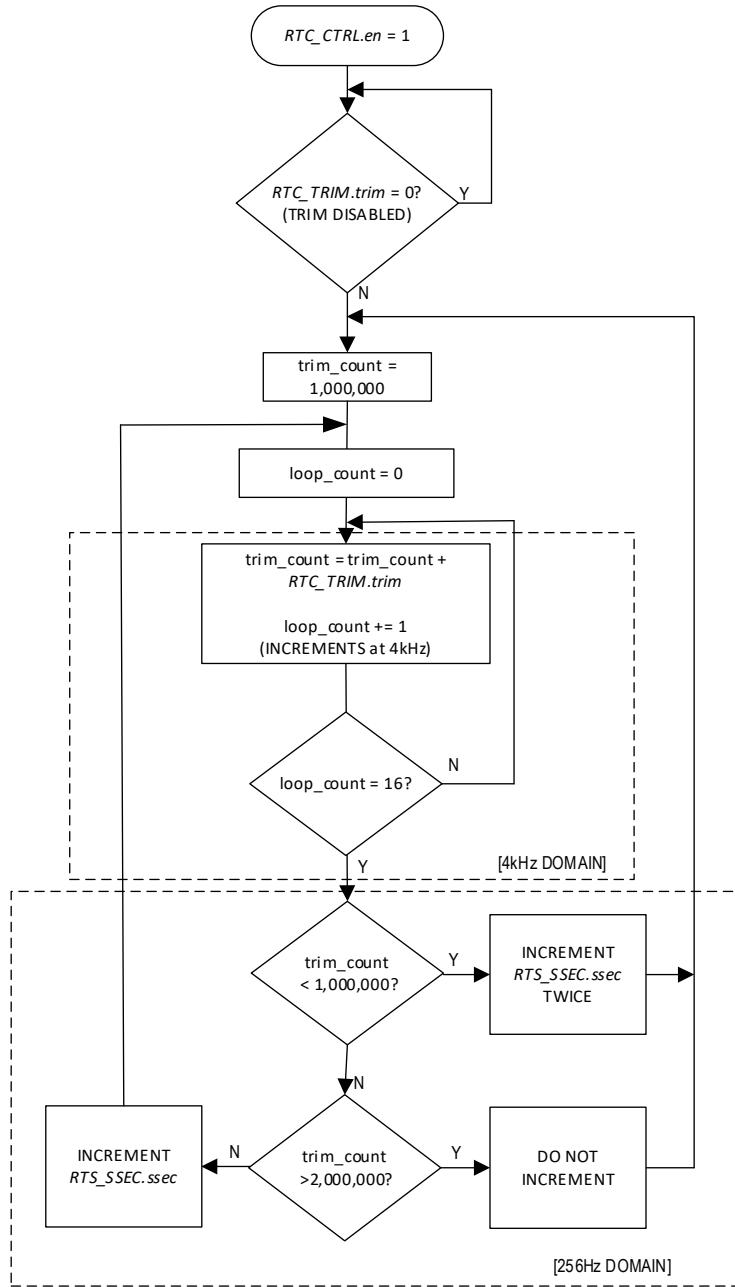
Use the following software procedure to generate and output the square wave:

1. Select the desired frequency to output:
  - 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.32k\\_out\*](#) to 1 for the 32kHz frequency output
2. Enable the system level output pin by setting the Output Pin as shown in [Table 17-3](#).
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.

#### 17.5 RTC Calibration

A digital trim facility provides the ability to compensate for RTC inaccuracies of up to  $\pm 127\text{ppm}$  when compared against an external reference clock. The trimming function utilizes an independent dedicated timer that increments the sub-second register based on a user-supplied, two's compliment value in the [\*RTC\\_TRIM\*](#) register as shown in [Figure 17-3](#).

Figure 17-3: 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 dethroughtion 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 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.

## 17.6 Registers

See [Table 2-4](#) for the base address of this peripheral/module. See [Table 2-1](#) 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 17-4: RTC Register Summary*

Offset	Register	Description
[0x0000]	<i>RTC_SEC</i>	RTC Seconds Counter Register
[0x0004]	<i>RTC_SSEC</i>	RTC Sub-Second Counter Register
[0x0008]	<i>RTC_TODA</i>	RTC Time-of-Day Alarm Register
[0x000C]	<i>RTC_SSECA</i>	RTC Sub-Second Alarm Register
[0x0010]	<i>RTC_CTRL</i>	RTC Control Register
[0x0014]	<i>RTC_TRIM</i>	RTC 32KHz Oscillator Digital Trim Register
[0x0018]	<i>RTC_OSCCTRL</i>	RTC 32KHz Oscillator Control Register

### 17.6.1 Register Details

*Table 17-5: RTC Seconds Counter Register*

RTC Seconds Counter			RTC_SEC		[0x0000]
Bits	Field	Access	Reset	Description	
31:0	sec	R/W	0	<b>Seconds Counter</b> This register is a binary count of seconds.	

Table 17-6: RTC Sub-Second Counter Register (12-bit)

RTC Sub-Seconds Counter			RTC_SSEC		[0x0004]
Bits	Field	Access	Reset	Description	
31:12	-	RO	0	Reserved	
11:0	ssec	R/W	0	Sub-Seconds Counter (12-bit) <i>RTC_SEC</i> increments when this field rolls from 0xFFFF to 0x0000	

Table 17-7: RTC Time-of-Day Alarm Register

RTC Time-of-Day Alarm			RTC_TODA		[0x0008]
Bits	Field	Access	Reset	Description	
31:20	-	RO	0	Reserved	
19:0	tod_alarm	R/W	0	Time-of-Day Alarm Sets the time-of-day alarm from 1 second up to 12-days. When this field matches <i>RTC_SEC</i> [19:0], an RTC system interrupt is generated.	

Table 17-8: RTC Sub-Second Alarm Register

RTC Sub-Second Alarm			RTC_SSECA		[0x000C]
Bits	Field	Access	Reset	Description	
31:0	ssec_alarm	R/W	0	Sub-second Alarm (4KHz) Sets the starting and reload value of the internal sub-second alarm counter. The internal counter increments and generates an alarm when the internal counter rolls from 0xFFFF FFFF to 0x0000 0000.	

Table 17-9: RTC Control Register

RTC Control Register			RTC_CTRL		[0x0010]
Bits	Field	Access	Reset	Description	
31:16	-	RO	0	Reserved	
15	wr_en	R/W	0*	Write Enable This field controls access to the <i>RTC_TRIM</i> register, the RTC enable ( <i>RTC_CTRL.en</i> ) fields.  1: Writes to the <i>RTC_TRIM</i> register and the <i>RTC_CTRL.en</i> field are allowed. 0: Writes to the <i>RTC_TRIM</i> register and the <i>RTC_CTRL.en</i> field are ignored.  <i>*Note: Reset on System Reset, Soft Reset, and GCR_RST0.rtc assertion.</i>	
14	rd_en	R/W	0	Asynchronous Counter Read Enable Set this field to 1 to allow direct read access of the <i>RTC_SEC</i> and <i>RTC_SSEC</i> registers without waiting for <i>RTC_CTRL.rdy</i> . Multiple consecutive reads of <i>RTC_SEC</i> and <i>RTC_SSEC</i> must be executed until two consecutive reads are identical to ensure data accuracy.  0: <i>RTC_SEC</i> and <i>RTC_SSEC</i> registers are synchronized and should only be accessed while <i>RTC_CTRL.rdy</i> = 1. 1: <i>RTC_SEC</i> and <i>RTC_SSEC</i> registers are asynchronous and will require the software interaction to ensure data accuracy.	
13:11	-	RO	0	Reserved	
10:9	sqw_sel	R/W	0*	Frequency Output Select Selects the RTC-derived frequency to output on the square wave output pin. See <a href="#">Table 17-3</a> for configuration details.  0: 1Hz (Compensated) 1: 512Hz (Compensated) 2: 4kHz  <i>*Note: Reset on POR only.</i>	

RTC Control Register				RTC_CTRL	[0x0010]
Bits	Field	Access	Reset	Description	
8	sqw_en	R/W	0*	<b>Square Wave Output Enable</b> Enables the square wave output. See <a href="#">Table 17-3</a> for configuration details. 0: Disabled. 1: Enabled. <i>*Note: Reset on POR only.</i>	
7	ssec_alarm	R/W	0*	<b>Sub-second Alarm Interrupt Flag</b> 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. <i>*Note: Reset on POR only.</i>	
6	tod_alarm	R/W	0*	<b>Time-of-Day Alarm Interrupt Flag</b> 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. <i>*Note: Reset on POR only.</i>	
5	rdy_ie	R/W	0*	<b>RTC Ready Interrupt Enable</b> 0: Disabled. 1: Enabled. <i>*Note: Reset on System Reset, Soft Reset, and GCR_RST0.rtc assertion.</i>	
4	rdy	R/W0	0*	<b>RTC Ready</b> This bit is set to 1 for 120µs by the hardware once a hardware update of the <a href="#">RTC_SEC</a> and <a href="#">RTC_SSEC</a> registers has occurred. The software should read <a href="#">RTC_SEC</a> and <a href="#">RTC_SSEC</a> while this hardware bit is set to 1. The software can clear this bit at any time. An RTC interrupt is generated if <a href="#">RTC_CTRL.rdy_ie</a> = 1. 0: Software reads of <a href="#">RTC_SEC</a> and <a href="#">RTC_SSEC</a> are invalid. 1: Software reads of <a href="#">RTC_SEC</a> and <a href="#">RTC_SSEC</a> are valid. <i>*Note: Reset on System Reset, Soft Reset, and GCR_RST0.rtc assertion.</i>	

RTC Control Register			RTC_CTRL		[0x0010]
Bits	Field	Access	Reset	Description	
3	busy	RO	0*	<p><b>RTC Busy Flag</b>            This field is set to 1 by the hardware while a register update is in progress when the software writes to the following registers:</p> <ul style="list-style-type: none"> <li>• <a href="#">RTC_SEC</a></li> <li>• <a href="#">RTC_SSEC</a></li> <li>• <a href="#">RTC_TRIM</a></li> </ul> <p>The following fields cannot be written when this field is set to 1:</p> <ul style="list-style-type: none"> <li>• <a href="#">RTC_CTRL.en</a></li> <li>• <a href="#">RTC_CTRL.tod_alarm_ie</a></li> <li>• <a href="#">RTC_CTRL.ssec_alarm_ie</a></li> <li>• <a href="#">RTC_CTRL.rdy_ie</a></li> <li>• <a href="#">RTC_CTRL.tod_alarm</a></li> <li>• <a href="#">RTC_CTRL.ssec_alarm</a></li> <li>• <a href="#">RTC_CTRL.sqw_en</a></li> <li>• <a href="#">RTC_CTRL.ft</a></li> <li>• <a href="#">RTC_CTRL.rd_en</a></li> </ul> <p>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 <a href="#">RTC_SEC</a>, <a href="#">RTC_SSEC</a>, or <a href="#">RTC_TRIM</a> register, prior to making any other RTC register modifications.</p> <p>0: RTC not busy            1: RTC busy</p> <p>*Note: Reset on POR only.</p>	
2	ssec_alarm_ie	R/W	0*	<p><b>Sub-Second Alarm Interrupt Enable</b>            Check the <a href="#">RTC_CTRL.busy</a> flag after writing to this field to determine when the RTC synchronization is complete.</p> <p>0: Disable.            1: Enable.</p> <p>*Note: Reset on POR only.</p>	
1	tod_alarm_ie	R/W	0*	<p><b>Time-of-Day Alarm Interrupt Enable</b>            Check the <a href="#">RTC_CTRL.busy</a> flag after writing to this field to determine when the RTC synchronization is complete.</p> <p>0: Disable.            1: Enable.</p> <p>*Note: Reset on POR only.</p>	
0	en	R/W	0*	<p><b>Real-Time Clock Enable</b>            The RTC write enable (<a href="#">RTC_CTRL.wr_en</a>) bit must be set and RTC Busy (<a href="#">RTC_CTRL.busy</a>) must read 0 before writing to this field. After writing to this bit, check the <a href="#">RTC_CTRL.busy</a> flag for 0 to determine when the RTC synchronization is complete.</p> <p>0: Disabled.            1: Enabled.</p> <p>*Note: Reset on POR only.</p>	

Table 17-10: RTC 32KHz Oscillator Digital Trim Register

RTC 32KHz Oscillator Digital Trim			RTC_TRIM		[0x0014]
Bits	Field	Access	Reset	Description	
31:8	vrtc_tmr	R/W	0*	<p><b>VRTC Time Counter</b>            The hardware increments this field every 32 seconds while the RTC is enabled.</p> <p>*Note: Reset on POR only.</p>	

RTC 32KHz Oscillator Digital Trim			RTC_TRIM		[0x0014]
Bits	Field	Access	Reset	Description	
7:0	trim	R/W	0*	<b>RTC Trim</b> This field specifies the 2s complement value of the trim resolution. Each increment or decrement of the field adds or subtracts 1ppm at each 4kHz clock value with a maximum correction of ± 127ppm. <i>*Note: Reset on POR only.</i>	

Table 17-11: RTC 32KHz Oscillator Control Register

RTC Oscillator Control			RTC_OSCCTRL		[0x0018]
Bits	Field	Access	Reset	Description	
31:6	-	R/W	0	<b>Reserved</b>	
5	sqw_32k	R/W	0	<b>RTC Square Wave Output</b> 0: Disabled. 1: Enables the 32kHz oscillator output or the external clock source is output on square wave output pin. See <a href="#">Table 17-3</a> for configuration details. <i>*Note: Reset on POR only.</i>	
4	bypass	R/W	0	<b>RTC Crystal Bypass</b> This field disables the RTC oscillator and allows an external clock source to drive the 32KIN pin. 0: Disable bypass. RTC time base is external 32kHz crystal. 1: Enable bypass. RTC time base is external square wave driven on 32KIN. <i>*Note: Reset on POR only.</i>	
3:0	-	DNM	0b1001	<b>Reserved Do Not Modify</b>	

## 18. 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, mapped as alternate functions.
- 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 18-1](#), 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 terminal value then repeats.
- Counter: the timer counts input edges received on the timer input pin.
- PWM / PWM Differential.
- 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.

## 18.1 Instances

Instances of the peripheral are listed in *Table 18-1*. 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 18-1: MAX78000 TMR/LPTMR Instances*

Instance	Register Access Name	Cascade 32-Bit Mode	16-Bit Mode	Operating Modes	CLK0	CLK1	CLK2	CLK3
TMR0	TMR0	Yes	Dual	<i>ACTIVE</i> <i>SLEEP</i> <i>LPM</i>	PCLK	ISO	IBRO	ERTCO
TMR1	TMR1							
TMR2	TMR2							
TMR3	TMR3							
LPTMRO	TMR4	No	Single	<i>ACTIVE</i> <i>SLEEP</i> <i>LPM</i>	IBRO	ERTCO	INRO	LPTMRO_CLK P2.6 (AF1)
				UPM	N/A	N/A	ERTCO	INRO
LPTMR1	TMR5	No	Single	<i>ACTIVE</i> <i>SLEEP</i> <i>LPM</i>	IBRO	<i>IBRO</i> 8	INRO	LPTMR1_CLK P2.7 (AF1)
				UPM	N/A	N/A	ERTCO	INRO

*Table 18-2: MAX78000 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	-	-	-	-
LPTMRO	LPTMROB_IOA	LPCMP0 Interrupt	LPCMP1 Interrupt	-
LPTMR1	LPTMR1B_IOA	LPCMP0 Interrupt	LPCMP1 Interrupt	-

## 18.2 Basic Timer Operation

The timer modes operate by incrementing the *TMRn\_CNT.count* register, driven by either the timer clock, an external stimulus on the timer pin, or a combination of both. The *TMRn\_CNT.count* register is always readable, even while the timer is enabled and counting.

Each timer mode has a user-configurable timer period, which terminates on the timer clock cycle following the end of 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.count* 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.count* to 0x0000 0001 at the end of a timer period, but *TMRn\_CNT.count* 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.count* is not initialized to 0x0000 0001 during the timer configuration step.

## 18.3 32-Bit Single / 32-Bit Cascade / Dual 16-Bit

Most instances contain two 16-bit timers, which may support combinations of single or cascaded 32-bit modes, and single or dual 16-bit modes as shown in [Table 18-1](#). In most cases, the two 16-bit timers have the same functionality.

The terminology TimerA and TimerB are used to differentiate the organization of the 32-bit registers shown in [Table 18-3](#). 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 are used to control the 32-bit timer functionality. In single 16-bit timer mode, the TimerA fields are used to control the single 16-bit timer and the TimerB fields are ignored. In dual 16-bit timer modes, both TimerA and TimerB fields are used to 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 18-3: TimerA/TimerB 32-Bit Field Allocations*

Register	Cascade 32-Bit Mode	Dual 16-Bit Mode		Single 16-Bit Mode
Timer Counter	TimerA Count = <i>TMRn_CNT.count[31:0]</i>	TimerA Compare = <i>TMRn_CNT.compare[15:0]</i>	TimerB Count = <i>TMRn_CNT.count[31:16]</i>	TimerA Compare = <i>TMRn_CNT.compare[15:0]</i>
Timer Compare	TimerA Compare = <i>TMRn_CMP.compare[31:0]</i>	TimerA Compare = <i>TMRn_CMP.compare[15:0]</i>	TimerB Compare = <i>TMRn_CMP.compare[31:16]</i>	TimerA Compare = <i>TMRn_CMP.compare[15:0]</i>
Timer PWM	TimerA Count = <i>TMRn_PWM.pwm[31:0]</i>	TimerA Count = <i>TMRn_PWM.pwm[15:0]</i>	TimerB Count = <i>TMRn_PWM.pwm[31:16]</i>	TimerA Count = <i>TMRn_PWM.pwm[15:0]</i>

## 18.4 Timer Clock Sources

Clocking of timer functions is driven by the timer clock frequency,  $f_{CNT\_CLK}$ , which is a function of the selected clock source shown in [Table 18-1](#). Most modes support multiple clock sources and prescaler values, which can be chosen independently for TimerA and TimerB when the peripheral is operating in dual 16-bit mode. The prescaler can be set from 1 to 4096 using the *TMRn\_CNT.pres* field.

*Equation 18-1: Timer Peripheral Clock Equation*

$$f_{CNT\_CLK} = \frac{f_{CLK\_SOURCE}}{\text{prescaler}}$$

The software configures and controls the timer's 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\_CNT.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.

## 18.5 Timer Pin Functionality

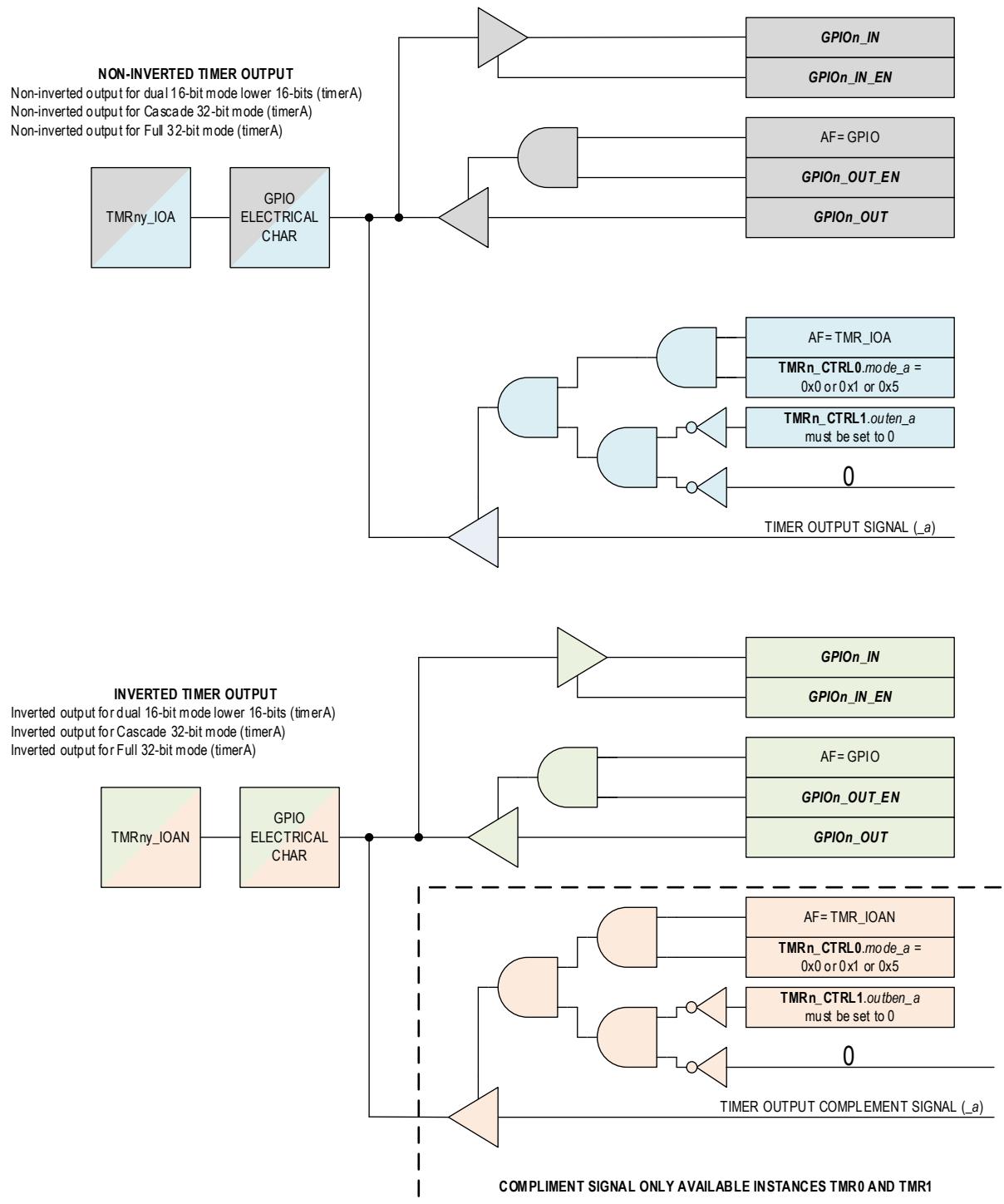
Each timer instance may have an input signal and/or output signal depending on the operating mode. Not all instances of the peripheral are available in all packages. The number of input and output signals per peripheral instance may vary as well. Refer to the data sheet for I/O signal configurations and alternate functions for each Timer instance.

The physical pin location of the timer input and/or output signals may vary between packages. The timer functionality, however, 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, such as pullup/pulldown strength, drive strength, etc., as the GPIO mode settings for that pin. The pin characteristics must be configured before enabling the timer. When configured as an output, the corresponding bit in the *GPIO\_OUT* register should be configured to match the inactive state of the timer pin for that mode. Consult the GPIO section for details on how to configure the electrical characteristics for the pin.

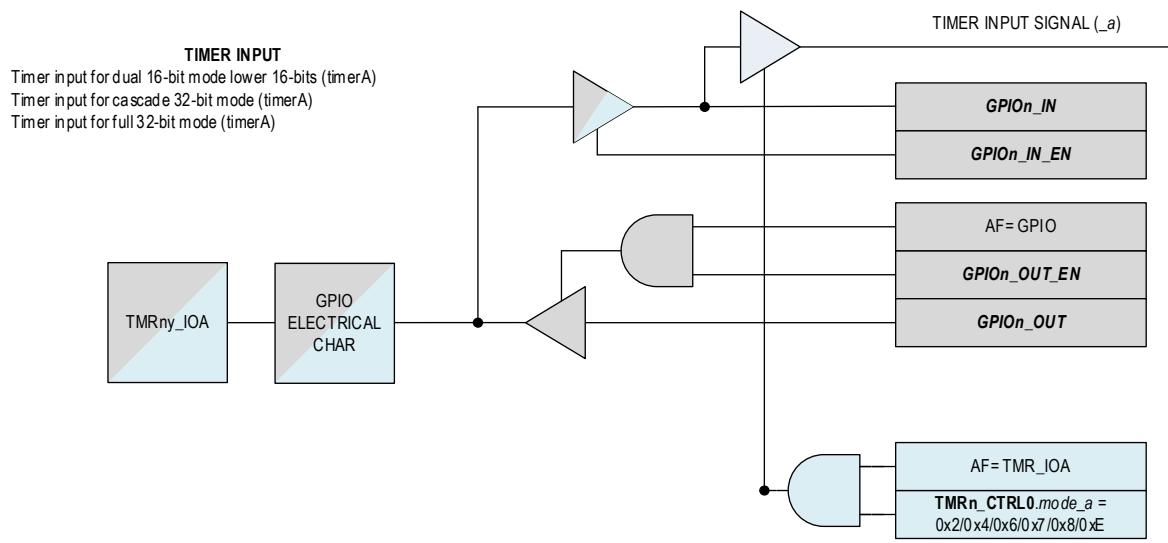
The TimerA output controls for modes 0, 1, 3, and 5 output signals are shown in *Figure 18-1*. The TimerA input controls for modes 2, 4, 6, 7, 8, and 14 input signals are shown in *Figure 18-2*.

Figure 18-1: MAX78000 TimerA Output Functionality, Modes 0/1/3/5



Preliminary Draft 05/21/2021

Figure 18-2: MAX78000 TimerA Input Functionality, Modes 2/4/6/7/8/14



## 18.6 Wakeup Events

In low-power modes, the system clock may be turned off to conserve power. LPTMR instances can continue to run from the clock sources shown in [Table 18-1](#). In this case, a wakeup 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 wakeup events for the timer.
6. If desired, enable the timer interrupt and provide a [`TMRn\_IRQn`](#) for the timer.
7. 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 is the result of the wakeup event.

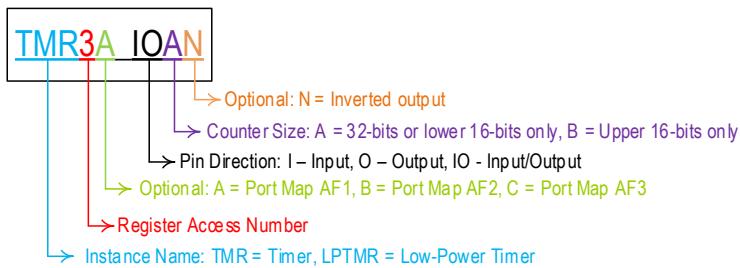
Table 18-4: MAX78000 Wakeup Events

Condition	Peripheral Wakeup Flag <a href="#"><code>TMRn_INTFL</code></a>	Peripheral Wakeup Enable	Low-Power Peripheral Wakeup Flag	Low-Power Peripheral Wakeup Enable	Power Management Wakeup Enable
Any event for LPTMR0	<a href="#"><code>irq_a</code></a>	N/A	<a href="#"><code>PWRSEQ_LPPWST</code></a> .lptmr0	<a href="#"><code>PWRSEQ_</code></a> .lptmr0	N/A
Any event for LPTMR1	<a href="#"><code>irq_a</code></a>	N/A	<a href="#"><code>PWRSEQ_LPPWST</code></a> .lptmr1	<a href="#"><code>PWRSEQ_</code></a> .lptmr1	N/A

## 18.7 Operating Modes

Multiple operating modes are supported. The availability of some operating modes is dependent on the device and package-specific implementation of the external input and output signals. Refer to the data sheet for I/O signal configurations and alternate functions for each Timer instance.

*Figure 18-3: Timer I/O Signal Naming Conventions*



In *Table 18-5*, *Table 18-6*, and *Table 18-7*, 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 18-3* for details of the timer's naming convention for I/O signals.

*Table 18-5: MAX78000 Operating Mode Signals for Timer 0 and Timer 1*

Timer Mode	TMRO/TMR1 <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i>	I/O Signal Name <sup>†</sup>	Pin Required
<i>One-Shot Mode (0)</i>	TimerA Output Signal	TMRny_IOA	Optional
	TimerA Complementary Output Signal	TMRny_IOAN	Optional
	TimerB Output Signal	TMRny_IOB	Optional
	TimerB Complementary Output Signal	TMRny_IOBN	Optional
<i>Continuous Mode (1)</i>	TimerA Output Signal	TMRny_IOA	Optional
	TimerA Complementary Output Signal	TMRny_IOAN	Optional
	TimerB Output Signal	TMRny_IOB	Optional
	TimerB Complementary Output Signal	TMRny_IOBN	Optional
<i>Counter Mode (2)</i>	TimerA Input Signal	TMRny_IOA	Yes
	TimerB Input Signal	TMRny_IOB	Yes
<i>Capture Mode (4)</i>	TimerA Input Signal	TMRny_IOA	Yes
	TimerB Input Signal	TMRny_IOB	Yes
<i>Compare Mode (5)</i>	TimerA Output Signal	TMRny_IOA	Optional
	TimerA Complementary Output Signal	TMRny_IOAN	Optional
	TimerB Output Signal	TMRny_IOB	Optional
	TimerB Complementary Output Signal	TMRny_IOBN	Optional
<i>Gated Mode (6)</i>	TimerA Input Signal	TMRny_IOA	Yes
	TimerB Input Signal	TMRny_IOB	Yes
<i>Capture/Compare Mode (7)</i>	TimerA Input Signal	TMRny_IOA	Yes
	TimerB Input Signal	TMRny_IOB	Yes

Timer Mode	TMR0/TMR1 <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i>	I/O Signal Name <sup>†</sup>	Pin Required
<i>Dual Edge Capture Mode (8)</i>	TimerA Input Signal	TMRny_IOA	Yes
	TimerB Input Signal	TMRny_IOB	Yes
Reserved (9 - 13)	-	-	-
<i>Inactive Gated Mode (14)</i>	TimerA Input Signal	TMRny_IOA	Yes
	TimerB Input Signal	TMRny_IOB	Yes
Reserved (15)	-	-	-

<sup>†</sup> See Figure 18-3 for details on the timer I/O signal naming convention and the device data sheet for the alternate functions.

Table 18-6: MAX78000 Operating Mode Signals for Timer 2 and Timer 3

Timer Mode	TMR2/TMR3 <i>TMRn_CTRL1.outen_a = 0</i> <i>TMRn_CTRL1.outben_a = 0</i>	I/O Signal Name <sup>†</sup>	Required?
<i>One-Shot Mode (0)</i>	TimerA Output Signal TMRn_CTRL1	TMRny_IOA	Optional
	TimerB Output Signal	TMRny_IOB	Optional
<i>Continuous Mode (1)</i>	TimerA Output Signal	TMRny_IOA	Optional
	TimerB Output Signal	TMRny_IOB	Optional
<i>Counter Mode (2)</i>	TimerA Input Signal	TMRny_IOA	Yes
	TimerB Input Signal	TMRny_IOB	Yes
<i>Capture Mode (4)</i>	TimerA Input Signal	TMRny_IOA	Yes
	TimerB Input Signal	TMRny_IOB	Yes
<i>Compare Mode (5)</i>	TimerA Output Signal	TMRny_IOA	Optional
	TimerB Output Signal	TMRny_IOB	Optional
<i>Gated Mode (6)</i>	TimerA Input Signal	TMRny_IOA	Yes
	TimerB Input Signal	TMRny_IOB	Yes
<i>Capture/Compare Mode (7)</i>	TimerA Input Signal	TMRny_IOA	Yes
	TimerB Input Signal	TMRny_IOB	Yes
<i>Dual Edge Capture Mode (8)</i>	TimerA Input Signal	TMRny_IOA	Yes
	TimerB Input Signal	TMRny_IOB	Yes
Reserved (0 - 13)	-	-	-
<i>Inactive Gated Mode (14)</i>	TimerA Input Signal	TMRny_IOA	Yes
	TimerB Input Signal	TMRny_IOB	Yes
Reserved (15)	-	-	-

<sup>†</sup> See Figure 18-3 for details on the timer I/O signal naming convention and the device data sheet for the alternate functions.

Table 18-7: MAX78000 Operating Mode Signals for Low-Power Timer 0 and Low-Power Timer 1

Timer mode	TMR4/TMR5 <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i>	I/O Signal Name <sup>†</sup>	Required?
<i>One-Shot Mode (0)</i>	TimerA Output Signal	LPTMRny_IOB	Optional
<i>Continuous Mode (1)</i>	TimerA Output Signal	LPTMRny_IOB	Optional
<i>Counter Mode (2)</i>	TimerA Input Signal	LPTMRny_IOB	Yes
<i>Capture Mode (4)</i>	TimerA Input Signal	LPTMRny_IOB	Yes

Timer mode	TMR4/TMR5 <i>TMRn_CTRL1.outen = 0</i> <i>TMRn_CTRL1.outben = 0</i>	I/O Signal Name <sup>†</sup>	Required?
<i>Compare Mode (5)</i>	TimerA Output Signal	LPTMRny_IOB	Optional
<i>Gated Mode (6)</i>	TimerA Input Signal	LPTMRny_IOB	Yes
<i>Capture/Compare Mode (7)</i>	TimerA Input Signal	LPTMRny_IOB	Yes
<i>Dual Edge Capture Mode (8)</i>	TimerA Input Signal	LPTMRny_IOB	Yes
Reserved (9 - 13)	-	-	-
<i>Inactive Gated Mode (14)</i>	TimerA Input Signal	LPTMRny_IOB	Yes
Reserved (15)	-	-	-

<sup>†</sup> See [Figure 18-3](#) for details on the timer I/O signal naming convention and the device data sheet for the alternate functions.

### 18.7.1 One-Shot Mode (0)

In one-shot mode, the timer peripheral increments the timer's *TMRn\_CNT.count* field until it reaches the timer's *TMRn\_CMP.compare* 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.count = TMRn\_CMP.compare*. The timer peripheral hardware automatically performs the following actions at the end of the timer period:

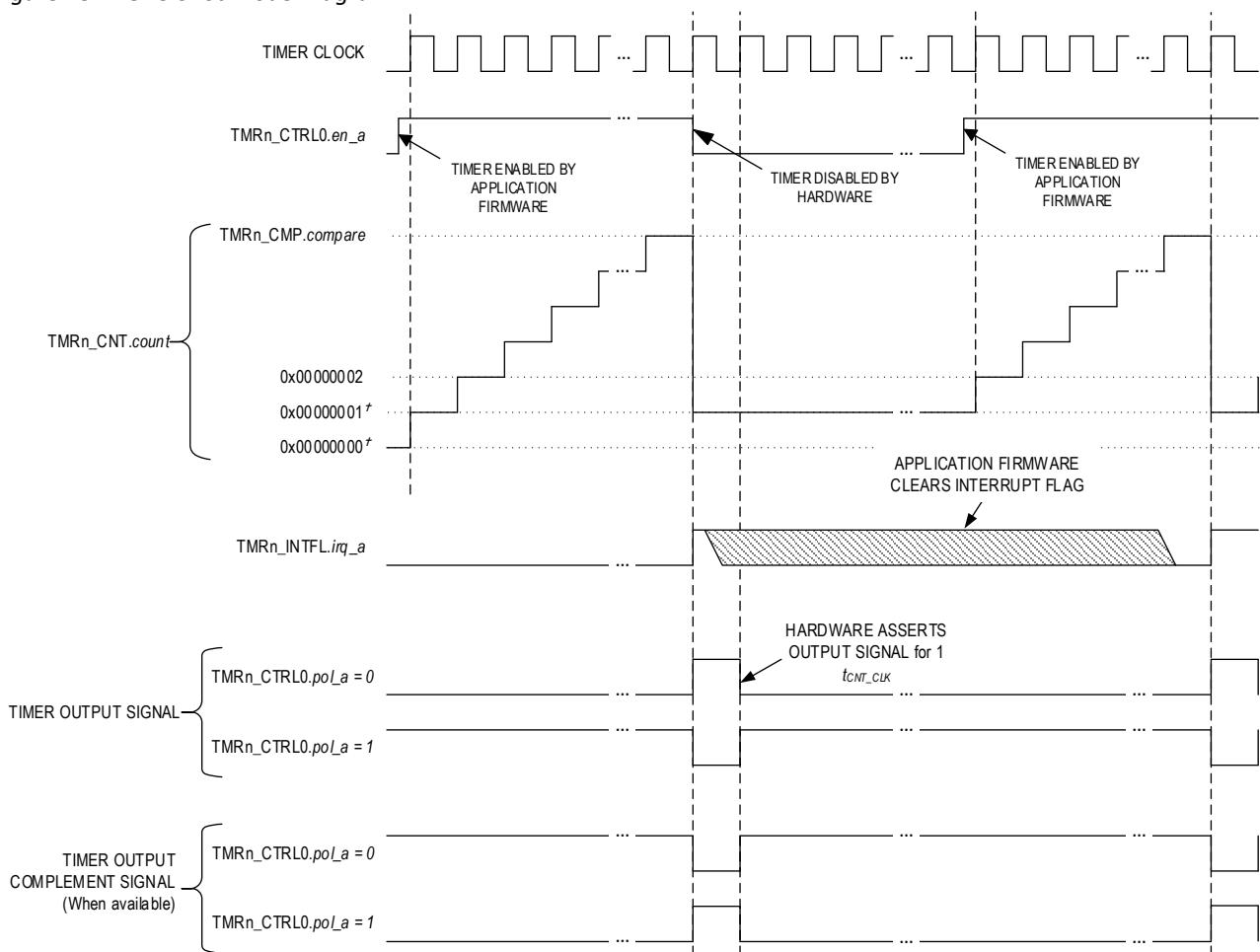
- The *TMRn\_CNT.count* field is set to 0x0000 0001,
- the timer is disabled (*TMRn\_CTRL0.en* = 0),
- the timer output, if enabled, is driven to its active state for one timer clock period,
- the *TMRn\_INTFL.irq* field is set to 1 to indicate a timer interrupt event occurred.

The timer period is calculated using [Equation 18-2](#).

*Equation 18-2: One-shot Mode Timer Period*

$$\text{One-shot mode timer period in seconds} = \frac{\text{TMRn\_CMP} - \text{TMRn\_CNT}_{\text{INITIAL\_VALUE}} + 1}{f_{\text{CNT\_CLK}}(\text{Hz})}$$

*Figure 18-4: One-Shot Mode Diagram*



This example uses the following configuration in addition to the settings shown above:

TMR<sub>n</sub>\_CTRL1.cascade = 1 (32-bit Cascade Timer)  
 TMR<sub>n</sub>\_CTRL0.mode\_a = 0 (One-shot)

<sup>\*</sup>TMR<sub>n</sub>\_CNT.count defaults to 0x00000000 on a timer reset. TMR<sub>n</sub>\_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 corresponding field in the `TMRn_CTRL1` register.
7. Write the compare value to the `TMRn_CMP.compare` field.
8. If desired, write an initial value to `TMRn_CNT.count` field.
  - a. This effects only the first period; subsequent timer periods always reset the `TMRn_CNT.count` field to 0x0000 0001.
9. Enable the timer peripheral as described in [Timer Clock Sources](#).

### 18.7.2 Continuous Mode (1)

In continuous mode, the `TMRn_CNT.count` field increments until it matches the `TMRn_CMP.compare` field; the `TMRn_CNT.count` field is then set to 0x0000 0001 and the count continues to increment. Optionally, application firmware 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.count = TMRn_CMP.compare`).

The timer peripheral hardware automatically performs the following actions on the timer clock cycle after the period ends:

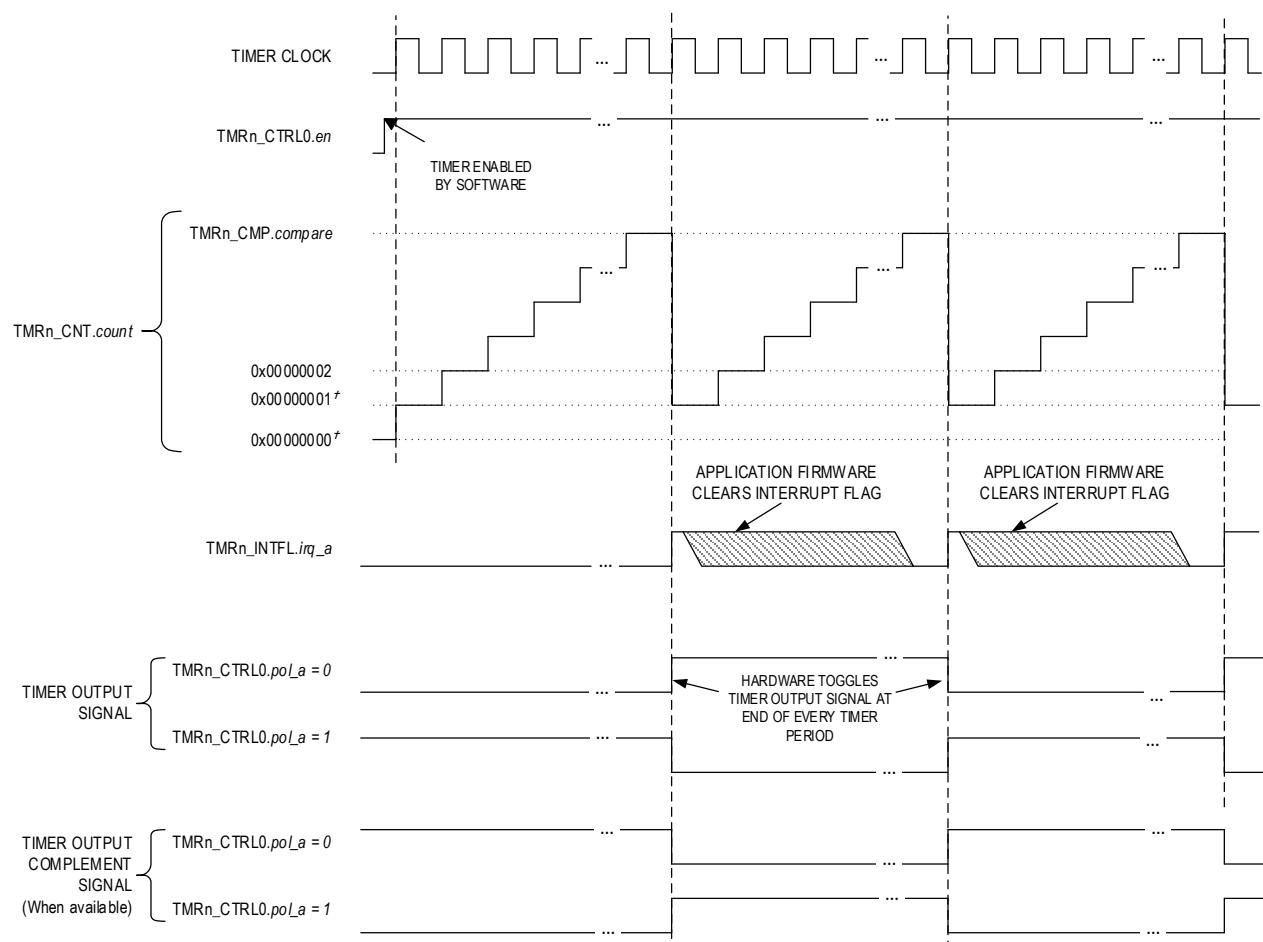
- The `TMRn_CNT.count` field is set to 0x0000 0001,
- if the timer output signal is toggled,
- the corresponding `TMRn_INTFL.irq` field will be set to 1 to indicate a timer interrupt event occurred.

The continuous mode timer period is calculated using [Equation 18-3: Continuous Mode Timer Period](#).

*Equation 18-3: Continuous Mode Timer Period*

$$\text{Continuous mode timer period (s)} = \frac{TMRn\_CMP - TMRn\_CNT_{\text{INITIAL\_VALUE}} + 1}{f_{\text{CNT\_CLK}} (\text{Hz})}$$

Figure 18-5: Continuous Mode Diagram



This example 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. If, 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.compare` field.
8. If desired, write an initial value to the `TMRn_CNT.count` field.
  - a. This effects only the first period; subsequent timer periods always reset the `TMRn_CNT.count` field to 0x0000 0001.
9. Enable the timer peripheral as described in [Timer Clock Sources](#).

### 18.7.3 Counter Mode (2)

In counter mode, the timer peripheral increments the `TMRn_CNT.count` 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.count` reaches the `TMRn_CMP.compare` field, the hardware automatically sets the interrupt bit to 1 (`TMRn_INTFL.irq`), sets the `TMRn_CNT.count` field to 0x0000 0001, and continues incrementing. The timer can be configured to increment on either the rising edge or falling edge of the timer's input signal, 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 frequency of the timer's input signal ( $f_{CTR\_CLK}$ ) must not exceed 25 percent of the PCLK frequency as shown [Equation 18-4](#).

*Note: If the input signal's frequency is equal to  $f_{PCLK}$ , it is possible the transition can be missed by the timer hardware due to PCLK being an asynchronous internal clock. A minimum of 4 PCLK cycles is required for a count to occur. To guarantee a count occurs, the timer input signal should be greater than 4 PCKL cycles.*

[Equation 18-4: Counter Mode Maximum Clock Frequency](#)

$$f_{CTR\_CLK} \leq \frac{f_{PCLK} \text{ (Hz)}}{4}$$

The timer period ends on the rising edge of PCLK following `TMRn_CNT.count = TMRn_CMP.compare`.

The timer peripheral's hardware automatically performs the following actions at the end of the timer period:

- The `TMRn_CNT.count` 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.*

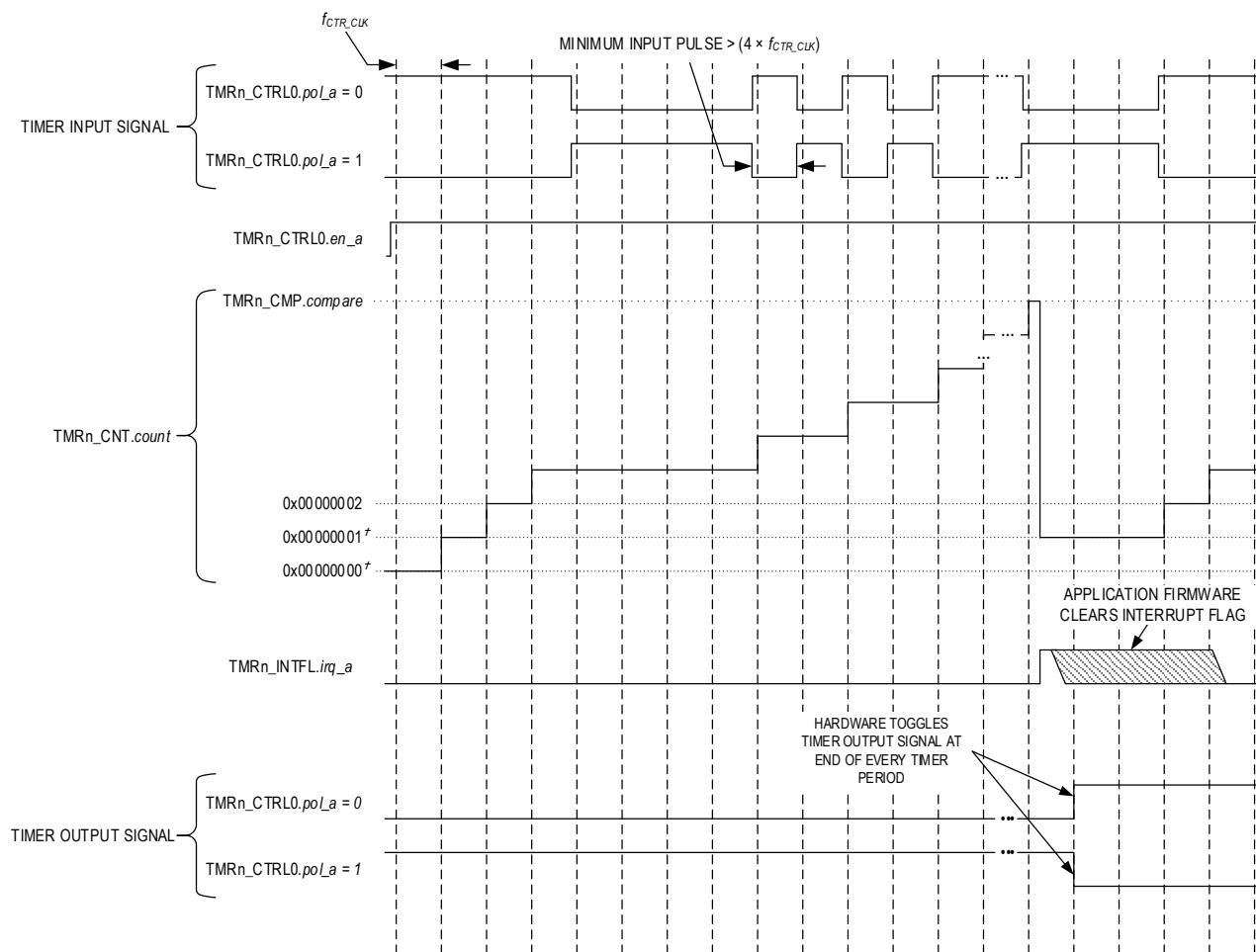
In counter mode, the number of timer input transitions that occurred during a period is equal to the `TMRn_CMP.compare` field's setting. Use [Equation 18-5](#) to determine the number of transitions that occurred prior to the end of the timer's period.

*Note: Equation 18-5 is only valid during an active timer count prior to the end of the timer's period.*

*Equation 18-5: Counter Mode Timer Input Transitions*

$$\text{Counter mode timer input transitions} = \text{TMR\_CNT}_{\text{CURRENT\_VALUE}}$$

*Figure 18-6: Counter Mode Diagram*



This example 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` 0x2 to select Counter mode.
4. Configure the timer input function:
  - a. Set `TMRn_CTRL0.pol` to match the desired (inactive) state.
  - b. Configure the GPIO electrical characteristics as desired.
  - c. Set `TMRn_CTRL1.outen_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.compare`.
6. If desired, write an initial value to `TMRn_CNT.count`. This effects only the first period; subsequent timer periods always reset `TMRn_CNT.count` = 0x0000 0001.
7. Enable the timer peripheral as described in [Timer Clock Sources](#).

#### 18.7.4 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.count` value matches the `TMRn_PWM.pwm`, the timer output signal toggles state. The timer continues counting until it reaches the `TMRn_CMP.compare` value.

The timer period ends on the rising edge of  $f_{CNT\_CLK}$  following `TMRn_CNT.count = TMRn_CMP.compare`.

The timer peripheral automatically performs the following actions at the end of the timer period:

- The `TMRn_CNT.count` is reset to 0x0000 0001 and the timer resumes counting,
- the timer output signal is toggled,
- the corresponding `TMRn_INTFL.irq` field 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.count` value matches the `TMRn_PWM` value. The timer output signal remains high until the `TMRn_CNT.count` value reaches the `TMRn_CMP.compare`, resulting in the timer output signal transitioning low, and the `TMRn_CNT.count` value resetting to 0x0000 0001.

When `TMRn_CTRL0.pol` = 1, the Timer output signal starts high and transitions low when the `TMRn_CNT.count` value matches the `TMRn_PWM` value. The timer output signal remains low until the `TMRn_CNT.count` value reaches `TMRn_CMP.compare`, resulting in the timer output signal transitioning high, and the `TMRn_CNT.count` value resetting to 0x0000 0001.

Complete the following steps to configure a timer for PWM mode and initiate the PWM operation:

1. Disable the timer peripheral as described in [Timer Clock Sources](#).
2. If desired, change the timer clock source as described in [Timer Clock Sources](#).
3. Set 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.count* initial value if desired.
  - a. The initial *TMRn\_CNT.count* value only effects the initial period in PWM mode with subsequent periods always setting *TMRn\_CNT.count* to 0x0000 0001.
9. Set the *TMRn\_PWM* value to the transition period count.
10. Set the *TMRn\_CMP.compare* value for the PWM second transition period. Note: *TMRn\_CMP.compare* must be greater than the *TMRn\_PWM* value.
11. If using the timer interrupt, set the interrupt priority and enable the interrupt.
12. Enable the timer peripheral as described in [Timer Clock Sources](#).

[Equation 18-6](#) shows the formula for calculating the timer PWM period.

[Equation 18-6: Timer PWM Period](#)

$$\text{PWM period (s)} = \frac{\text{TMRn_CNT}}{f_{\text{CNT_CLK}} (\text{Hz})}$$

If an initial starting value other than 0x0000 0001 is loaded into the *TMRn\_CNT.count* register, use the one-shot mode equation, [Equation 18-2](#), 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 18-7](#).

[Equation 18-7: Timer PWM Output High Time Ratio with Polarity 0](#)

$$\text{PWM output high time ratio (\%)} = \frac{(\text{TMR_CMP} - \text{TMR_PWM})}{\text{TMR_CMP}} \times 100$$

If *TMRn\_CTRL0.pol* is set to 1, the ratio of the PWM output high time to the total period is calculated using [Equation 18-8](#).

[Equation 18-8: Timer PWM Output High Time Ratio with Polarity 1](#)

$$\text{PWM output high time ratio (\%)} = \frac{\text{TMR_PWM}}{\text{TMR_CMP}} \times 100$$

### 18.7.5 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 18-9](#) shows the formula for calculating the capture event's elapsed time.

If a capture event does not occur prior to the timer's count value reaching the timer's compare value (*TMRn\_CNT.count* = *TMRn\_CMP.compare*), a rollover event occurs. Both the capture event and the rollover event set the timer's interrupt flag, *TMRn\_INTFL.irq* to 1 and result 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 prior to 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.*

#### 18.7.5.1 Capture Event

When a capture event occurs, the timer hardware, on the next timer clock cycle, automatically performs the following actions:

- The `TMRn_CNT.count` value is copied to the `TMRn_PWM.pwm` field,
- the `TMRn_INTFL.irq` field is set to 1,
- and 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 18-9: Capture Mode Elapsed Time Calculation in Seconds*

*Capture elapsed time (s)*

$$= \frac{(TMR\_PWM - TMR\_CNT_{INITIAL\_VALUE}) + ((\text{Number of rollover events}) \times (TMR\_CMP - TMR\_CNT_{INITIAL\_VALUE}))}{f_{CNT\_CLK}}$$

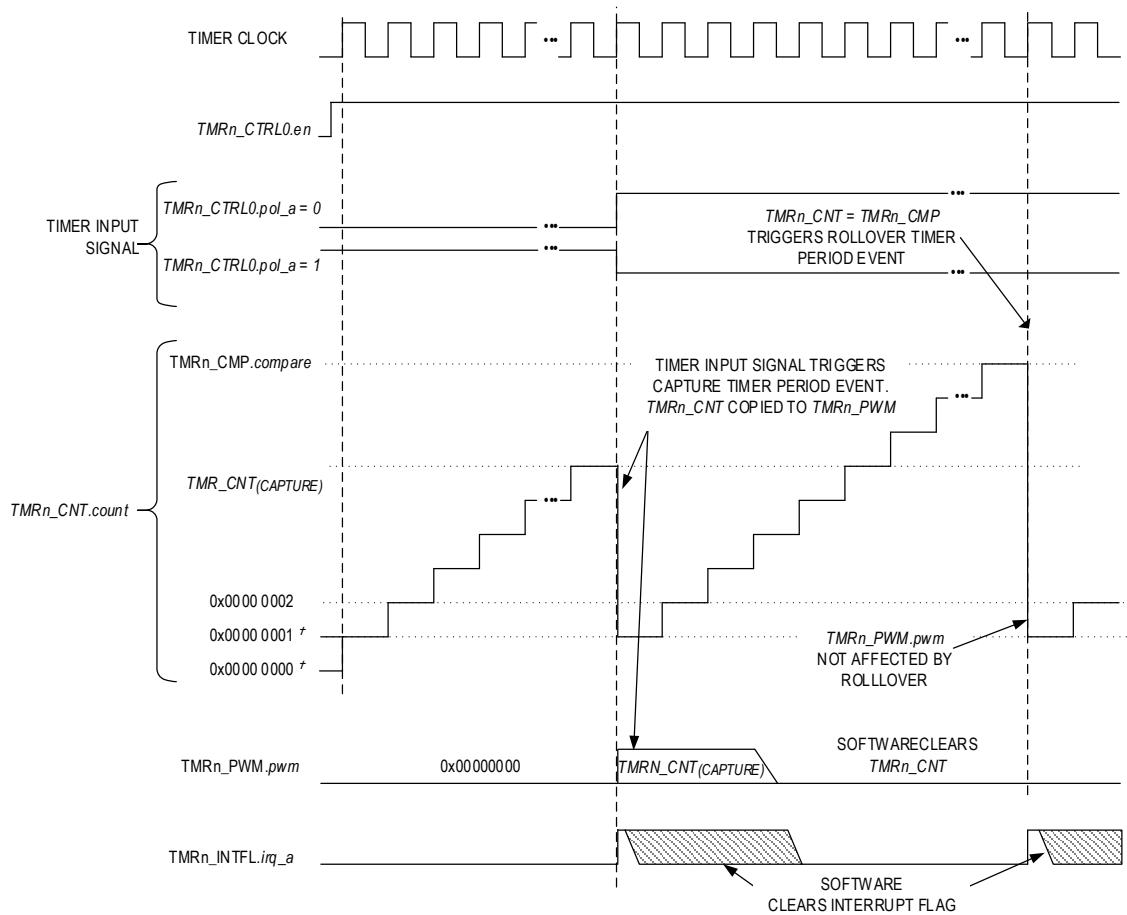
*Note: The capture elapsed time calculation is only valid after the capture event occurs and the timer stores the captured count in the `TMRn_PWM` register.*

#### 18.7.5.2 Rollover Event

A rollover event occurs when the timer's count value reaches the timer's compare value (`TMRn_CNT.count = TMRn_CMP.compare`). 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.count` field is set to 0x0000 0001,
- the `TMRn_INTFL.irq` field is set to 1,
- and the timer remains enabled and continues counting.

Figure 18-7: Capture Mode Diagram



This example uses the following configuration in addition to the settings shown above:

*TMR<sub>n</sub>\_CTRL1.cascade* = 1 (32-bit Cascade Timer)  
*TMR<sub>n</sub>\_CTRL0.mode\_a* = 2 (Counter)

<sup>†</sup> *TMR<sub>n</sub>\_CNT.count* defaults to 0x00000000 on a timer reset. *TMR<sub>n</sub>\_CNT.count* reloads to 0x00000001 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.count`, if desired.
  - a. This effects only the first period; subsequent timer periods always reset `TMRn_CNT.count` = 0x0000 0001.
6. Write the compare value to the `TMRn_CMP.compare` field.
7. Select the capture event by setting `TMRn_CTRL1.capecvents`.
8. Enable the timer peripheral as described in [Timer Clock Sources](#).

The timer period is calculated using the following equation:

*Equation 18-10: Capture Mode Elapsed Time Calculation in Seconds*

$$\text{Capture elapsed time in seconds} = \frac{TMR\_PWM - TMR\_CNT_{\text{INITIAL\_VALUE}}}{f_{CNT\_CLK}}$$

*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.*

### 18.7.6 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.count` = `TMRn_CMP.compare`.

The timer peripheral automatically performs the following actions when a timer period event:

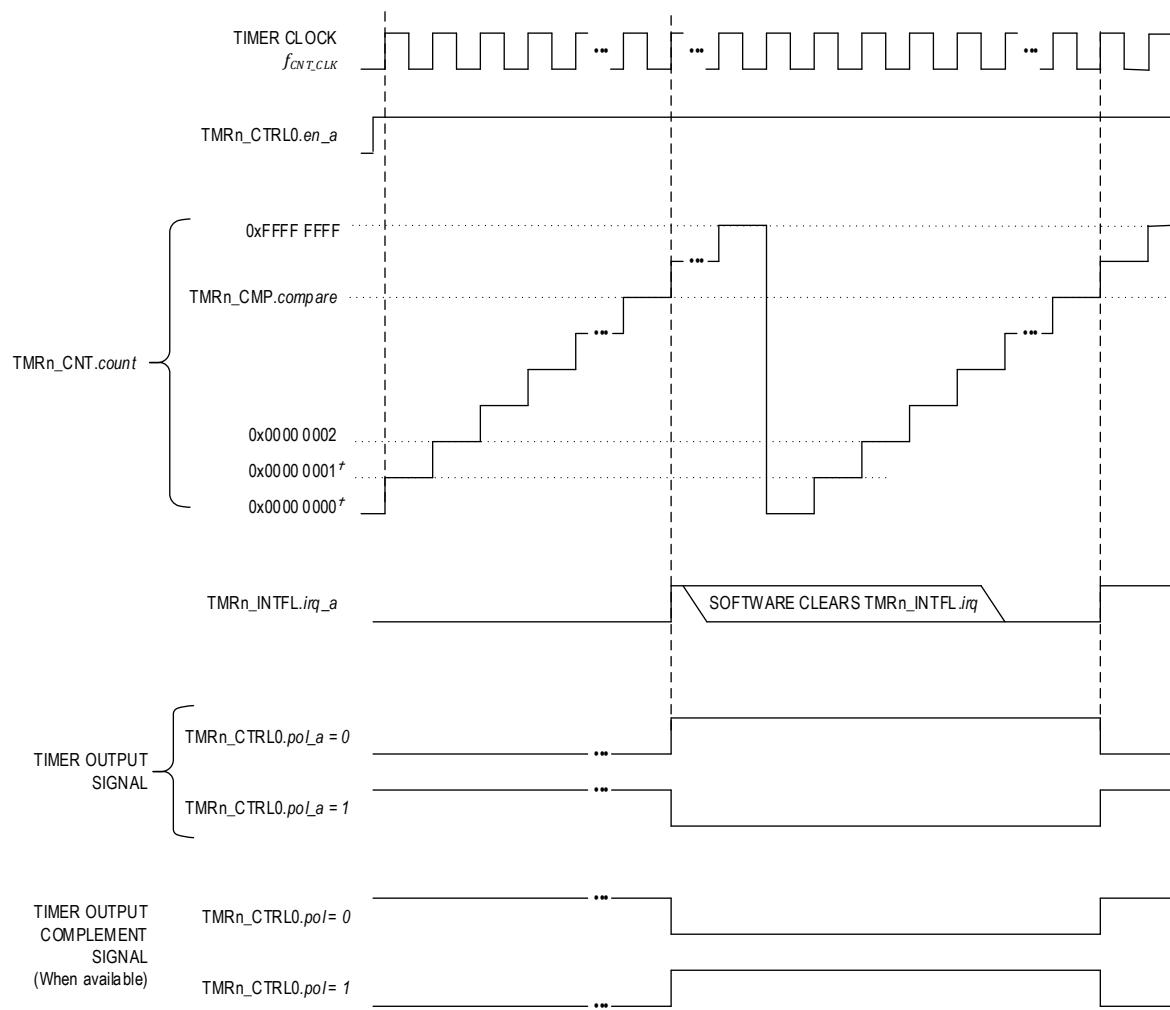
- Unlike other modes, `TMRn_CNT.count` is reset to 0x0000 0000 not 0x0000 0001 at the end of the timer period.
- The corresponding `TMRn_INTFL.irq` field is set to 1 to indicate a timer interrupt event occurred.
- The hardware toggles the state of the timer output signal. The timer output pin will change state if the timer output is enabled.
- The timer remains enabled and continues incrementing.

The compare Mode timer period is calculated using [Equation 18-12: Capture Mode Elapsed Time](#).

*Equation 18-11: Compare Mode Timer Period*

$$\text{Compare mode timer period in second} = \frac{(TMR\_CMP - TMR\_CNT_{\text{INITIAL\_VALUE}} + 1)}{f_{CNT\_CLK}(\text{Hz})}$$

Figure 18-8: Compare Mode Diagram



This example 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)

<sup>t</sup>TMRn\_CNT.count defaults to 0x0000 0000 on a timer reset. TMRn\_CNT.count reloads to 0x0000 0001 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 corresponding field in the `TMRn_CTRL1` register.
8. Write the compare value to `TMRn_CMP.compare`.
9. If desired, write an initial value to `TMRn_CNT.count`.
  - a. This effects only the first period; subsequent timer periods always reset `TMRn_CNT.count` = 0x0000 0001.
10. Enable the timer peripheral as described in [Timer Clock Sources](#).

### 18.7.7 Gated Mode (6)

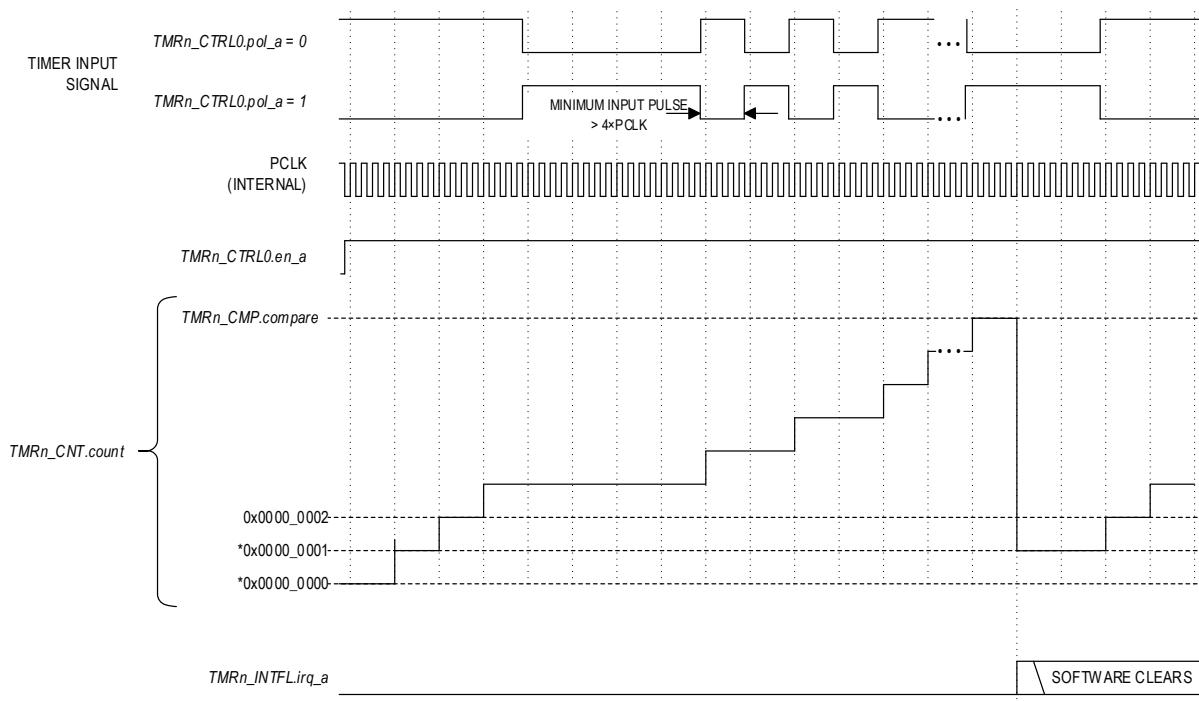
Gated mode is similar to continuous mode, except that `TMRn_CNT.count` only increments when the timer input signal is in its active state.

The timer period ends on the timer clock following `TMRn_CNT.count` = `TMRn_CMP.compare`.

The timer peripheral automatically performs the following actions at the end of the timer period:

- The `TMRn_CNT.count` 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 will be set to 1 to indicate a timer interrupt event occurred.

Figure 18-9: Gated Mode Diagram



This example 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)

<sup>†</sup> *TMRn\_CNT.count* defaults to 0x0000 0000 on a timer reset. *TMRn\_CNT.count* reloads to 0x0000 0001 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.count* field.
  - a. This only effects the first period; subsequent timer periods always reset *TMRn\_CNT.count* = 0x0000 0001.
6. Write the compare value to *TMRn\_CMP.compare*.
7. Enable the timer peripheral as described in [Timer Clock Sources](#).

### 18.7.8 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.

Each subsequent transition, after the first transition of the timer input signal, captures the *TMRn\_CNT.count* value, writing it to the *TMRn\_PWM.pwm* register (capture event). When a capture event occurs, a timer interrupt is generated, the *TMRn\_CNT.count* value is reset to 0x0000\_0001, and the timer resumes counting.

If no capture event occurs, the timer counts up to *TMRn\_CMP.compare*. At the end of the cycle where the *TMRn\_CNT.count* equals the *TMRn\_CMP.compare*, a timer interrupt is generated, the *TMRn\_CNT.count* value is reset to 0x0000 0001, and the timer resumes counting.

The timer period ends when the selected transition occurs on the timer pin, or on the clock cycle following *TMRn\_CNT.count = TMRn\_CMP.compare*.

The actions performed at the end of the timer period are dependent on the event that ended the timer period:

If the end of the timer period was caused by a transition on the timer pin, the hardware automatically performs the following:

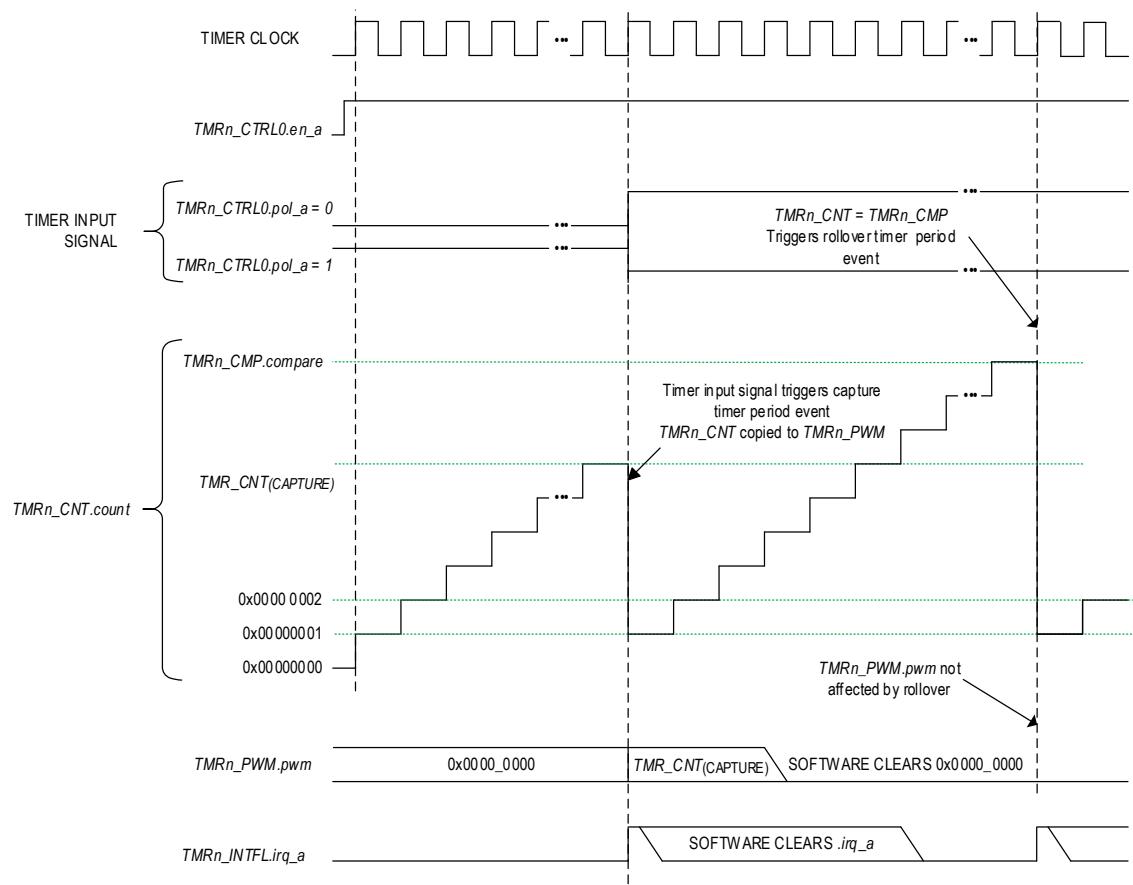
- The value in *TMRn\_CNT.count* field is copied to the *TMRn\_PWM.pwm* field,
- the *TMRn\_CNT.count* field is set to 0x0000 0001,
- the timer remains enabled and continues incrementing,
- the corresponding *TMRn\_INTFL.irq* field is set to 1 to indicate a timer interrupt event occurred.

In capture/compare mode, the elapsed time from the timer start to the capture event is calculated using [Equation 18-12](#).

*Equation 18-12: Capture Mode Elapsed Time*

$$\text{Capture elapsed time (seconds)} = \frac{TMRn\_PWM - TMRn\_CNT_{INITIAL\_CNT\_VALUE}}{f_{CNT\_CLK}(\text{Hz})}$$

Figure 18-10: Capture/Compare Mode Diagram



This example uses the following configuration in addition to the settings shown above:

*TMR<sub>n</sub>\_CTRL1.cascade* = 1 (32-bit Cascade Timer)  
*TMR<sub>n</sub>\_CTRL0.mode<sub>a</sub>* = 7 (Capture/Compare)

<sup>†</sup> *TMR<sub>n</sub>\_CNT.count* defaults to 0x0000\_0000 on a timer reset. *TMR<sub>n</sub>\_CNT.count* reloads to 0x0000\_0001 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.count\*](#) field.
  - a. This effects only the first period; subsequent timer periods always reset [\*TMRn\\_CNT.count\*](#) = 0x0000 0001.
6. Write the compare value to [\*TMRn\\_CMP.compare\*](#).
7. Enable the timer peripheral as described in [Timer Clock Sources](#).

*Note: No interrupt is generated by the first transition of the input signal.*

### 18.7.9 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.

### 18.7.10 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.

## 18.8 Registers

See [Table 2-4](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 18-8](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn\_CTRL resolves to PERIPHERAL0\_CTRL and PERIPHERAL1\_CTRL for instances 0 and 1, respectively.

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

*Table 18-8: Timer Register Summary*

Offset	Register	Description
[0x0000]	<a href="#"><i>TMRn_CNT</i></a>	Timer Counter Register
[0x0004]	<a href="#"><i>TMRn_CMP</i></a>	Timer Compare Register
[0x0008]	<a href="#"><i>TMRn_PWM</i></a>	Timer PWM Register
[0x000C]	<a href="#"><i>TMRn_INTFL</i></a>	Timer Interrupt Register
[0x0010]	<a href="#"><i>TMRn_CTRL0</i></a>	Timer Control Register
[0x0014]	<a href="#"><i>TMRn_NOLCMP</i></a>	Timer Non-Overlapping Compare Register
[0x0018]	<a href="#"><i>TMRn_CTRL1</i></a>	Timer Configuration Register
[0x001C]	<a href="#"><i>TMRn_WKFL</i></a>	Timer Wakeup Status Register

### 18.8.1 Register Details

*Table 18-9: Timer Count Register*

Timer Count			TMRn_CNT		[0x0000]
Bits	Field	Access	Reset	Description	
31:0	count	R/W	0	<b>Timer Count</b> This field increments at a rate dependent on the selected timer operating mode. The function of the bits in this field are dependent on the 32-bit/16-bit configuration. Reads of this register always return the current value.	

*Table 18-10: Timer Compare Register*

Timer Compare			TMRn_CMP		[0x0004]
Bits	Field	Access	Reset	Description	
31:0	compare	R/W	0	<b>Timer Compare Value</b> The value in this register is used as the compare value for the timer's count value. The compare field meaning is determined by the specific mode of the timer. See the timer mode's detailed configuration section for compare usage and meaning.	

*Table 18-11: Timer PWM Register*

Timer PWM			TMRn_PWM		[0x0008]
Bits	Field	Access	Reset	Description	
31:0	pwm	R/W	0	<b>Timer PWM Match</b> In PWM mode, this field sets the count value for the first transition period of the PWM cycle. At the end of the cycle when <i>TMRn_CNT.count</i> = <i>TMRn_CMP.compare</i> , the PWM output transitions to the second period of the PWM cycle. The second PWM period count is stored in <i>TMRn_CMP.compare</i> . <i>TMRn_PWM.pwm</i> must be less than <i>TMRn_CMP.compare</i> for PWM mode operation.  <b>Timer Capture Value</b> In capture, compare, and capture/compare modes, this field is used to store the <i>TMRn_CNT.count</i> value when a Capture, Compare, or Capture/Compare event occurs.	

*Table 18-12: Timer Interrupt Register*

Timer Interrupt			TMRn_INTFL		[0x000C]
Bits	Field	Access	Reset	Description	
31:26	-	RO	0	<b>Reserved</b>	
24	wr_dis_b	R/W	0	<b>TimerB Write Protect in Dual Timer Mode</b> Set this field to 0 to write protect the TimerB fields in the <i>TMRn_CNT.count[31:16]</i> and <i>TMRn_PWM.pwm[31:16]</i> . When this field is set to 0, 32-bit writes to the <i>TMRn_CNT</i> and <i>TMRn_PWM</i> registers only modify the lower 16-bits associated with TimerA.  0: Enabled 1: Disabled  <i>Note: This field always reads 0 if the timer is configured as a 32-bit cascade timer.</i>	

Timer Interrupt				TMRn_INTFL	[0x000C]
Bits	Field	Access	Reset	Description	
25	wrdone_b	R	0	<b>TimerB Write Done</b> This field is cleared to 0 by the hardware when the software performs a write to <i>TMRn_CNT.count[31:16]</i> or <i>TMRn_PWM.pwm[31:16]</i> when in dual timer mode. Wait until the field is set to 1 before proceeding. 0: Operation in progress. 1: Operation complete.	
23:17	-	RO	0	<b>Reserved</b>	
16	irq_b	R/W1C	0	<b>TimerB Interrupt Event</b> 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	<b>Reserved</b>	
9	wr_dis_a	R/W	0	<b>TimerB Dual Timer Mode Write Protect</b> This field disables write access to the <i>TMRn_CNT.count[31:16]</i> and <i>TMRn_PWM.pwm[31:16]</i> fields so that only the 16 bits associated with updating TimerA are modified during writes to the <i>TMRn_CNT</i> and <i>TMRn_PWM</i> registers. 0: Enabled 1: Disabled <i>Note: This field always reads 0 if the timer is configured as a 32-bit cascade timer.</i>	
8	wrdone_a	R	0	<b>TimerA Write Done</b> This field is cleared to 0 by the hardware when firmware performs a write to <i>TMRn_CNT.count[31:16]</i> or <i>TMRn_PWM.pwm[31:16]</i> when in dual 16-bit timer mode. Wait until the field reads 1 before proceeding. 0: Operation in progress 1: Operation complete	
7:1	-	RO	0	<b>Reserved</b>	
0	irq_a	W1C	0	<b>TimerA Interrupt Event</b> This field is set when a TimerA interrupt event occurs. Write 1 to clear. 0: No event 1: Interrupt event occurred	

Table 18-13: Timer Control 0 Register

Timer Control 0				TMRn_CTRL0	[0x0010]
Bits	Field	Access	Reset	Description	
31	en_b	R/W	0	<b>TimerB Enable</b> 0: Disabled 1: Enabled	
30	clken_b	R/W	0	<b>TimerB Clock Enable</b> 0: Disabled 1: Enabled	
29	rst_b	W1	0	<b>TimerB Reset</b> 0: No action 1: Reset TimerB	
28:24	-	RO	0	<b>Reserved</b>	

Timer Control 0				TMRn_CTRL0	[0x0010]																												
Bits	Field	Access	Reset	Description																													
23:20	clkdiv_b	R/W	0	<b>TimerB Prescaler Select</b> The <i>clkdiv_b</i> field selects a prescaler that divides the timer's source clock to set the timer's count clock as follows: $f_{CNT\_CLK} = f_{CLK\_SOURCE} / \text{prescaler}$ See <a href="#">Operating Modes</a> section for details on which timer modes use the prescaler. <table> <tr><td>0:</td><td>1</td></tr> <tr><td>1:</td><td>2</td></tr> <tr><td>2:</td><td>4</td></tr> <tr><td>3:</td><td>8</td></tr> <tr><td>4:</td><td>16</td></tr> <tr><td>5:</td><td>32</td></tr> <tr><td>6:</td><td>64</td></tr> <tr><td>7:</td><td>128</td></tr> <tr><td>8:</td><td>256</td></tr> <tr><td>9:</td><td>512</td></tr> <tr><td>10:</td><td>1024</td></tr> <tr><td>11:</td><td>2048</td></tr> <tr><td>12:</td><td>4096</td></tr> <tr><td>13-15:</td><td>Reserved</td></tr> </table>		0:	1	1:	2	2:	4	3:	8	4:	16	5:	32	6:	64	7:	128	8:	256	9:	512	10:	1024	11:	2048	12:	4096	13-15:	Reserved
0:	1																																
1:	2																																
2:	4																																
3:	8																																
4:	16																																
5:	32																																
6:	64																																
7:	128																																
8:	256																																
9:	512																																
10:	1024																																
11:	2048																																
12:	4096																																
13-15:	Reserved																																
19:16	mode_b	R/W	0	<b>TimerB Mode Select</b> Set this field to the desired mode for TimerB. <table> <tr><td>0:</td><td>One-Shot</td></tr> <tr><td>1:</td><td>Continuous</td></tr> <tr><td>2:</td><td>Counter</td></tr> <tr><td>3:</td><td>PWM</td></tr> <tr><td>4:</td><td>Capture</td></tr> <tr><td>5:</td><td>Compare</td></tr> <tr><td>6:</td><td>Gated</td></tr> <tr><td>7:</td><td>Capture/Compare</td></tr> <tr><td>8:</td><td>Dual-Edge Capture</td></tr> <tr><td>9-11:</td><td>Reserved</td></tr> <tr><td>12:</td><td>Internally Gated</td></tr> <tr><td>13-15:</td><td>Reserved</td></tr> </table>		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				
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	<b>TimerA Enable</b> 0: Disabled 1: Enabled																													
14	clken_a	R/W	0	<b>TimerA Clock Enable</b> 0: Disabled 1: Enabled																													
13	rst_a	R/W1O	0	<b>TimerA Reset</b> 0: No action 1: Reset TimerA																													

Timer Control 0			TMRn_CTRL0		[0x0010]																												
Bits	Field	Access	Reset	Description																													
12	pwmckbd_a	R/W	1	<b>TimerA PWM Output <math>\phi A'</math> Disable</b> 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	<b>TimerA PWM Output <math>\phi A'</math> Polarity Bit</b> 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	<b>TimerA PWM Output <math>\phi A</math> Polarity Bit</b> Set this field to 1 to invert the PWM $\phi A$ signal. 0: Do not invert the $\phi A$ PWM output signal. 1: Invert the $\phi A$ output signal.																													
9	pwmsync_a	R/W	0	<b>TimerA/TimerB PWM Synchronization Mode</b> 0: Disabled 1: Enabled																													
8	pol_a	R/W	0	<b>TimerA Polarity</b> 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. Refer to the <a href="#">Operating Modes</a> section for details on the mode selected.																													
7:4	clkdiv_a	R/W	0	<b>TimerA Prescaler Select</b> The <i>clkdiv_a</i> field selects a prescaler that divides the timer's clock source to set the timer's count clock as follows: $f_{CNT\_CLK} = f_{CLK\_SOURCE} / \text{prescaler}$ See the <a href="#">Operating Modes</a> section to determine which modes use the prescaler. <table> <tbody> <tr><td>0:</td><td>1</td></tr> <tr><td>1:</td><td>2</td></tr> <tr><td>2:</td><td>4</td></tr> <tr><td>3:</td><td>8</td></tr> <tr><td>4:</td><td>16</td></tr> <tr><td>5:</td><td>32</td></tr> <tr><td>6:</td><td>64</td></tr> <tr><td>7:</td><td>128</td></tr> <tr><td>8:</td><td>256</td></tr> <tr><td>9:</td><td>512</td></tr> <tr><td>10:</td><td>1024</td></tr> <tr><td>11:</td><td>2048</td></tr> <tr><td>12:</td><td>4096</td></tr> <tr><td>13-15:</td><td>Reserved</td></tr> </tbody> </table>		0:	1	1:	2	2:	4	3:	8	4:	16	5:	32	6:	64	7:	128	8:	256	9:	512	10:	1024	11:	2048	12:	4096	13-15:	Reserved
0:	1																																
1:	2																																
2:	4																																
3:	8																																
4:	16																																
5:	32																																
6:	64																																
7:	128																																
8:	256																																
9:	512																																
10:	1024																																
11:	2048																																
12:	4096																																
13-15:	Reserved																																

Timer Control 0			TMRn_CTRL0		[0x0010]
Bits	Field	Access	Reset	Description	
3:0	mode_a	R/W	0	<b>TimerA Mode Select</b> 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 for Future Use 12: Internally Gated 13-15: Reserved for Future Use	

Table 18-14: Timer Non-Overlapping Compare Register

Timer Non-Overlapping Compare			TMRn_NOLCMP		[0x0014]
Bits	Field	Access	Reset	Description	
31:24	hi_b	R/W	0	<b>TimerA Non-Overlapping High Compare 1</b> 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	<b>TimerA Non-Overlapping Low Compare 1</b> 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 $\phi A'$ .	
15:8	hi_a	R/W	0	<b>TimerA Non-Overlapping High Compare 0</b> 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 $\phi A$ .	
7:0	lo_a	R/W	0	<b>TimerA Non-Overlapping Low Compare 0</b> 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 $\phi A'$ .	

Table 18-15: Timer Control 1 Register

Timer Control 1			TMRn_CTRL1		[0x0018]
Bits	Field	Access	Reset	Description	
31	cascade	R/W	0	<b>32-bit Cascade Timer Enable</b> 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	outben_b	R/W	0	<b>TimerB Output B Enable</b> Reserved for future use	

Timer Control 1				TMRn_CTRL1	[0x0018]
Bits	Field	Access	Reset	Description	
29	outen_b	R/W	0	<b>TimerB Output Enable</b> Reserved for future use	
28	we_b	R/W	0	<b>TimerB Wakeup Function</b> 0: Disabled 1: Enabled	
27	sw_capevent_b	R/W	0	<b>TimerB Software Event Capture</b> 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	<b>TimerB Event Capture Selection</b> Set this field to the desired capture event source. See <a href="#">Table 18-2</a> for available capture event 0 and capture event 1 options. 0-3: Reserved	
24	ie_b	R/W	0	<b>TimerB Interrupt Enable</b> 0: Disabled 1: Enabled	
23	negtrig_b	R/W	0	<b>TimerB Negative Edge Trigger for Event</b> 0: Rising-edge trigger 1: Falling-edge trigger	
22:20	event_sel_b	R/W	0	<b>TimerB Event Selection</b> 0: Event disabled 1-7: Reserved	
19	clkrdy_b	RO	0	<b>TimerB Clock Ready Status</b> 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	<b>TimerB Clock Enable Status</b> This field indicates the status of the timer enable. 0: Timer not enabled or synchronization in progress 1: Timer is enabled	
17:16	clksel_b	R/W	0	<b>TimerB Clock Source</b> See <a href="#">Table 18-1</a> 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	<b>Reserved</b>	
14	outben_a	R/W	0	<b>Output B Enable</b> Reserved for future use	
13	outen_a	R/W	0	<b>Output Enable</b> Reserved for future use	

Timer Control 1				TMRn_CTRL1	[0x0018]
Bits	Field	Access	Reset	Description	
12	we_a	R/W	0	<b>TimerA Wakeup Function</b> 0: Disabled 1: Enabled.	
11	sw_capevent_a	R/W	0	<b>TimerA Software Event capture</b> 0: No software capture event triggered 1: Trigger software capture event	
10:9	capeventsels_a	R/W	0	<b>TimerA Event capture Selection</b> Set this field to the desired capture event source. See <a href="#">Table 18-2</a> 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	<b>TimerA Interrupt Enable</b> 0: Disabled 1: Enabled	
7	negtrig_a	R/W	0	<b>TimerA Edge Trigger Selection for Event</b> 0: Positive-edge triggered 1: Negative-edge triggered	
6:4	event_sel_a	R/W	0	<b>TimerA Event Selection</b> 0: Event disabled 1-7: Reserved	
3	clkrdy_a	RO	0	<b>TimerA Clock Ready</b> 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	<b>TimerA Clock Enable</b> 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	<b>Clock Source TimerA</b> See <a href="#">Table 18-1</a> 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 18-16: Timer Wakeup Status Register*

Timer Wakeup Status				TMRn_WKFL	[0x001C]
Bits	Field	Access	Reset	Description	
31:17	-	RO	0	<b>Reserved</b>	

Timer Wakeup Status				TMRn_WKFL	[0x001C]
Bits	Field	Access	Reset	Description	
16	b	R/W1C	1	<b>TimerB Wakeup Event</b> This flag is set when a wakeup event occurs for TimerB. Write 1 to clear. 0: No event 1: Wakeup event occurred	
15:1	-	RO	0	<b>Reserved</b>	
0	a	R/W1C	1	<b>TimerA Wakeup Event</b> This flag is set when a wakeup event occurs for TimerA. Write 1 to clear. 0: No event 1: Wakeup event occurred	

## 19. Wakeup Timer (WUT)

The Wakeup Timer (WUT) is a unique instance of a 32-bit timer.

- The wakeup timer uses the is 32.768kHz RTC source.
- Programmable prescaler with values from 1 to 4096.
- Supports two timer modes:
  - ◆ One-Shot: The timer counts up to the terminal value then halts.
  - ◆ Continuous: The timer counts up to the terminal value then repeats.
- Independent interrupt, WUTn\_IRQ.

### 19.1 Basic Operation

The timer modes operate by incrementing the *WUTn\_CNT.count* register. The *WUTn\_CNT.count* register is always readable, even while the timer is enabled and counting.

Each timer mode has a user-configurable timer period, which terminates on the timer clock cycle following the end of timer period condition. The end of a timer period always sets the corresponding interrupt flag and generates a wakeup timer interrupt (WUTn\_IRQ), if enabled.

The timer peripheral automatically sets *WUTn\_CNT.count* to 1 at the end of a timer period, but *WUTn\_CNT.count* is set to 0 following a system reset. This means the first timer period following a system reset is one timer clock longer than subsequent timer periods if *WUTn\_CNT.count* is not initialized to 1 during the timer configuration step.

The timer clock frequency,  $f_{CNT\_CLK}$ , is a divided version of the 32.768kHz RTC clock as shown in *Equation 19-1*.

*Equation 19-1: Wakeup Timer Clock Frequency*

$$f_{CNT\_CLK} = \frac{f_{RTC\_CLK}}{\text{prescaler}}$$

The divisor (prescaler) can be set from 1 to 4096 using the concatenated fields *WUTn\_CTRL.pres3*:*WUTn\_CTRL.pres* as shown in *Table 19-1*.

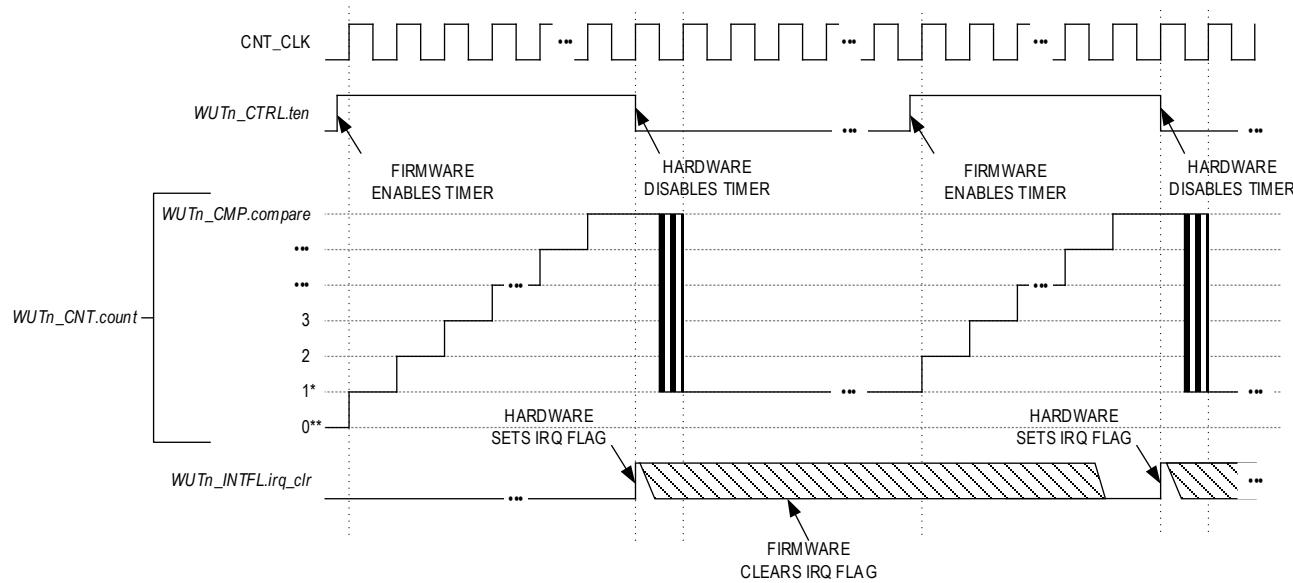
*Table 19-1: MAX78000 WUT Clock Period*

<i>WUTn_CTRL.pres3</i>	<i>WUTn_CTRL.pres</i>	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
1	0b101	4096	8
1	0b110	Reserved	Reserved
1	0b111	Reserved	Reserved

## 19.2 One-Shot Mode (0)

In one-shot mode, the timer peripheral increments the *WUTn\_CNT.count* field until it matches the *WUTn\_CMP.compare* field and then stops incrementing and disables the timer. In this mode, the timer must be re-enabled to start another One-Shot mode event.

*Figure 19-1: One-Shot Mode Diagram*



\* *WUTn\_CNT.count* automatically reloads with 1 at the end of the WUT PERIOD, but software can write any initial value to *WUTn\_CNT.count* prior to enabling the timer.

\*\* The default value of *WUTn\_CNT.count* for the first period after a system reset is 0 unless changed by software.

### 19.2.1 One-Shot Mode Timer Period

The timer period ends on the timer clock when *WUTn\_CNT.count* = *WUTn\_CMP.compare*.

The timer peripheral automatically performs the following actions at the end of the timer period:

1. *WUTn\_CNT.count* is reset to 1.
2. The timer is disabled by setting *WUTn\_CTRL.ten* = 0.
3. The timer interrupt bit *WUTn\_INTFL.irq\_clr* is set. An interrupt will be generated if enabled.

### 19.2.2 One-Shot Mode Configuration

Configure the timer for one-shot mode by doing the following:

1. Set *WUTn\_CTRL.ten* = 0 to disable the timer.
2. Set *WUTn\_CTRL.tmode* to 0 to select one-shot mode.
3. Set *WUTn\_CTRL.pres3:WUTn\_CTRL.pres* to determine the timer period.
4. Enable the interrupt, if desired, and set the interrupt priority.
5. Write an initial value to *WUTn\_CNT.count*, if desired. This effects only the first period; subsequent timer periods always reset *WUTn\_CNT.count* to 1.
6. Write the compare value to *WUTn\_CMP.compare*.
7. Set *WUTn\_CTRL.ten* = 1 to enable the timer.

The timer period is calculated using the following equation:

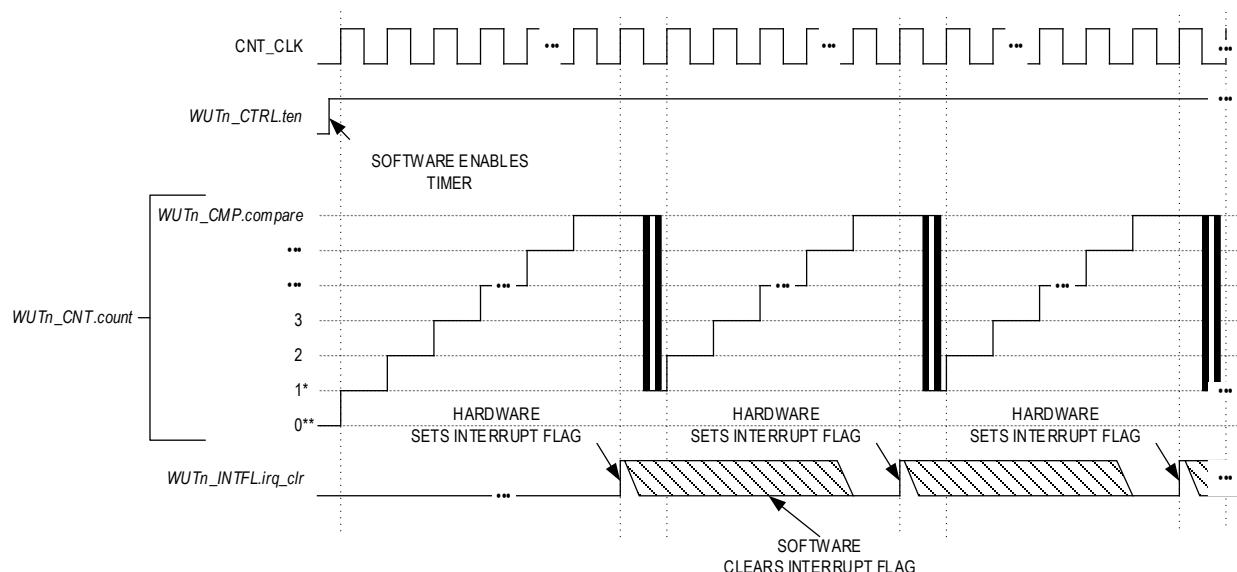
*Equation 19-2: One-Shot Mode Timer Period*

$$\text{One-Shot mode timer period in seconds} = \frac{WUTn\_CMP - WUTn\_CNT_{INITIAL\_VALUE} + 1}{f_{CNT\_CLK} \text{ (Hz)}}$$

### 19.3 Continuous Mode (1)

In continuous mode, the timer peripheral increments *WUTn\_CNT.count* until it matches *WUTn\_CMP.compare* and hardware resets *WUTn\_CNT.count* to 1, and continues incrementing.

*Figure 19-2: Continuous Mode Diagram*



\* *WUTn\_CNT.count* automatically reloads with 1 at the end of the wakeup timer period, but software can write any initial value to *WUTn\_CNT.count* prior to enabling the wakeup timer.

\*\* The value of *WUTn\_CNT.count* for the first period after a system reset is 0 unless changed by software.

### 19.3.1 Continuous Mode Timer Period

The wakeup timer period ends on the timer clock following  $WUTn\_CNT.count = WUTn\_CMP.compare$ .

The wakeup timer peripheral automatically performs the following actions at the end of the timer period:

1.  $WUTn\_CNT.count$  is reset to 1. The wakeup timer remains enabled and continues incrementing.
2. The timer interrupt bit  $WUTn\_INTFL.irq\_clr$  is set. An interrupt is generated if enabled.

### 19.3.2 Continuous Mode Configuration

Configure the timer for continuous mode by performing the steps following:

1. Set  $WUTn\_CTRL.ten = 0$  to disable the timer.
2. Set  $WUTn\_CTRL.tmode$  to 1 to select continuous mode.
3. Set  $WUTn\_CTRL.pres3:WUTn\_CTRL.pres$  to determines the timer period.
4. Enable the interrupt, if desired, and set the interrupt priority.
5. Write an initial value to  $WUTn\_CNT.count$ , if desired. The initial value is only used for the first period; subsequent timer periods always reset the  $WUTn\_CNT.count$  register to 1.
6. Write the compare value to  $WUTn\_CMP.compare$ .
7. Set  $WUTn\_CTRL.ten$  to 1 to enable the timer.

The Continuous Mode Timer Period is calculated using [Equation 19-3](#).

*Equation 19-3: Continuous Mode Timer Period*

$$\text{Continuous Mode Timer Period in seconds} = \frac{WUTn\_CMP - WUTn\_CNT_{INITIAL\_VALUE} + 1}{f_{CNT\_CLK} (\text{Hz})}$$

## 19.4 Registers

See [Table 2-4](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 19-2](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn\_CTRL resolves to PERIPHERAL0\_CTRL and PERIPHERAL1\_CTRL for instances 0 and 1, respectively.

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

*Table 19-2: Wakeup Timer Register Summary*

Offset	Register Name	Description
[0x0000]	<i>WUTn_CNT</i>	Wakeup Timer Counter Register
[0x0004]	<i>WUTn_CMP</i>	Wakeup Timer Compare Register
[0x0008]	<i>WUTn_PWM</i>	Wakeup Timer PWM Register
[0x000C]	<i>WUTn_INTFL</i>	Wakeup Timer Interrupt Register
[0x0010]	<i>WUTn_CTRL</i>	Wakeup Timer Control Register
[0x0014]	<i>WUTn_NOLCMP</i>	Wakeup Timer Non-Overlapping Compare Register

### 19.4.1 Register Details

*Table 19-3: Wakeup Timer Count Register*

Wakeup Timer Count			WUTn_CNT		[0x0000]
Bits	Name	Access	Reset	Description	
31:0	count	R/W	0	<b>Timer Count Value</b> The current count value for the timer. This field increments as the timer counts. Reads of this register are always valid. Prior to writing this field, disable the timer by clearing the bit <a href="#">WUTn_CTRL.ten</a> .	

*Table 19-4: Wakeup Timer Compare Register*

Wakeup Timer Compare			WUTn_CMP		[0x0004]
Bits	Name	Access	Reset	Description	
31:0	compare	R/W	0	<b>Timer Compare Value</b> The value in this register is used as the compare value for the timer's count value. The compare field meaning is determined by the specific mode of the timer. See the timer mode's detailed configuration section for compare usage and meaning.	

*Table 19-5: Wakeup Timer PWM Register*

Wakeup Timer PWM			WUTn_PWM		[0x0008]
Bits	Name	Access	Reset	Description	
31:0	-	RO	0	<b>Reserved</b>	

*Table 19-6: Wakeup Timer Interrupt Register*

Wakeup Timer Interrupt			WUTn_INTFL		[0x000C]
Bits	Name	Access	Reset	Description	
31:1	-	RO	0	<b>Reserved</b>	
0	irq_clr	RW	0	<b>Timer Interrupt Flag</b> If set, this field indicates a wakeup timer interrupt condition occurred. Writing any value to this bit clears the wakeup timer's interrupt. 0: Wakeup Timer interrupt is not active. 1: Wakeup Timer interrupt occurred.	

*Table 19-7: Wakeup Timer Control Register*

Wakeup Timer Control			WUTn_CTRL		[0x0010]
Bits	Name	Access	Reset	Description	
31:9	-	DNM	0	<b>Reserved, Do Not Modify</b>	
8	pres3	R/W	0	<b>Timer Prescaler Select MSB</b> See <a href="#">WUTn_CTRL.pres</a> for details on this field's usage.	
7	ten	R/W	0	<b>Timer Enable</b> 0: Disabled 1: Enabled	
6	tpol	DNM	0	<b>Reserved, Do Not Modify</b>	
5:3	pres	R/W	0	<b>Timer Prescaler Select</b> 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 <a href="#">Equation 19-1</a> . The wakeup timer's prescaler setting is a 4-bit value with <i>pres3</i> as the most significant bit and <i>pres</i> as the three least significant bits. See <a href="#">Table 19-1</a> for details.	

Wakeup Timer Control				WUTn_CTRL	[0x0010]
Bits	Name	Access	Reset	Description	
2:0	tmode	R/W	0	<b>Timer Mode Select</b> Sets the timer's operating mode. 0: One-shot 1: Continuous 2 – 7: Reserved	

Table 19-8: Wakeup Timer Non-Overlapping Compare Register

Wakeup Timer Non-Overlapping Compare				WUTn_NOLCMP	[0x0014]
Bits	Name	Access	Reset	Description	
31:0	-	DNM	0	<b>Reserved, Do Not Modify</b>	

## 20. Watchdog Timer (WDT)

The watchdog timer (WDT) protects against corrupt or unreliable software, power faults, and other system-level problems which may place the IC into an improper operating state. The software must periodically write a special sequence to a dedicated register to confirm the software is operating correctly. Failure to reset the watchdog timer within a user-specified time frame can first generate an interrupt allowing the software the opportunity to identify and correct the problem. In the event the software cannot regain normal operation, as a last resort the watchdog timer can generate a system reset.

Some instances provide a windowed timer function. These instances support an additional feature that can detect watchdog timer resets that occur too early as well as too late (or never). This could happen if program execution is corrupted and is accidentally forced into a tight loop of code that contains a watchdog sequence. This would not be detected with a traditional WDT, because the end of the timeout periods would never be reached. A new set of "watchdog timer early" fields are available to support the lower limits required for windowing. Traditional watchdog timers can only detect a loss of program control that fails to reset the watchdog timer.

Each time the software performs a reset, as early as possible in the software, the peripheral control register should be examined to determine if the reset was cause by at WDT late reset event (or WDT early reset event if the window function is supported). If so, the software should take the desired action as part of its restart sequence.

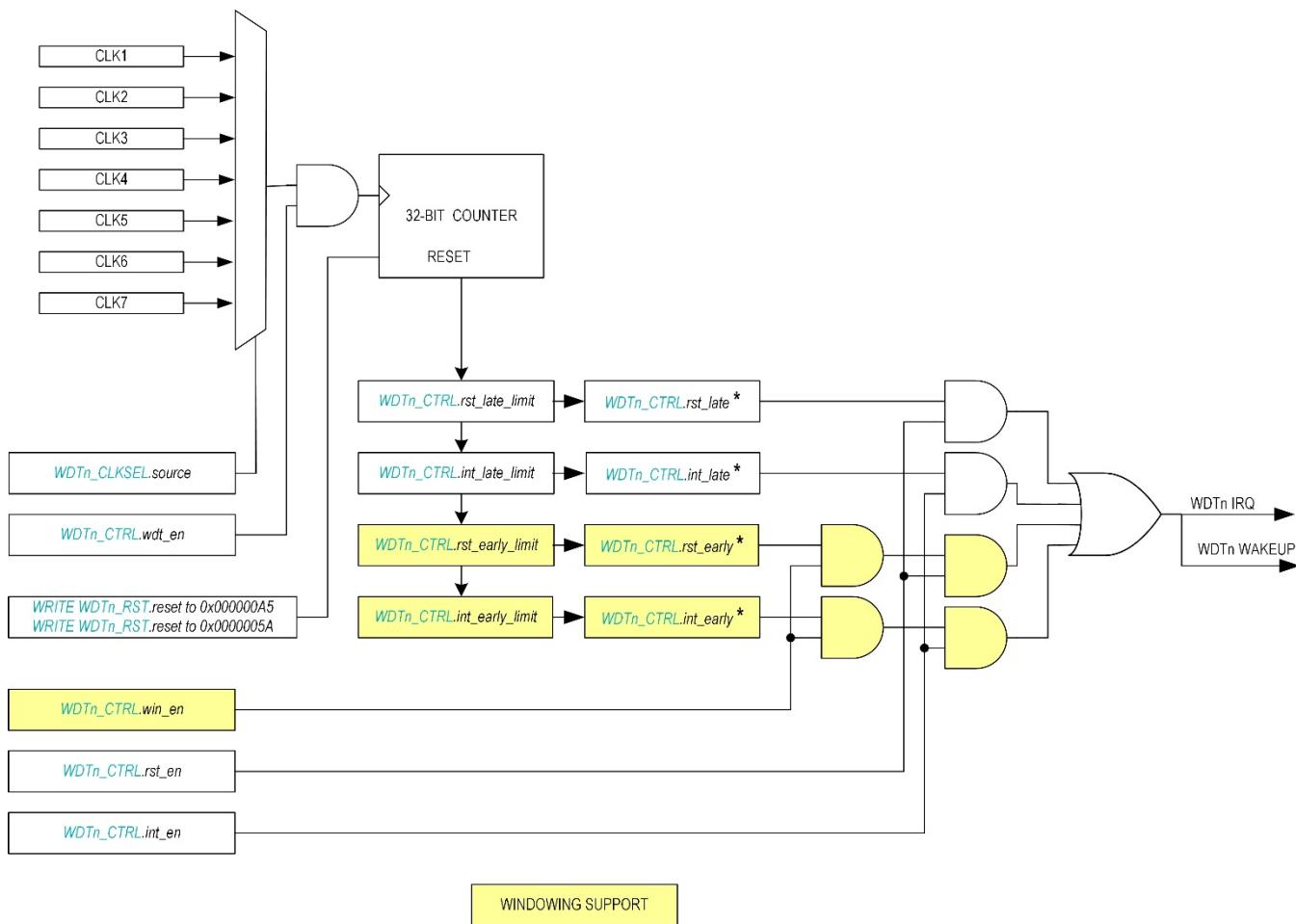
The WDT is a critical safety feature, and most fields are reset on POR or system reset events only.

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  $2^{1327}$  time-base ticks.
- New register to read the WDT counter register, simplifying code development.

*Figure 20-1* shows the block diagram of the WDT.

*Figure 20-1: Windowed Watchdog Timer Block Diagram*



\* INTERRUPT FLAGS ARE SET REGARDLESS OF .win\_en, AND .int\_en (FOR INTERRUPTS) and .rst\_en (FOR WDT RESETS).

## 20.1 Instances

*Table 20-1* shows the peripheral instances, available clock sources, and indicates which instances support the windowed watchdog functionality.

*Table 20-1: MAX78000 WDT Instances Summary*

Instance	Register Access Name	Window Support	External Clock	CLK1	CLK2	CLK3
WDTO	WDTO	Yes	N/A	PCLK	IBRO	-
LPWDTO	WDT1	Yes	N/A	IBRO	ERTCO	INRO

## 20.2 Usage

When enabled, *WDTn\_CNT.count* increments once every  $t_{WDTCLK}$  period. During correct operation, the WDT will periodically execute the feed sequence and reset *WDTn\_CNT.count* to 0x0000 0000 within the target window.

The upper and lower limits of the target window are user-configurable to accommodate different applications and non-deterministic execution times within an application.

The WDT can generate interrupt and/or reset events in response to the WDT activity. Interrupts are typically configured to respond first to an event outside the target window. The approach is that a minor system event may have temporarily delayed the execution of the feed sequence, so the event can be diagnosed in an interrupt routine and control returned to the system. When the WDT feed sequence occurs much earlier than expected or not at all, a reset event can be generated that 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 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 which support the window feature (`WDTn_CTRL.win_en = 1`) can generate a WDT early interrupt event if the WDT feed sequence occurs earlier than expected. In an analogous manner, the device 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 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. This includes the early interrupt and early event flags, even if the WDT is disabled (`WDTn_CTRL.win_en = 0`).

### 20.2.1 Using the WDT as a Long-Interval Timer

One use case 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 use case.

### 20.2.2 Using the WDT as a Long-Interval Wakeup Timer

The WDT can be used as a very long internal wakeup source from *SLEEP*. Additionally, the LPWDT can be used as a wakeup source from *SLEEP*, *LPM*, and *UPM*. The timer can be configured to generate a WDT wakeup event for as long as  $2^{32}$  periods of the selected watchdog clock source.

## 20.3 WDT Feed Sequence

The WDT feed sequence protects the system against unintentional altering of the WDT count and unintentional enabling or disabling of the timer itself.

Two consecutive write instructions to the `WDTn_RST.reset` field are required to reset the `WDTn_CNT.count = 0`. Global interrupts should be disabled immediately before, and reenabled after the writes, to ensure both writes to the `WDTn_RST.reset` field complete without interruption.

The feed sequence must also be performed immediately before enabling the WDT, to prevent an accidental triggering of the reset or interrupt as soon as the timer is enabled. There is no timed access window for these write operations; the operations can be separated by any length of time as long as they occur in the required sequence

1. Disable interrupts.
2. In consecutive write operations:
  - a. Write `WDTn_RST.reset`: 0x0000 00A5.
  - b. Write `WDTn_RST.reset`: 0x0000 005A.
3. If desired, enable or disable the timer.
4. Re-enable interrupts.

## 20.4 WDT Events

Multiple events are supported as shown in *Table 20-2*. The corresponding event flag is set when the event occurs.

Typically the system is configured such that the late interrupt events occur prior to the late reset events, and early interrupts occur when the feed sequence has the least error from the target time, prior to the early reset events.

The event flags will be set even if the corresponding interrupt or reset enable flags regardless of the state of the corresponding enable fields. This includes the early interrupt and early event flags, even if `WDTn_CTRL.win_en` = 0.

Software must clear the all event flags before enabling the timers.

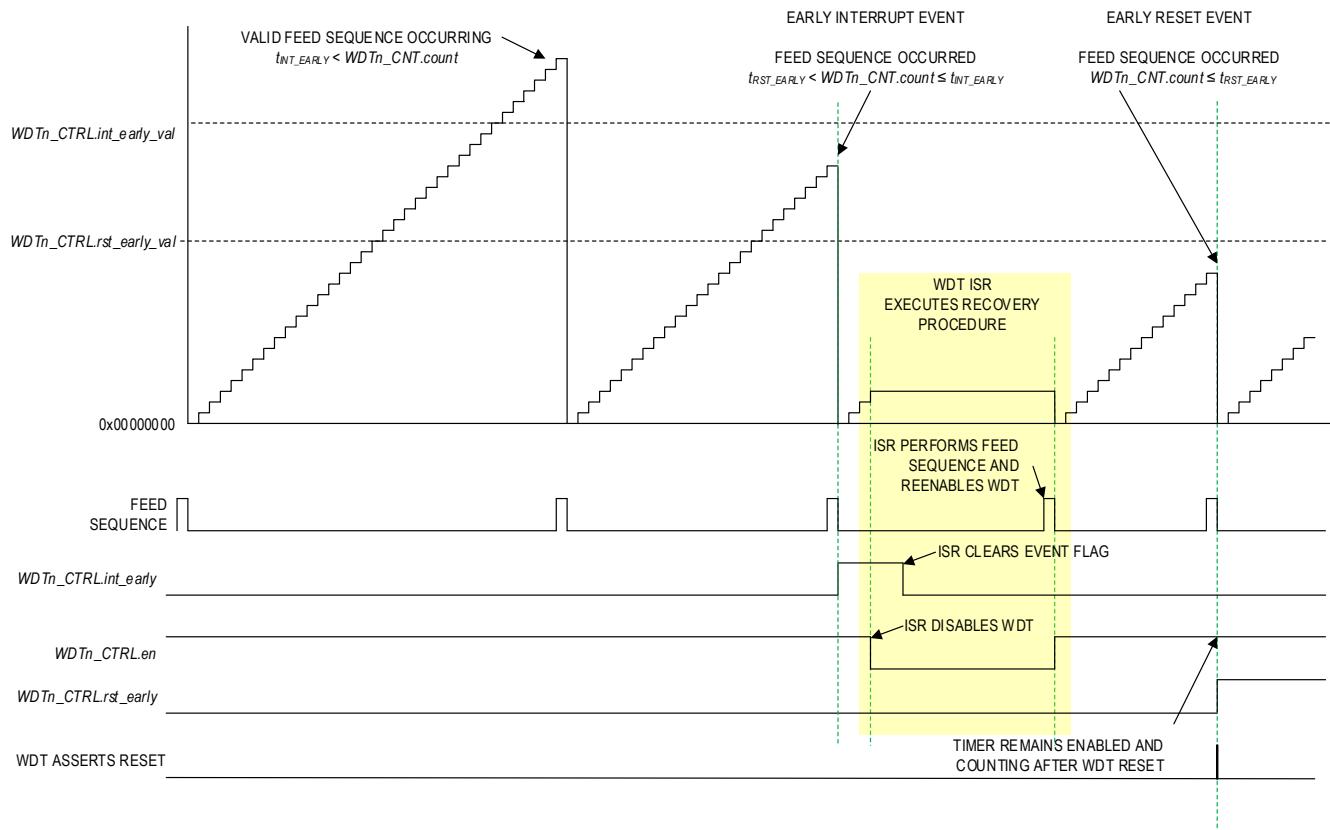
*Table 20-2: MAX78000 WDT Event Summary*

Event	Condition	Local Interrupt Event Flag	Local Interrupt Event Enable
Early Interrupt	Feed sequence occurs while <code>WDTn_CTRL.rst_early_val</code> ≤ <code>WDTn_CNT.count</code> < <code>WDTn_CTRL.int_early_val</code> <code>WDTn_CTRL.win_en</code> = 1	<code>WDTn_CTRL.int_early</code>	<code>WDTn_CTRL.wdt_int_en</code>
Early Reset	Feed sequence occurs while <code>WDTn_CNT.count</code> < <code>WDTn_CTRL.rst_early_val</code> <code>WDTn_CTRL.win_en</code> = 1	<code>WDTn_CTRL.rst_early</code>	<code>WDTn_CTRL.wdt_RST_en</code>
Interrupt Late	<code>WDTn_CNT.count</code> = <code>WDTn_CTRL.int_late_val</code>	<code>WDTn_CTRL.int_late</code>	<code>WDTn_CTRL.wdt_int_en</code>
Reset Late	<code>WDTn_CNT.count</code> = <code>WDTn_CTRL.rst_late_val</code>	<code>WDTn_CTRL.rst_late</code>	<code>WDTn_CTRL.wdt_RST_en</code>
Timer Enabled	<code>WDTn_CTRL.clkrdy</code> 0 → 1	No event flags are set by a timer enabled event	

### 20.4.1 WDT Early Reset

The early reset event occurs if the software performs the WDT feed sequence while `WDTn_CNT.count` < `WDTn_CTRL.rst_late_val` threshold as shown in *Table 20-2*. *Figure 20-2* shows the sequencing details associated with an early reset event.

*Figure 20-2: WDT Early Interrupt and Reset Event Sequencing Details*



The following occurs when a WDT early reset event occurs:

1. The hardware sets `WDtn_CTRL.rst_early` to 1.
2. The hardware initiates a system reset:
3. The hardware resets `WDtn_CNT.count` to 0x0000 0000 during the reset event.
4. The `WDtn_CTRL.en` field is unaffected by a system reset. The WDT continues incrementing.
5. The `WDtn_CTRL.rst_early` field is unaffected by a system reset, allowing the software to determine an early reset event occurred.

#### 20.4.2 WDT Early Interrupt

The early interrupt event occurs if software performs the WDT feed sequence while  $WDtn\_CTRL.rst\_early\_val \leq WDtn\_CNT.count < WDtn\_CTRL.int\_early\_val$  as shown in [Table 20-2](#). [Figure 20-2](#) shows the sequencing details associated with an early interrupt event, including the required functions performed by the WDT ISR.

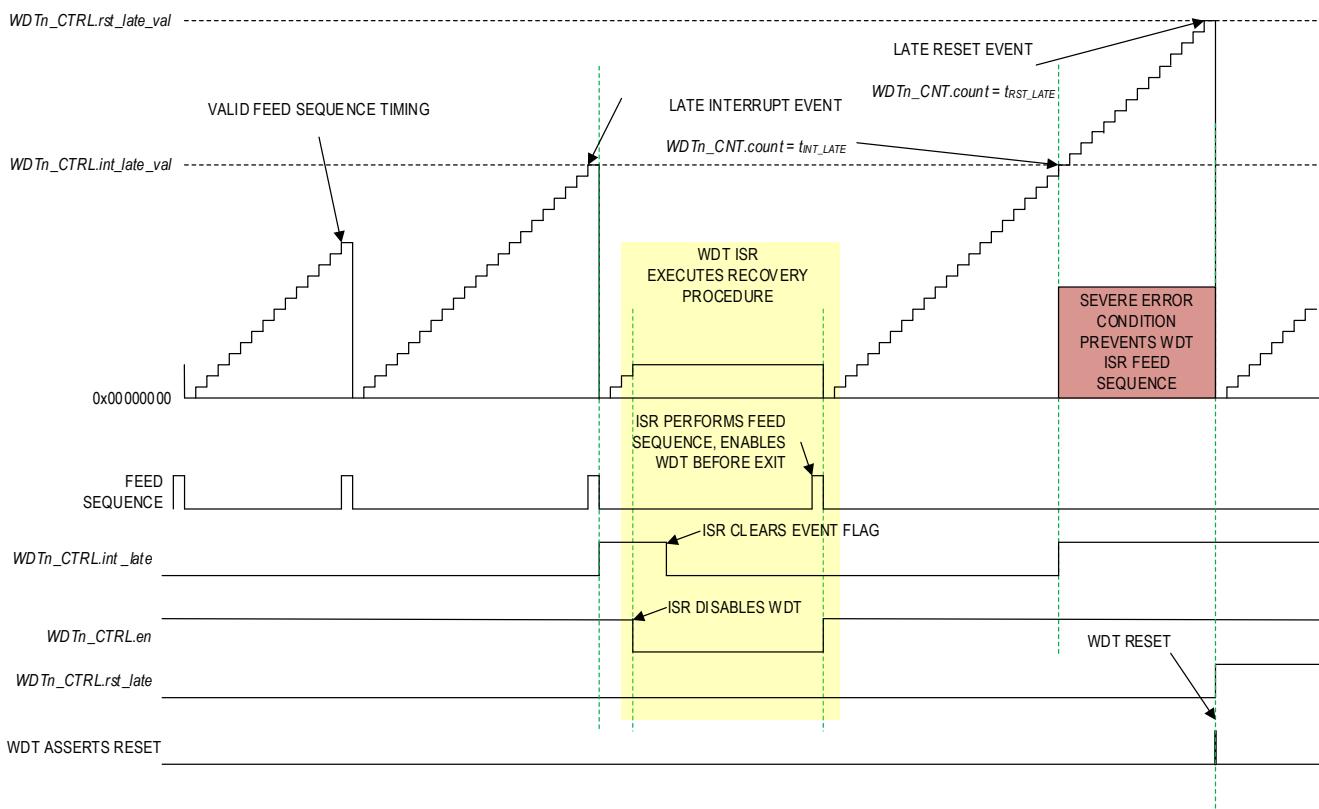
The hardware performs the following when a WDT early interrupt event occurs:

1. The `WDtn_CTRL.int_early` field is set to 1.
2. An interrupt occurs if enabled.

#### 20.4.3 WDT Late Reset

The late reset event occurs if the counter increments to the point where `WDtn_CNT.count` is equal to the `WDtn_CTRL.rst_late_val` threshold as shown in [Table 20-2](#). [Figure 20-3](#) shows the sequencing details associated with a late reset event.

*Figure 20-3: WDT Late Interrupt and Reset Event Sequencing Details*



The following occurs when a WDT late reset event occurs:

1. The hardware sets `WDn_CTRL.rst_late` to 1.
2. The hardware initiates a system reset:
3. The hardware resets `WDn_CNT.count` to 0x0000 0000 during the reset event.
4. The `WDn_CTRL.en` field is unaffected by a system reset. The WDT continues incrementing.
5. The `WDn_CTRL.rst_late` field is unaffected by a system reset.

#### 20.4.4 WDT Late Interrupt

The late reset event occurs if the counter increments to the point where `WDn_CNT.count = WDn_CTRL.int_late_val` threshold as shown in Table 20-2. *Figure 20-3* shows the sequencing details associated with a late interrupt event, including the required functions performed by the WDT ISR.

The following occurs when WDT late interrupt event occurs:

1. The hardware sets `WDn_CTRL.int_late` to 1.
2. The hardware initiates an interrupt if enabled.

## 20.5 Initializing the WDT

The full procedure for configuring the WDT is shown below:

1. Execute the WDT feed sequence and disable the WDT:
  - a. Disable global interrupts
  - b. Write `WDTn_RST.reset` to 0x0000 00A5.
  - c. Write `WDTn_RST.reset` to 0x0000 005A. Hardware will reset `WDTn_CNT.count` = 0x0000 0000.
  - d. 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, or
  - b. Set `WDTn_CTRL.clk_rdy_ie` = 1 to generate a WDT enabled interrupt event.
3. Re-enable global interrupts.
4. Configure `WDTn_CLKSEL.source` to select the clock source.
5. Configure the traditional/legacy thresholds:
  - a. Configure `WDTn_CTRL.int_late_val` 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.
6. 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.
  - d. Set `WDTn_CTRL.wdt_int_en` to generate an interrupt when a WDT late interrupt event occurs. If `WDTn_CTRL.win_en` = 1 an interrupt will be generated by both a WDT late interrupt event and also a WDT early interrupt event.
  - e. Set `WDTn_CTRL.wdt_rst_en` to generate an interrupt when a WDT late reset event occurs. If `WDTn_CTRL.win_en` = 1 an interrupt will be generated by a WDT late reset event and also a WDT early reset event.
7. Execute the WDT feed sequence and enable the WDT:
  - a. Disable global interrupts
  - b. Write `WDTn_RST.reset` to 0x0000 00A5.
  - c. Write `WDTn_RST.reset` to 0x0000 005A. Hardware will reset `WDTn_CNT.count` = 0x0000 0000.
  - d. Set `WDTn_CTRL.en` to 1 to enable the WDT.
8. Verify the peripheral is enabled before proceeding:
  - a. Poll `WDTn_CTRL.clkrdy` until it reads 1, or
  - b. Set `WDTn_CTRL.clkrdy_ie` = 1 to generate a WDT enabled event interrupt.
9. Re-enable global interrupts.

## 20.6 Resets

The WDT is a critical safety feature, and most of the fields are set on POR or system reset events only.

## 20.7 Registers

See [Table 2-4](#) for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in [Table 20-3](#). Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn\_CTRL resolves to PERIPHERAL0\_CTRL and PERIPHERAL1\_CTRL for instances 0 and 1, respectively.

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

*Table 20-3: WDT Register Summary*

Offset	Register	Name
[0x0000]	<i>WDTn_CTRL</i>	WDT Control Register
[0x0004]	<i>WDTn_RST</i>	WDT Reset Register
[0x0008]	<i>WDTn_CLKSEL</i>	WDT Clock Select Register
[0x000C]	<i>WDTn_CNT</i>	WDT Count Register

## 20.7.1 Register Details

*Table 20-4: WDT Control Register*

WDT Control				<i>WDTn_CTRL</i>	[0x0000]
Bits	Name	Access	Reset	Description	
31	rst_late	R/W	0	<b>Reset Late Event</b> A watchdog reset event occurred after the time specified in <i>WDTn_CTRL.rst_late_val</i> . This flag is set even if <i>WDTn_CTRL.win_en</i> = 0 or <i>WDTn_CTRL.wdt_RST_en</i> = 0. The software has to clear the flag appropriately in case of carried over flags from prior operations.  0: Watchdog did not cause reset event. 1: Watchdog reset occurred after <i>WDTn_CTRL.rst_early_val</i> .	
30	rst_early	R/W	0	<b>Reset Early Event</b> A watchdog reset event occurred before the time specified in <i>WDTn_CTRL.rst_early_val</i> . This flag is set even if <i>WDTn_CTRL.win_en</i> = 0 or <i>WDTn_CTRL.wdt_RST_en</i> = 0. The software has to clear the flag appropriately in case of carried over flags from prior operations.  0: Watchdog did not cause reset event. 1: Watchdog reset occurred before <i>WDTn_CTRL.rst_early_val</i> .	
29	win_en	R/W	0	<b>Window Function Enable</b> 0: Disabled. WDT recognizes interrupt late and reset late events, supporting legacy implementations. 1: Enabled	
28	clkrdy	R	0	<b>Clock Status</b> This field is cleared by the hardware when software changes the state of <i>WDTn_CTRL.en</i> . The hardware sets the field to 1 when the change to the requested enable/disable is complete.  0: WDT clock is off. 1: WDT clock is on.	
27	clkrdy_ie	R/W	0	<b>Clock Switch Ready Interrupt Enable</b> This interrupt prevents the software from needing to poll the <i>WDTn_CTRL.clkrdy</i> all the time. When <i>WDTn_CTRL.clkrdy</i> goes from high to low, the interrupt signals the transition is complete.  0: Disabled 1: Enabled	
26:24	-	RO	0	<b>Reserved</b>	

WDT Control				WDTn_CTRL	[0x0000]
Bits	Name	Access	Reset	Description	
23:20	rst_early_val	R/W	0	<b>Reset Early Event Threshold</b> 0x0: $2^{31} \times t_{WDTCLOCK}$ 0x1: $2^{30} \times t_{WDTCLOCK}$ 0x2: $2^{29} \times t_{WDTCLOCK}$ 0x3: $2^{28} \times t_{WDTCLOCK}$ 0x4: $2^{27} \times t_{WDTCLOCK}$ 0x5: $2^{26} \times t_{WDTCLOCK}$ 0x6: $2^{25} \times t_{WDTCLOCK}$ 0x7: $2^{24} \times t_{WDTCLOCK}$ 0x8: $2^{23} \times t_{WDTCLOCK}$ 0x9: $2^{22} \times t_{WDTCLOCK}$ 0xA: $2^{21} \times t_{WDTCLOCK}$ 0xB: $2^{20} \times t_{WDTCLOCK}$ 0xC: $2^{19} \times t_{WDTCLOCK}$ 0xD: $2^{18} \times t_{WDTCLOCK}$ 0xE: $2^{17} \times t_{WDTCLOCK}$ 0xF: $2^{16} \times t_{WDTCLOCK}$	
19:16	int_early_val	R/W	0	<b>Interrupt Early Event Threshold</b> 0x0: $2^{31} \times t_{WDTCLOCK}$ 0x1: $2^{30} \times t_{WDTCLOCK}$ 0x2: $2^{29} \times t_{WDTCLOCK}$ 0x3: $2^{28} \times t_{WDTCLOCK}$ 0x4: $2^{27} \times t_{WDTCLOCK}$ 0x5: $2^{26} \times t_{WDTCLOCK}$ 0x6: $2^{25} \times t_{WDTCLOCK}$ 0x7: $2^{24} \times t_{WDTCLOCK}$ 0x8: $2^{23} \times t_{WDTCLOCK}$ 0x9: $2^{22} \times t_{WDTCLOCK}$ 0xA: $2^{21} \times t_{WDTCLOCK}$ 0xB: $2^{20} \times t_{WDTCLOCK}$ 0xC: $2^{19} \times t_{WDTCLOCK}$ 0xD: $2^{18} \times t_{WDTCLOCK}$ 0xE: $2^{17} \times t_{WDTCLOCK}$ 0xF: $2^{16} \times t_{WDTCLOCK}$	
15:13	-	RO	0	<b>Reserved</b>	
12	int_early	R/W	0	<b>Interrupt Early Flag</b> A feed sequence occurred earlier than the time determined by the <a href="#">WDTn_CTRL.int_early_val</a> field. This flag will be set even if <a href="#">WDTn_CTRL.win_en</a> = 0. 0: No interrupt event 1: Interrupt event occurred. Generates a WDT interrupt if <a href="#">WDTn_CTRL.wdt_int_en</a> = 1.	
11	wdt_RST_en	R/W	0	<b>WDT Reset Enable</b> 0: Disabled 1: Enabled	
10	wdt_INT_en	R/W	0	<b>WDT Interrupt Enable</b> 0: Disabled 1: Enabled	

WDT Control				WDTn_CTRL	[0x0000]
Bits	Name	Access	Reset	Description	
9	int_late	R/W	0	<b>Interrupt Late Flag</b> A watchdog feed sequence did not occur before the time determined by the <a href="#">WDTn_CTRL.int_late_val</a> field. 0: No interrupt event. 1: Interrupt event occurred. <i>Note: A WDT interrupt is generated if the WDT interrupt is enabled (<a href="#">WDTn_CTRL.wdt_int_en</a> = 1).</i>	
8	en	R/W	0	<b>WDT Enable</b> This field enables/disables the WDT peripheral clock. <a href="#">WDTn_CTRL.count</a> holds its value while the WDT is disabled. The WDT feed sequence must be performed immediately prior to any change to this field. 0: Disabled 1: Enabled	
7:4	rst_late_val	R/W	0	<b>Reset Late Event Threshold</b> 0x0: $2^{31} \times t_{WDTCLOCK}$ 0x1: $2^{30} \times t_{WDTCLOCK}$ 0x2: $2^{29} \times t_{WDTCLOCK}$ 0x3: $2^{28} \times t_{WDTCLOCK}$ 0x4: $2^{27} \times t_{WDTCLOCK}$ 0x5: $2^{26} \times t_{WDTCLOCK}$ 0x6: $2^{25} \times t_{WDTCLOCK}$ 0x7: $2^{24} \times t_{WDTCLOCK}$ 0x8: $2^{23} \times t_{WDTCLOCK}$ 0x9: $2^{22} \times t_{WDTCLOCK}$ 0xA: $2^{21} \times t_{WDTCLOCK}$ 0xB: $2^{20} \times t_{WDTCLOCK}$ 0xC: $2^{19} \times t_{WDTCLOCK}$ 0xD: $2^{18} \times t_{WDTCLOCK}$ 0xE: $2^{17} \times t_{WDTCLOCK}$ 0xF: $2^{16} \times t_{WDTCLOCK}$	
3:0	int_late_val	R/W	0	<b>Interrupt Late Event Threshold</b> 0x0: $2^{31} \times t_{WDTCLOCK}$ 0x1: $2^{30} \times t_{WDTCLOCK}$ 0x2: $2^{29} \times t_{WDTCLOCK}$ 0x3: $2^{28} \times t_{WDTCLOCK}$ 0x4: $2^{27} \times t_{WDTCLOCK}$ 0x5: $2^{26} \times t_{WDTCLOCK}$ 0x6: $2^{25} \times t_{WDTCLOCK}$ 0x7: $2^{24} \times t_{WDTCLOCK}$ 0x8: $2^{23} \times t_{WDTCLOCK}$ 0x9: $2^{22} \times t_{WDTCLOCK}$ 0xA: $2^{21} \times t_{WDTCLOCK}$ 0xB: $2^{20} \times t_{WDTCLOCK}$ 0xC: $2^{19} \times t_{WDTCLOCK}$ 0xD: $2^{18} \times t_{WDTCLOCK}$ 0xE: $2^{17} \times t_{WDTCLOCK}$ 0xF: $2^{16} \times t_{WDTCLOCK}$	

Table 20-5: WDT Reset Register

WDT Reset				WDTn_RST	[0x0004]
Bits	Name	Access	Reset	Description	
31:8	-	RO	0	<b>Reserved</b>	
7:0	reset	R/W	0 <sup>†</sup>	<b>Reset Watchdog Timer Count</b> Writing the WDT feed sequence, in two consecutive write instructions, to this register resets the internal counter to 0x0000 0000. Perform the following steps to perform a WDT feed sequence: 1. Write <a href="#">WDTn_RST.reset</a> = 0x0000 00A5 2. Write <a href="#">WDTn_RST.reset</a> = 0x0000 005A Writes to the <a href="#">WDTn_CTRL.en</a> field, which enable or disable the WDT, must be the next instruction following the WDT feed sequence. <sup>†</sup> <i>Note: This field is set to 0 on a POR and is not affected by any other for of reset.</i>	

Table 20-6: WDT Clock Source Select Register

WDT Clock Source Select				WDTn_CLKSEL	[0x0008]
Bits	Name	Access	Reset	Description	
31:3	-	RO	0	<b>Reserved</b>	
2:0	source	R/W	0 <sup>†</sup>	<b>Clock Source Select</b> See <a href="#">Table 20-1</a> for available clock options. 0: CLK0 1: CLK1 2: CLK2 3: CLK3 4: CLK4 5: CLK5 6: CLK6 7: CLK7 <sup>†</sup> <i>Note: This field is only reset on a POR and unaffected by other resets.</i>	

Table 20-7: WDT Count Register

WDT Count				WDTn_CNT	[0x000C]
Bits	Name	Access	Reset	Description	
31:0	count	R	0	<b>WDT Counter</b> The counter value for debug. This register is reset by a system reset, as well as the watchdog feeding sequence. When the WDT clock is off, the feeding sequence generates an asynchronous reset of 1 PCLK width. When the WDT clock is on, the feeding sequence generates a synchronous reset that is a handshake to the WDT clock domain.	

## 21. Pulse Train Engine (PT)

Each independent pulse train engine operates either in square wave mode, which generates a continuous 50% duty-cycle square wave, or pulse train mode, which generates a continuous programmed bit pattern from 2- to 32-bits in length. Pulse train engines are used independently or may be synchronized together to generate signals in unison. The frequency of each generated output can be set separately based on a divisor of the Peripheral Clock.

### 21.1 Instances

The device provides four instances of the pulse train engine peripheral.

- PTO to PT3

All peripheral registers share a common register set.

### 21.2 Features

The pulse train outputs with individually programmable modes, patterns and output enables. The pulse train engine uses the PCLK,  $f_{PTE\_CLK} = f_{PCLK}$ , 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.

### 21.3 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.

#### 21.3.1 Pulse Train Output Modes

Each pulse train output supports the following modes:

- Pulse train mode
- Bit pattern length
- Square wave mode

##### 21.3.1.1 Pulse Train Mode

When pulse train  $n$  ( $PTn$ ) is configured in pulse train mode, the configuration also includes the bit length (up to 32-bits) of the custom pulse train. This is configured using the 5-bit field  $PTn\_RATE\_LENGTH.mode$  as follows:

*PTn\_RATE\_LENGTH.mode* = 1:

*PTn* configured in square wave mode.

*PTn\_RATE\_LENGTH.mode* > 1:

*PTn* 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).

### 21.3.1.2 In Pulse Train Mode, Set the Bit Pattern

If an output is set to pulse train mode, then configure a custom bit pattern from 2- 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 21-1: Pulse Train Mode Output Function*

$$PTn\_TRAIN = [\text{Bit pattern for } PTn]$$

### 21.3.1.3 Synchronize Two or More Outputs, if Needed

The write-only register *PTG\_RESET* “PT Global Resync” allows two or more outputs to be reset and synchronized. Write to any bit in *PTG\_RESET* to simultaneously reset any outputs in pulse train mode to the beginning of the pattern (the LSB) set in the *PTn\_TRAIN* bit-pattern register, and reset the output to 0 for outputs in square wave mode.

### 21.3.1.4 Pulse Train Loop Mode

By default, a pulse train engine runs indefinitely until it is disabled by the software.

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.

### 21.3.1.5 Pulse Train Loop Delay

If the pulse train is configured in loop mode, a delay can be inserted after each repeated output pattern. To enable a delay, 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. During this delay, the output is held at the MSB of the last pattern. If the loop counter has not reached 0, then it is decremented when the next pattern starts.

### 21.3.1.6 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 in square wave mode.

If a running pulse train engine is triggered by another pulse train’s stop event, Automatic restart restarts the running pulse train engine from the beginning of its pattern. If a pulse train engine is triggered by another pulse train’s stop event, and it is not running, automatic restart sets the enable bit to 1, and starts the pulse train engine.

The settings for this mode are contained in the *PTn\_RESTART* register for each pulse train engine.

*Note: 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 PT8\_RESTART register configures which pulse train engine triggers PT8 to restart.*

Each pulse train engine can be configured to perform an automatic restart when it detects a stop event from one or two pulse trains.

If `PTn_RESTART.on_pt_x_loop_exit = 1`, then pulse train engine  $n$  automatically restarts when it detects a stop event from pulse train  $x$ , where  $x$  is the value in the 5-bit field `PTn_RESTART.pt_x_select`.

If `PTn_RESTART.on_pt_y_loop_exit = 1`, then pulse train engine  $n$  automatically restarts when it detects a stop event from pulse train  $y$ , where  $y$  is the value in 5-bit field `PTn_RESTART.pt_y_select`.

A pulse train engine can be configured to restart on its own stop event, allowing the pulse train to run indefinitely.

Each individual pulse train can be configured for:

- No automatic restart.
- Automatic restart triggered by a stop event from pulse train  $x$  only.
- Automatic restart triggered by a stop event from pulse train  $y$  only.
- Automatic restart triggered by a stop event from both pulse train  $x$  and pulse train  $y$

## 21.4 Enabling and Disabling a Pulse Train Output

The `PTG_ENABLE` register is used to enable and disable each of the individual pulse train outputs. Enable a given pulse train output by setting the respective bit in the `PTG_ENABLE` register. Halt a pulse train output by clearing the respective bit in the `PTG_ENABLE` register.

*Note: Prior to changing a pulse train output's configuration the corresponding pulse train output should be halted to prevent unexpected behavior.*

## 21.5 Atomic Pulse Train Output Enable and Disable

Deterministic enable and disable operations are critical for pulse train outputs that must be synchronized in an application. The `PTG_ENABLE` register does not perform atomic access directly. Atomic operations are supported using the registers `PTG_SAFE_EN`, `PTG_SAFE_DIS`.

For most pulse train peripherals, enabling and disabling individual pulse trains is performed by setting and clearing bits in the global enable/disable register, which for this peripheral is `PTG_ENABLE`. For most Arm Cortex-M microcontrollers, this is usually done by bit banding. Because bit banding performs a read, modify, write (RMW), some pulse trains could start and end during the RMW operation, often with unpredictable results.

To ensure safe and predictable operation, two additional registers are used to enable and disable the outputs.

### 21.5.1 Pulse Train Atomic Enable

`PTG_SAFE_EN` “Global Safe Enable” is a write-only register. To safely enable outputs without a RMW, write a 32-bit value to this register with a 1 in the bit positions corresponding to the pulse train engines to be enabled. This immediately sets to 1 the corresponding bits in the `PTG_ENABLE` register to 1, which enables the corresponding pulse train engine. Writing a 0 to any bit position in the `PTG_SAFE_EN` register has no effect on the state of the corresponding pulse train enable bit. If the corresponding pulse train engine is already enabled and running, writing a 1 to that bit position in the `PTG_SAFE_EN` register has no effect.

### 21.5.2 Pulse Train Atomic Disable

`PTG_SAFE_DIS` “Global Safe Disable” is a write-only register for disabling a pulse train engine without performing a RMW. To safely disable pulse train engines, write a 32-bit value to this register with a 1 in the bit positions corresponding to the pulse train engines to be disabled. This immediately clears to 0 the corresponding bits in `PTG_ENABLE` which disables the corresponding pulse train engines. Writing a 0 to any bit position in the `PTG_SAFE_DIS` register has no effect on the state of the corresponding pulse train enable bit.

Bit banding is not supported for the *PTG\_ENABLE*, *PTG\_SAFE\_EN*, and *PTG\_SAFE\_DIS* registers and can have unpredictable results.

## 21.6 Halt and Disable

Once a pulse train engine is enabled and running, it continues to run until one of the following events stops the output:

- The corresponding enable bit in the *PTG\_ENABLE* register is cleared to 0 to halt the output.
- A 1 is written to the corresponding disable bit in the *PTG\_SAFE\_DIS* register to halt the output.
- The corresponding resync bit in the *PTG\_RESYNC* register is cleared to 0 to halt and reset the output.
- *PTn\_LOOP* was initialized to a non-zero value, and the loop count has reached 0 (this has no effect in square wave mode; it only applies to pulse train mode).

When a pulse train is halted, the corresponding enable bit in *PTG\_ENABLE* is automatically cleared to 0.

## 21.7 Interrupts

Each pulse train can generate an interrupt only if it is configured in pulse train mode, and the loop counter *PTG\_SAFE\_DIS* was initialized to a non-zero number. When *PTG\_SAFE\_DIS* counts down to 0, the corresponding status flag in the *PTG\_INTFL* register is set. If the corresponding interrupt enable bit in the *PTG\_INTEN* register is set, the event also generates an interrupt.

## 21.8 Registers

See *Table 2-4* for the base address of this peripheral/module. If multiple instances of the peripheral are provided, each instance has its own, independent set of the registers shown in *Table 21-1*. Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register *PERIPHERALn\_CTRL* resolves to *PERIPHERAL0\_CTRL* and *PERIPHERAL1\_CTRL* for instances 0 and 1, respectively.

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

*Table 21-1: Pulse Train Engine Register Summary*

Offset	Register	Description
[0x0000]	<i>PTG_ENABLE</i>	PT Global Enable/Disable Control
[0x0004]	<i>PTG_RESYNC</i>	PT Global Resync
[0x0008]	<i>PTG_INTFL</i>	PT Stopped Global Status Flags
[0x000C]	<i>PTG_INTEN</i>	PT Global Interrupt Enable
[0x0010]	<i>PTG_SAFE_EN</i>	PT Global Safe Enable
[0x0014]	<i>PTG_SAFE_DIS</i>	PT Global Safe Disable
[0x0020]	<i>PTn_RATE_LENGTH</i>	PTn Configuration
[0x0024]	<i>PTn_TRAIN</i>	PTn Pulse Train Mode Bit Pattern
[0x0028]	<i>PTn_LOOP</i>	PTn Loop Control
[0x002C]	<i>PTn_RESTART</i>	PTn Automatic Restart
[0x0030]	<i>PTn_RATE_LENGTH</i>	PTn Configuration
[0x0034]	<i>PTn_TRAIN</i>	PT1 Pulse Train Mode Bit Pattern
[0x0038]	<i>PTn_LOOP</i>	PT1 Loop Control
[0x003C]	<i>PTn_RESTART</i>	PT1 Automatic Restart
[0x0040]	<i>PTn_RATE_LENGTH</i>	PT2 Configuration
[0x0044]	<i>PTn_TRAIN</i>	PT2 Pulse Train Mode Bit Pattern
[0x0048]	<i>PTn_LOOP</i>	PT2 Loop Control
[0x004C]	<i>PTn_RESTART</i>	PT2 Automatic Restart

Offset	Register	Description
[0x0050]	<i>PTn_RATE_LENGTH</i>	PT3 Configuration
[0x0054]	<i>PTn_TRAIN</i>	PT3 Pulse Train Mode Bit Pattern
[0x0058]	<i>PTn_LOOP</i>	PT3 Loop Control
[0x005C]	<i>PTn_RESTART</i>	PT3 Automatic Restart

## 21.8.1 Register Details

### 21.8.1.1 Pulse Train Engine Global Enable/Disable Register

This register enables each of the individual pulse trains. Write a 1 to the induthroughl 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 21-2: 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	<b>Enable PT3</b> 0: Disabled 1: Enabled <i>Note: Disabling an active pulse train halts the output and does not generate a stop event.</i>		
2	pt2	R/W	0	<b>Enable PT2</b> 0: Disabled 1: Enabled <i>Note: Disabling an active pulse train halts the output and does not generate a stop event.</i>		
1	pt1	R/W	0	<b>Enable PT1</b> 0: Disabled 1: Enabled <i>Note: Disabling an active pulse train halts the output and does not generate a stop event.</i>		
0	pt0	R/W	0	<b>Enable PTO</b> 0: Disabled 1: Enabled <i>Note: Disabling an active pulse train halts the output and does not generate a stop event.</i>		

Table 21-3: 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	-	<b>Resync Control for PT3</b> Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For square wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i>	
2	pt2	WO	-	<b>Resync Control for PT2</b> Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For square wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i>	
1	pt1	WO	-	<b>Resync Control for PT1</b> Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For square wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i>	
0	pt0	WO	-	<b>Resync Control for PTO</b> Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For square wave mode the output is reset to 0. Setting multiple bits simultaneously in this register synchronizes the set outputs. 1: Reset/restart the pulse train 0: No effect <i>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</i>	

Table 21-4: Pulse Train Engine Stopped Interrupt Flag Register

PT Stopped Interrupt Flag Register			PTG_INTFL		[0x0008]
Bits	Field	Access	Reset	Description	
31:4	-	RO	0	<b>Reserved</b>	
3	pt3	R/W1C	0	<b>PT3 Stopped Status Flag</b> 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	<b>PT2 Stopped Status Flag</b> 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	<b>PT1 Stopped Status Flag</b> 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	<b>PT0 Stopped Status Flag</b> This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In square wave mode, this field is not used. Write 1 to clear. 1: Pulse Train is stopped.	

Table 21-5: Pulse Train Engine Interrupt Enable Register

PT Interrupt Enable Register				PTG_INTEN	[0x000C]
Bits	Field	Access	Reset	Description	
31:4	-	RO	0	<b>Reserved</b>	
3	pt3	R/W	0	<b>PT3 Interrupt Enable</b> Write 1 to enable the interrupt for the corresponding PT when the flag is set in the <a href="#">PTG_INTFL</a> register. 0: Disabled. 1: Enabled.	
2	pt2	R/W	0	<b>PT2 Interrupt Enable</b> Write 1 to enable the interrupt for the corresponding PT when the flag is set in the <a href="#">PTG_INTFL</a> register. 0: Disabled. 1: Enabled.	
1	pt1	R/W	0	<b>PT1 Interrupt Enable</b> Write 1 to enable the interrupt for the corresponding PT when the flag is set in the <a href="#">PTG_INTFL</a> register. 0: Disabled. 1: Enabled.	
0	pt0	R/W	0	<b>PT0 Interrupt Enable</b> Write 1 to enable the interrupt for the corresponding PT when the flag is set in the <a href="#">PTG_INTFL</a> register. 0: Disabled. 1: Enabled.	

### 21.8.1.2 Pulse Train Engine Safe Enable Register

A 32-bit value written to this register performs an immediate binary OR with the contents of [PTG\\_ENABLE](#). The result is immediately stored in the [PTG\\_ENABLE](#).

*Table 21-6: 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	<b>Reserved</b>	
3	pt3	WO	-	<b>Safe Enable Control for PT3</b> Writing a 1 sets <a href="#">PTG_ENABLE.enable_pt3</a> . 1: Enable corresponding pulse train 0: No effect	
2	pt2	WO	-	<b>Safe Enable Control for PT2</b> Writing a 1 sets <a href="#">PTG_ENABLE.enable_pt2</a> . 1: Enable corresponding pulse train 0: No effect	
1	pt1	WO	-	<b>Safe Enable Control for PT1</b> Writing a 1 sets <a href="#">PTG_ENABLE.enable_pt1</a> . 1: Enable corresponding pulse train 0: No effect	
0	pt0	WO	-	<b>Safe Enable Control for PTO</b> Writing a 1 sets <a href="#">PTG_ENABLE.enable_pt0</a> . 1: Enable corresponding pulse train 0: No effect	

### 21.8.1.3 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 21-7: 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	<b>Reserved</b>	
3	pt3	WO	-	<b>Safe Disable Control for PT3</b> Writing a 1 clears <a href="#">PTG_ENABLE.enable_pt3</a> . 1: Disable corresponding pulse train 0: No effect	
2	pt2	WO	-	<b>Safe Disable Control for PT2</b> Writing a 1 clears <a href="#">PTG_ENABLE.enable_pt2</a> . 1: Disable corresponding pulse train 0: No effect	
1	pt1	WO	-	<b>Safe Disable Control for PT1</b> Writing a 1 clears <a href="#">PTG_ENABLE.enable_pt1</a> . 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	-	<b>Safe Disable Control for PT0</b> Writing a 1 clears <a href="#">PTG_ENABLE.enable_pt0</a> . 1: Disable corresponding pulse train 0: No effect	

#### 21.8.1.4 Pulse Train Registers

Table 21-8: Pulse Train Engine Configuration Register

Pulse Train <i>n</i> Configuration Register				PTn_RATE_LENGTH	[0x0020]
Bits	Field	Access	Reset	Description	
31:27	mode	R/W	1	<b>Square Wave or Pulse Train Output Mode</b> Sets either pulse train mode with length, or square wave mode. 0: Pulse train mode, 32-bits long 1: Square wave mode 2: Pulse train mode, 2-bits long 3: Pulse train mode, 3-bits long ...: 31: Pulse train mode, 31-bits long <i>Note: If this field is set to 1, square wave mode, the PTn_TRAIN register is not used.</i>	
26:0	rate_control	R/W	0	<b>Pulse Train Enable and Rate Control</b> Defines the rate at which the output for PTn changes state by setting the divisor of the PT Clock. Setting this field to 0 disables the PTn. For all other values, the following equation is used to calculate the rate.: $f_{PTn} = \frac{f_{PTE\_CLK}}{2^{rate\_control} - 1}$ 0: Output halted 1: $f_{PTn} = f_{PTE\_CLK}$ 2: $f_{PTn} = f_{PTE\_CLK}/2$ 3: $f_{PTn} = f_{PTE\_CLK}/4$ ...	

Table 21-9: Pulse Train Mode Bit Pattern Register

Pulse Train Mode Bit Pattern				PTn_TRAIN	[0x0024]
Bits	Field	Access	Reset	Description	
31:0	ptn_train	R/W	0	<b>Pulse Train Mode Bit Pattern</b> Write the repeating bit pattern that is shifted out, LSB first, when configured in pulse train mode. Set the bit pattern length with the <a href="#">PTn_RATE_LENGTH.mode</a> field. <i>Note: This register is ignored in square wave mode.</i> <i>Note: 0x0000_0000 and 0x0001 0000 are invalid values for this register.</i>	

**Table 21-10: Pulse Train n Loop Configuration Register**

Pulse Train Loop Configuration			PTn_LOOP		[0x0028]
Bits	Field	Access	Reset	Description	
31:28	-	RO	0	<b>Reserved</b>	
27:16	delay	R/W	0	<b>Pulse Train Delay Between Loops</b> Sets the delay, in number of Peripheral Clock cycles, that the output pauses between loops. The <i>PTn_LOOP.count</i> is decremented after the delay. <i>Note: This field is ignored if firmware writes 0 to PTn_LOOP.count field.</i>	
15:0	count	R/W	0	<b>Pulse Train Loop Countdown</b> 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 <i>PTn_INTFL</i> flag is set. Writing this field to 0 to repeat the pulse train pattern indefinitely. <i>Note: This field is ignored in square wave mode.</i>	

**Table 21-11: Pulse Train n Automatic Restart Configuration Register**

Pulse Train Automatic Restart Configuration			PTn_RESTART		[0x002C]
Bits	Field	Access	Reset	Description	
31:16	-	RO	0	<b>Reserved</b>	
15	on_pt_y_loop_exit	R/W	0	<b>Enable Automatic Restart for This Pulse Train on PTy Stop Event</b> 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	<b>Reserved</b>	
12:8	pt_y_select	R/W	0	<b>Select PTy</b> Select the pulse train number to be associated with PTy. This engine must be in pulse train mode. 0b00000: PT0 0b00001: PT1 0b00010: PT2 0b00011: PT3 0b00100 - 0b11111: Reserved	
7	on_pt_x_loop_exit	R/W	0	<b>Enable Automatic Restart for this Pulse Train on a PTn Stop Event</b> 0: Disable automatic restart 1: When PTn has a Stop Event, automatically restart the pulse train from the beginning of its pattern.	
6:5	-	RO	0	<b>Reserved</b>	
4:0	pt_x_select	R/W	0	<b>Select PTn</b> Select the pulse train number to be associated with PTn. This engine must be in pulse train mode. 0b00000: PT0 0b00001: PT1 0b00010: PT2 0b00011: PT3 0b00100-0b1xxxx: Reserved	



## 22. Cyclic Redundancy Check (CRC)

The CRC engine performs CRC calculations on data stored in RAM. The CRC engine cannot directly perform a CRC of data stored in flash memory.

The features include:

- User-definable polynomials up to  $x^{32}$  (33 terms).
- DMA support.
- Optional automatic byte swap of data input and calculated output.
- Most-significant bit or least-significant bit data input and calculated output.

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.
- 99.9999998% for a 32-bit CRC.

### 22.1 Instances

Instances of the peripheral are listed in *Table 22-1*.

Table 22-1: MAX78000 CRC Instances

Instance	Maximum Terms	DMA Support	Big- and Little Endian
CRC	33 ( $2^{32}$ )	Yes	Yes

### 22.2 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 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 `CRC_DATAIN32.data` field in either 8-bit, 16-bit or 32-bit wide data. The hardware automatically sets the `CRC_CTRL.busy` field to 1 while the CRC engine is calculating a CRC over the input data; the software must not access any of the CRC registers until the `CRC_CTRL.busy` field reads 0. The CRC value, `CRC_VAL.value`, is updated by the hardware after each write of data to the `CRC_DATAIN32.data` field. 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, this means 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 backwards. They reverse the polynomial and do right shifts on the data. The resulting CRC ends up being bit swapped and in the correct format.

By default, the CRC is calculated using the LSB first (`CRC_CTRL.msb` = 0.) When calculating the CRC using MSB first data, 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 ( $x^0 \dots x^{32}$ ).

Table 22-2: Organization of Calculated Result in the [CRC\\_VAL.value](#) field

<a href="#">CRC_CTRL.msb</a>	<a href="#">CRC_CTRL.byte_swap_out</a>	Order
0	0	The CRC value returned is the raw value
1	0	The CRC value returned is mirrored, but not byte swapped
0	1	The CRC value returned is byte swapped, but not mirrored
1	1	The CRC value returned is mirrored and then byte swapped

By default, the CRC engine calculates the CRC on the LSB of the data first.

The CRC can be calculated on the MSB of the data first by setting the [CRC\\_CTRL.msb](#) field to 1. 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.

## 22.3 Polynomial Generation

The CRC can be configured for any polynomial up to  $x^{32}$  (33 terms) by writing to the [CRC\\_POLY.poly](#) field. [Table 22-3](#) 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 22-3: 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

## 22.4 Calculations Using Software Writes to the FIFO

The software can perform CRC calculations by writing directly to the [CRC\\_DATAIN32.data](#) field. Each write to the [CRC\\_DATAIN32.data](#) field triggers the hardware to compute the CRC value. The software is responsible for loading all data for the CRC into the [CRC\\_DATAIN32.data](#) field. When complete, the result is read from the [CRC\\_VAL.value](#) field.

Use the following procedure to calculate a CRC value by writing to the FIFO:

1. Disable the CRC peripheral by setting the field `CRC_CTRL.en` to 0.
2. Configure the following fields for the desired input and output data formats:
  - a. `CRC_CTRL.msb`
  - b. `CRC_CTRL.byte_swap_in`
  - c. `CRC_CTRL.byte_swap_out`
3. Configure the CRC polynomial by writing to the `CRC_POLY.poly` field.
4. Set the CRC initial value by writing to the `CRC_VAL.value` field.
5. Set the `CRC_CTRL.en` field to 1 to enable the peripheral.
6. Write a value to be processed to `CRC_DATAIN32.data`; the hardware automatically sets the `CRC_CTRL.busy` field to 1 while performing the CRC.
7. At the end of the calculation, the hardware automatically:
  - a. Clears the `CRC_CTRL.busy` field to 0.
  - b. Loads the computed CRC value into the `CRC_VAL.value` field.
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.

## 22.5 Calculations Using DMA

The CRC engine requests new data from the DMA controller while the fields `CRC_CTRL.en` and `CRC_CTRL.dma_en` are both set to 1. Enable the corresponding DMA interrupt to receive an interrupt event when the CRC engine completes the CRC of the input data.

Use the following procedure to calculate a CRC value using the DMA:

1. Set `CRC_CTRL.en` = 0 to disable the peripheral.
2. Configure the DMA:
  - a. Set `CRC_CTRL.dma_en` = 1 to enable DMA mode.
3. Configure the peripheral for the required input and output data formats:
  - a. `CRC_CTRL.msb`
  - b. `CRC_CTRL.byte_swap_in`
  - c. `CRC_CTRL.byte_swap_out`
4. Configure the CRC polynomial by writing to the `CRC_POLY.poly` field.
5. Set the CRC initial value by writing to the `CRC_VAL.value` field.
6. Set the `CRC_CTRL.en` field to 1 to enable the peripheral:
  - a. The hardware automatically clears the `CRC_CTRL.busy` field to 0.
  - b. The DMA loads data into the `CRC_DATAIN32.data` field.
7. When the DMA operation completes, the hardware automatically:
  - a. Clears the `CRC_CTRL.busy` field to 0.
  - b. Loads the new CRC value into the `CRC_VAL.value` field.
  - c. Generates a DMA interrupt request.
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.

## 22.6 Registers

See *Table 2-4* for the base address of this peripheral/module. See *Table 2-1* 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 22-4: CRC Register Summary*

Offset	Name	Description
[0x0000]	<a href="#">CRC_CTRL</a>	CRC Control Register
[0x0004]	<a href="#">CRC_DATAIN32</a>	CRC Data Input Register
[0x0008]	<a href="#">CRC_POLY</a>	CRC Polynomial Register
[0x000C]	<a href="#">CRC_VAL</a>	CRC Value Register

### 22.6.1 Register Details

*Table 22-5: CRC Control Register*

CRC Control			CRC_CTRL	[0x0000]
Bits	Field	Access	Reset	Description
31:17	-	RO	0	<b>Reserved</b>
16	busy	R	0	<b>CRC Busy</b> 0: Not busy 1: Busy
15:5	-	RO	0	<b>Reserved</b>
4	byte_swap_out	R/W	0	<b>Byte Swap CRC Value Output</b> 0: <a href="#">CRC_VAL</a> .value is not byte swapped. 1: <a href="#">CRC_VAL</a> .value is byte swapped.
3	byte_swap_in	R/W	0	<b>Byte Swap CRC Data Input</b> 0: <a href="#">CRC_DATAIN32</a> .data is processed least significant byte first. 1: <a href="#">CRC_DATAIN32</a> .data is processed most significant byte first.
2	msb	R/W	0	<b>Most Significant Bit Order</b> 0: LSB data first 1: MSB data first (mirrored)
1	dma_en	R/W	0	<b>DMA Enable</b> Set this field to 1 to enable a DMA request when the output FIFO is full. 0: Disabled 1: Enabled
0	en	R/W	0	<b>CRC Enable</b> 0: Disabled 1: Enabled

*Table 22-6: CRC Data Input 32 Register*

CRC Data In 32-bit			CRC_DATAIN32	[0x0004]
Bits	Field	Access	Reset	Description
31:0	data	W	0	<b>CRC Data Input</b> This register can be written as a word, half-word, or byte. See <i>Table 22-2</i> for details on the byte and bit ordering of the data in this register. <i>Note: Do not write to this register if <a href="#">CRC_CTRL</a>.busy = 1 or <a href="#">CRC_CTRL</a>.en = 0.</i>

*Table 22-7: CRC Polynomial Register*

CRC Polynomial			CRC_POLY	[0x0008]
Bits	Field	Access	Reset	Description
31:0	poly	R/W	0xEDB8 8320	<b>CRC Polynomial</b> See <a href="#">Table 22-2</a> for details on the byte and bit ordering of the data in this register.

*Table 22-8: CRC Value Register*

CRC Value			CRC_VAL	[0x000C]
Bits	Field	Access	Reset	Description
31:0	value	R/W	0	<b>Current CRC Value</b> The software can write to this register to set the initial state of the accelerator. This register should only be read or written when <a href="#">CRC_CTRL.busy</a> = 0. See <a href="#">Table 22-2</a> for details on the byte and bit ordering of the data in this register.

Preliminary Draft 05/21/2021

## 23. 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 RX and TX channels.
- Supports multiple key sources:
  - ◆ Encryption using the external AES key.
  - ◆ Decryption using the external AES key.
  - ◆ Decryption using the locally generated decryption key.

### 23.1 Instances

Instances of the peripheral are listed in *Table 23-1*. Disable the peripheral by clearing *AES\_CTRL.en* = 0 before writing to any register field.

*Table 23-1: MAX78000 AES Instances*

Instance	128-Bit Key	192-Bit Key	256-Bit Key	DMA Support
AES	Yes	Yes	Yes	Yes

### 23.2 Encryption of 128-Bit Blocks of Data Using FIFO

AES operations are typically performed on 128-bit of data at a time. The simplest use case is to have software encrypt 128-bit blocks of data. The *AES\_CTRL.start* field is unused in this case.

1. Generate 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.

### 23.3 Encryption of 128-Bit Blocks Using DMA

For this example, it is assumed that the DMA both reads and writes data to/from the AES 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\_INFL.done* are 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 complete 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 prior to the next calculation being 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 that the interface from the DMA to the AES only works when the amount of data is a multiple of 128-bits. For non-multiples of 128-bits, the remainder after calculating all of the 128-bit blocks must be encrypted manually. See [Encryption of Blocks Less Than 128-Bits](#) for details*

## 23.4 Encryption of Blocks Less Than 128-Bits

The AES engine automatically starts a calculation when 128 bits (four writes of 32 bits) occurs. Operations of less than 128-bits use the start field to initiate the AES calculation.

1. Generate 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 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).

## 23.5 Decryption

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 prior to doing the first decryption to ensure that it has been generated.

## 23.6 Interrupt Events

The peripheral generates interrupts for the events shown in *Table 23-2*. 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 23-2: Interrupt Events*

Event	Local Interrupt Flag	Local Interrupt Enable
Data Output FIFO Overrun	<code>AES_INTFL.ov</code>	<code>AES_INTEN.ov</code>
Key Zero	<code>AES_INTFL.key_zero</code>	<code>AES_INTEN.key_zero</code>
Key Change	<code>AES_INTFL.key_change</code>	<code>AES_INTEN.key_change</code>
Calculation Done	<code>AES_INTFL.done</code>	<code>AES_INTEN.done</code>

### 23.6.1 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 prior to the next calculation being complete, a data output FIFO overrun event occurs and the corresponding local interrupt flag is set.

### 23.6.2 Key Zero

Attempting a calculation with a key of all zeros generates a key zero event.

### 23.6.3 Key Change

Writing to any key register while [AES\\_STATUS.busy = 1](#) generates a key change event.

### 23.6.4 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.

## 23.7 Registers

See [Table 2-4](#) for the base address of this peripheral/module. See [Table 2-1](#) 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 23-3: AES Register Summary*

Offset	Name	Description
[0x0000]	<a href="#">AES_CTRL</a>	AES Control Register
[0x0004]	<a href="#">AES_STATUS</a>	AES Status Register
[0x0008]	<a href="#">AES_INTFL</a>	AES Interrupt Flag Register
[0x000C]	<a href="#">AES_INTEN</a>	AES Interrupt Enable Register
[0x0010]	<a href="#">AES_FIFO</a>	AES Data FIFO

### 23.7.1 Register Details

*Table 23-4: AES Control Register*

AES Control				<a href="#">AES_CTRL</a>	[0x0000]
Bits	Field	Access	Reset	Description	
31:10	-	RO	0	<b>Reserved</b>	
9:8	type	R/W	0	<b>Encryption Type</b> 00: Encryption using the external AES key. 01: Decryption using the external AES key. 10: Decryption using the locally generated decryption key. 11: Reserved	
7:6	key_size	R/W	0	<b>Encryption Key Size</b> 00: 128-bits 01: 192-bits 10: 256-bits 11: Reserved	
5	output_flush	W1	0	<b>Flush Data Output FIFO</b> This field always read 0. 0: No action 1: Flush	

AES Control				AES_CTRL	[0x0000]
Bits	Field	Access	Reset	Description	
4	input_flush	W1	0	<b>Flush Data Input FIFO</b> This field always read 0. 0: No action 1: Flush	
3	start	W1	0	<b>Start AES Calculation</b> This field forces the start of an AES calculation regardless of the state of the data input FIFO. This allows doing an AES calculation on less than 128-bits of data since normally an AES calculation starts when the data input FIFO is full. This field always read 0. 0: No action 1: Start calculation	
2	dma_tx_en	R/W	0	<b>DMA Request To Write Data Input FIFO</b> 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	<b>DMA Request To Read Data Output FIFO</b> When enabled, a DMA request is generated when the data output FIFO is full. 0: Disabled 1: Enabled	
0	en	R/W	0	<b>AES Enable</b> 0: Disabled 1: Enabled.	

Table 23-5: AES Status Register

AES Status				AES_STATUS	[0x0004]
Bits	Field	Access	Reset	Description	
31:5	-	RO	0	<b>Reserved</b>	
4	output_full	R	0	<b>Output FIFO Full</b> 0: Not full 1: Full	
3	output_em	R	0	<b>Output FIFO Empty</b> 0: Not empty 1: Empty	
2	input_full	R	0	<b>Input FIFO Full</b> 0: Not full 1: Full	
1	input_em	R	0	<b>Input FIFO Empty</b> 0: Not empty 1: Empty	
0	busy	R	0	<b>AES Busy</b> 0: Not busy 1: Busy	

Table 23-6: AES Interrupt Flag Register

AES Interrupt Flag				AES_INTFL	[0x0008]
Bits	Field	Access	Reset	Description	
31:4	-	RO	0	<b>Reserved</b>	
3	ov	W1C	0	<b>Data Output FIFO Overrun Event Interrupt</b> 0: No event 1: Event occurred	

AES Interrupt Flag				AES_INTFL	[0x0008]
Bits	Field	Access	Reset	Description	
2	key_zero	W1C	0	<b>Key Zero Event Interrupt</b> 0: No event 1: Event occurred	
1	key_change	W1C	0	<b>Key Change Event Interrupt</b> 0: No event 1: Event occurred	
0	done	W1C	0	<b>Calculation Done Event Interrupt</b> 0: No event 1: Event occurred	

Table 23-7: AES Interrupt Enable Register

AES Interrupt Enable				AES_INTEN	[0x000C]
Bits	Field	Access	Reset	Description	
31:4	-	RO	0	<b>Reserved</b>	
3	ov	W1C	0	<b>Data Output FIFO Overrun Event Interrupt Enable</b> 0: Enabled 1: Disabled	
2	key_zero	W1C	0	<b>Key Zero Event Interrupt Enable</b> 0: Enabled 1: Disabled	
1	key_change	W1C	0	<b>Key Change Event Interrupt Enable</b> 0: Enabled 1: Disabled	
0	done	W1C	0	<b>Calculation Done Event Interrupt Enable</b> This interrupt must be disabled when using the DMA.  0: Enabled 1: Disabled	

Table 23-8: AES FIFO Register

AES Data				AES_FIFO	[0x0010]
Bits	Field	Access	Reset	Description	
31:0	data	R/W	0	<b>AES FIFO</b> 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.	

## 24. TRNG Engine

The Maxim-supplied Universal Cryptographic Library (UCL) provides a function to generate random numbers intended to meet the requirements of commonly security validations. The entropy from one or more internal noise sources continually feeds a TRNG, the output of which is then processed by software and hardware to generate the number returned by the UCL function. Maxim 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 Maxim UCL for the generation of random number.

The TRNG engine can also be used to generate AES keys by firmware using a Hardware Key Derivation Function and using the TRNG as input to the HKDF.

### 24.1 Registers

See [Table 2-4](#) for the base address of this peripheral/module. See [Table 2-1](#) 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 24-1: TRNG Register Summary*

Offset	Register	Name
[0x0000]	<a href="#">TRNG_CTRL</a>	TRNG Control Register
[0x0004]	<a href="#">TRNG_STATUS</a>	TRNG Status Register
[0x0008]	<a href="#">TRNG_DATA</a>	TRNG Data Register

#### 24.1.1 Register Details

*Table 24-2: TRNG Control Register*

Cryptographic Control		TRNG_CTRL			[0x0000]
Bits	Name	Access	Reset	Description	
31:16	-	RO	0	<b>Reserved</b>	
15	keywipe	R/W	0	<b>Wipe Key</b> Write this field to 1 to wipe the TRNG key.	
14:4	-	RO	0	<b>Reserved</b>	
3	keygen	R/W	0	<b>Generate Key</b> Write this field to 1 to generate a key using the TRNG.	
2	-	RO	0	<b>Reserved</b>	
1	rnd_ie	R/W	0	<b>Random Number Interrupt Enable</b> This bit enables an interrupt to be generated when <a href="#">TRNG_STATUS.rdy</a> = 1. 0: Disabled 1: Enabled	
0	-	RO	0	<b>Reserved</b>	

*Table 24-3: TRNG Status Register*

TRNG Status			TRNG_STATUS		[0x0004]
Bits	Name	Access	Reset	Description	
31:1	-	RO	0	<b>Reserved</b>	

TRNG Status				TRNG_STATUS	[0x0004]
Bits	Name	Access	Reset	Description	
0	rdy	R	0	<b>Random Number Ready</b> This bit is automatically cleared to 0 and a new random number is generated when <i>TRNG_DATA.data</i> is read. 0: Random number generation in progress. The content of <i>TRNG_DATA.data</i> is invalid. 1: <i>TRNG_DATA.data</i> contains new 32-bit random data. An interrupt is generated if <i>TRNG_CTRL.rnd_ie</i> = 1.	

Table 24-4: TRNG Data Register

TRNG Data				TRNG_DATA	[0x0008]
Bits	Name	Access	Reset	Description	
31:0	data	RO	0	<b>TRNG Data</b> The 32-bit random number generated is available in this field when <i>TRNG_STATUS.rdy</i> = 1.	

## 25. Bootloader

The ROM bootloader provides for program loading and verification. The physical interface between the external host and the bootloader defaults to the UART.

The secure bootloader (SBL) employs a hash-based message authentication code (HMAC SHA-256) to guarantee both the authenticity and 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 the bootloader and secure bootloader.
- Checksum verification of the ROM image before further ROM execution.
- SWD is disabled in LOCKED and PERMLOCKED states.
- Programmable through the UART or SWD interface.
- UART operates at 115200 bps.
- LOCKED mode disables the 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 SHA256 using secret key-based transactions.
- The 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 the bootloader.

### 25.1 Instances

*Table 25-1* shows the dedicated pins and features of the bootloader.

*Table 25-1:MAX78000 Bootloader Instances*

Part Number	Activation Pins		Bootloader	Secure Bootloader	Secure Boot	Flash Memory Page Size
	UART0 RX	SWDCLK				
MAX78000EXG	P0.0	P0.29	No	Yes	Yes	8KB

## 25.2 Bootloader Operating States

Each bootloader supports the modes shown in *Table 25-2*. Each bootloader mode has a unique prompt.

*Table 25-2: The Bootloader Operating States and Prompts*

State	Bootloader	Secure Bootloader	Recognized Commands	Prompt
<i>UNLOCKED</i>	Yes	Yes	All Commands U/L/P	"ULDR> " <0x55> <0x4C> <0x44> <0x52> <0x3E> <0x20>
<i>LOCKED</i>	Yes	Yes	Only L/P	"LLDR> " <0x4C> <0x4C> <0x44> <0x52> <0x3E> <0x20>
<i>PERMLOCKED</i>	Yes	Yes	Only P	"PLLDR> " <0x50> <0x4C> <0x4C> <0x44> <0x52> <0x3E> <0x20>
<i>CHALLENGE</i>	No	Yes	<i>GC – Get Challenge</i> <i>SR – Send Response</i>	"CR> " <0x43> <0x52> <0x3E> <0x20>
<i>APPVERIFY</i>	No	Yes	N/A	N/A

The *LOCK – Lock Device* and *PLOCK – Permanent Lock* commands do not change the bootloader prompt or take effect until the bootloader is reset.

### 25.2.1 UNLOCKED

The UNLOCKED state provides access to load secure keys and configuration information. Program loading, verification, and status are available in the UNLOCKED state. The SWD interface is available for use.

Transitioning from the LOCKED to UNLOCKED states erases all program memory. In the SBL it also clears the challenge/response and application keys stored by the SBL.

The challenge and application keys can be erased by executing the Unlock command while in the UNLOCKED state and resetting the device. This eliminates the need to transition through the LOCKED state.

### 25.2.2 LOCKED

The LOCKED state disables access to program memory, other than to verify it. The application and challenge/response keys cannot be changed without first changing to the UNLOCKED state.

For the SBL, if the optional challenge key is activated, the bootloader starts in the CHALLENGE state. Successfully completing the challenge/response transitions to the previous PERMLOCKED or LOCKED state.

The application key should be configured before executing the *LOCK – Lock Device* command.

### 25.2.3 PERMLOCKED

The PERMLOCKED state disables access to program memory, other than to verify it with a simple SHA256 hash. The commands available in the PERMLOCKED state are shown in *Table 25-3*.

*Table 25-3: PERMLOCK Command Summary*

Command
<i>H – Check Device</i>
<i>I – Get ID</i>

For the SBL, if the optional challenge key is activated, the bootloader starts in the CHALLENGE state. Successfully completing the challenge/response transitions to the previous PERMLOCKED state.

The application key should be configured before executing the *PLOCK – Permanent Lock* command.

### 25.2.4 CHALLENGE (Secure Bootloader Only)

The CHALLENGE state provides an extra layer of security by requiring the host to authenticate itself using the HMAC SHA-256 key before executing any bootloader commands. If enabled, the device enters CHALLENGE mode following a reset if the external bootloader pins are active. CHALLENGE mode can be identified by the “CR>” prompt.

The host first requests a 128-bit random number, the challenge, from the bootloader in CHALLENGE mode. The host encrypts the challenge using the mutually known HMAC SHA256 key and sends the response back to the bootloader. The correct response transitions from the CHALLENGE state to the previous state of the bootloader. An incorrect response keeps the bootloader in the CHALLENGE state. The host must request a new challenge and send a response based on the new challenge. There is no limit to the number of challenge attempts.

### 25.2.5 APPVERIFY (Secure Bootloader only)

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 in the following conditions:

- 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, enabling the device to 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. See *Creating the Motorola SREC File* for details.
2. The host activates the SBL. See *Bootloader Activation* for details.
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 automatically verifies program memory on all subsequent resets and attempts to execute the Lock, and Plock commands.

## 25.3 Creating the Motorola SREC File

The Motorola SREC file must include the program bytes and the MAC required for the APPVERIFY process. Address records must be 32-bit aligned, and the length of each line must be a multiple of 4 bytes. Any unused memory locations within the program must be defined with a constant value.

The information here is presented for completeness; Maxim Integrated can provide customers with a complete toolset for generating a Motorola SREC file that meets the SBL requirements.

*Note: The length of the Motorola SREC file is not the same as the code length used for the WL command, as explained below.*

The procedure for generating the SREC file is:

1. Define the 128-bit HMAC secret key.
2. Generate a binary image.
3. Pad the binary image with a constant value to the next 32-byte boundary. The address of the last pad byte is the code length argument for the WL command.
4. Calculate the 32-byte HMAC SHA256 using the secret key over the length of the padded binary image.
5. Append the 32-byte HMAC SHA256, calculated in step 4, to the binary image, after the last pad byte.
6. Generate the Motorola SREC file.

## 25.4 Bootloader Activation

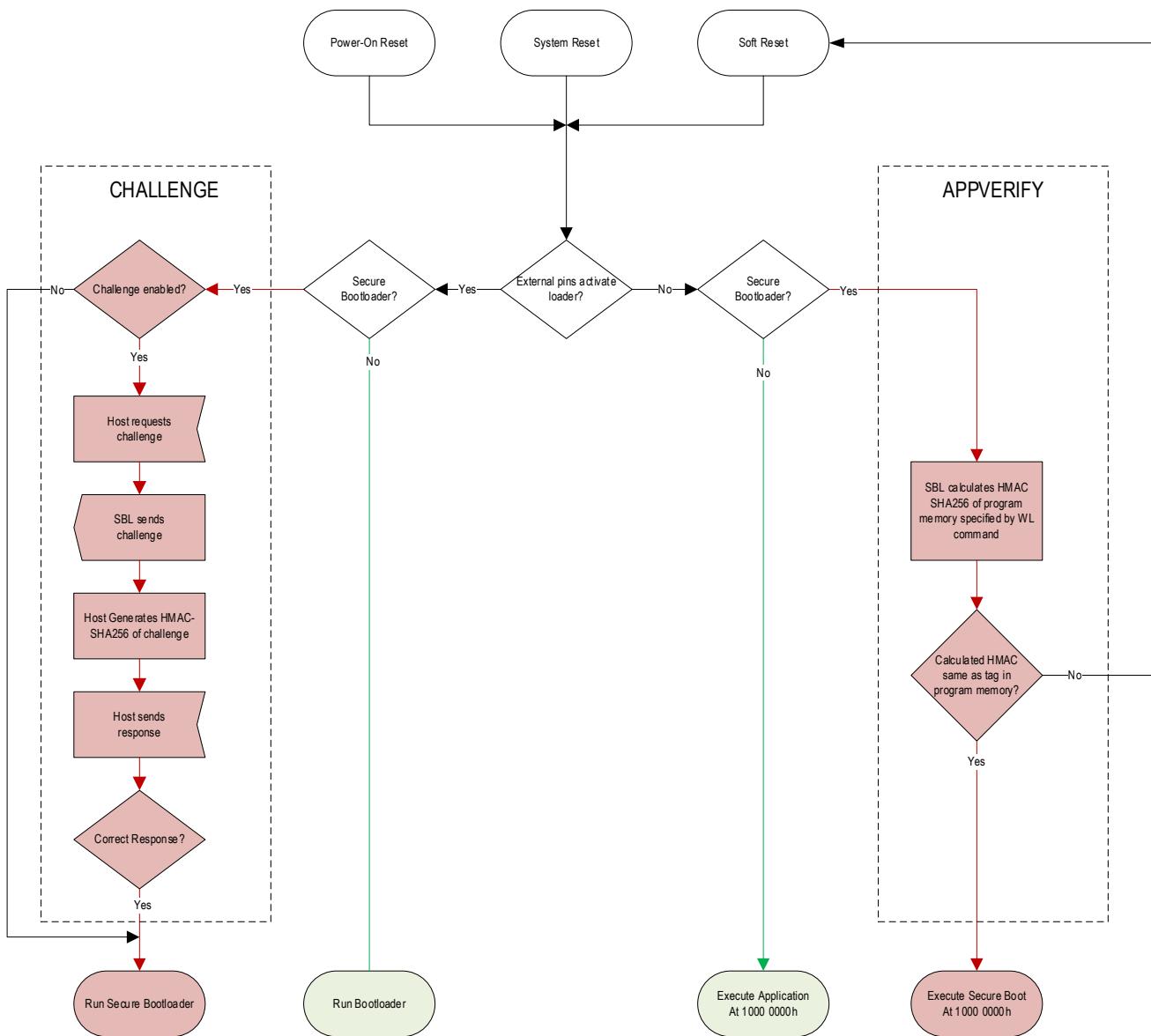
Perform the following sequence to activate the bootloader:

1. The host asserts the UART0 RX pin and SWDCLK pins low, as shown in [Table 25-1](#).
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 activates the bootloader.
5. Bootloader performs its system initialization and configures the bootloader for 115200 bps.
6. The bootloader outputs the status prompt on the UART0 TX pin. The prompt is unique for each bootloader state, as shown in [Table 25-2](#).
7. The host releases the UART0 RX and SWDCLK pins once the host confirms the correct bootloader prompt.
8. The host begins the bootloader session by sending commands on the UART0 RX pin.

## 25.5 Bootloader

The bootloader is invoked following a reset when the bootloader activation. The flow chart of the operation following a reset of the device is shown [Figure 25-1](#). Features exclusive to the SBL are highlighted in red.

*Figure 25-1: MAX78000 Combined Bootloader Flow*



## 25.6 Secure Bootloader

The secure version of the bootloader provides additional features for secure and authenticated loading. These features are highlighted in *Figure 25-1*.

### 25.6.1 Secure Boot

The SBL performs a *Secure Boot* by entering the APPVERIFY state following a reset in which the bootloader activation pins are not active. Failure of the secure boot places the device in a reset loop to prevent the 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 forces the device into a continual reset state.

### 25.6.2 Secure Challenge/Response Authentication

The optional secure Challenge/Response authentication provides an extra layer of security by requiring the host to authenticate itself using the mutual HMAC SHA-256 key before executing any bootloader commands. If the challenge key is activated, the device enters CHALLENGE mode following a reset if the external bootloader pins are active. The bootloader displays the CHALLENGE mode prompt shown in [Table 25-2](#).

Only two commands are available in the CHALLENGE state:

*Table 25-4: CHALLENGE Command Summary*

Command
<a href="#">GC – Get Challenge</a>
<a href="#">SR – Send Response</a>

The host first requests a 128-bit random number (the challenge) from the bootloader. The host encrypts the challenge using the HMAC SHA256 key (the response) and sends it back to the 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 is required after the next device reset. It does not affect the current operation until the next reset.

Follow this procedure to successfully perform the Challenge/Response:

1. The host executes the GC command.
2. 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 returns the prompt of the last bootloader state. An incorrect response returns 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 returns the prompt appropriate to the last state of the SBL.

## 25.7 Command Protocol

The bootloader presents a mode-specific prompt based on the current state of the loader, as shown in [Table 25-2](#). 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.

## 25.8 General Commands

Table 25-5: MAX78000 General Command Summary

Command
<i>L – Load</i>
<i>P – Page Erase</i>
<i>V – Verify</i>
<i>LOCK – Lock Device</i>
<i>PLOCK – Permanent Lock</i>
<i>UNLOCK – Unlock Device</i>
<i>H – Check Device</i>
<i>I – Get ID</i>
<i>S – Status</i>
<i>Q – Quit</i>

### 25.8.1 General Command Details

Table 25-6: L - Load

L - Load	Load Motorola SREC File into Program Memory
Description	Load a Motorola SREC formatted file into Flash program memory. See <a href="#">Creating the Motorola SREC File</a> for the details of the format required for the SBL. After typing the L command, the bootloader responds 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 of the padded image plus the 32-bytes of the HMAC is different than the length used for the WL command.
Modes	U
Command	L<0x0D><0x0A> Ready to load SREC<0x0D><0x0A> [Motorola SREC File]
Response: Success	Load success, image loaded with the following parameters:<0x0D><0x0A> Base address: 0xnnnnnnnn<0x0D><0x0A> Length: 0xnnnnnnnn<0x0D><0x0A>
Response: Failure	Load failed.<0x0D><0x0A>

Table 25-7: P – Page Erase

P – Page Erase	Erase Page of Flash Program Memory
Description	Erases a page of memory associated with the 32-bit input address. Addresses must be aligned on the device-specific page boundaries.
Modes	U
Command	P 0xnnnnnnnn<0x0D><0x0A>
Response: Success	Erase Page Address: 0xnnnnnnnn<0x0D><0x0A>OK<0x0D><0x0A>
Response: Failure	Bad page address input<0x0D><0x0A> or Erase failed<0x0D><0x0A> or Invalid Page Address: 0xnnnnnnnn<0x0D><0x0A>

Table 25-8: V – Verify

V – Verify	Verify Flash Program Memory Against Motorola SREC File
Description	Verifies the contents of fflash program memory against a Motorola SREC file.
Modes	U
Command	V<0x0D><0x0A> Ready to verify SREC<0x0D><0x0A> [Motorola SREC File]
Response: Success	Verify success, image verified with the following parameters: <0x0D><0x0A> Base address: 0xnnnnnnnn<0x0D><0x0A> Length: 0xnnnnnnnn<0x0D><0x0A>
Response: Failure	Verify failed.<0x0D><0x0A>

Table 25-9: LOCK – Lock Device

LOCK – Lock Device	Lock Device
Description	Locks the device and disables SWD on the next device reset. See <i>LOCKED</i> 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 continues to display the locked prompt, but the <i>S – Status</i> command shows the Locked mode is active. The Lock command should be followed by the Q command (which generates a device reset) for the Lock command to take effect. The SBL performs an APPVERIFY check before executing the Lock command. Failure of the Lock command means that the APPVERIFY check failed.
Modes	U
Command	LOCK<0x0D><0x0A>
Response: Success	OK<0x0D><0x0A>
Response: Failure	Failed<0x0D><0x0A>

Table 25-10: PLOCK – Permanent Lock

PLOCK – Permanent Lock	Permanently Lock Device
Description	Permanently locks the device if the argument matches the device ID. The effects of the Plock command do not take effect until the next time the device is reset. The bootloader continues to display the LOCKED or UNLOCKED state prompt, but the <i>S – Status</i> command shows the LOCKED or UNLOCKED state is active. The Lock command should be followed by the Q command (which generates a device reset) for the Lock command to take effect. The SBL performs an APPVERIFY check before executing the Plock command. Failure of the Plock command means that the APPVERIFY check failed.
Modes	U/L
Command	PLOCK <USN><0x0D><0x0A>
Response: Success	OK<0x0D><0x0A>
Response: Failure	Failed<0x0D><0x0A>

**Table 25-11: UNLOCK – Unlock Device**

<b>UNLOCK – Unlock Device</b>	<b>Unlock Device</b>
Description	Changes the bootloader state to UNLOCKED if in the 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 displays the UNLOCKED mode prompt the next time the bootloader is activated.
Response: Failure	None.

**Table 25-12: H – Check Device**

<b>H – Check Device</b>	<b>Perform SHA-256 Hash Over Memory Range</b>
Description	Performs a simple SHA-256 (not HMAC SHA-256) hash of bytes starting at 32-bit address 0xnnnnnnnn to 0xmmmmmmmm. The minimum hash input size is 512 bytes. The function returns a 32-byte hash value.
Modes	U/L/P
Command	H 0xnnnnnnnn 0xmmmmmmmm<0x0D><0x0A>
Response: Success	> yy<0x0D><0x0A>
Response: Failure	<0x0D><0x0A>

**Table 25-13: I – Get ID**

<b>I – Get ID</b>	<b>Read Universal Serial Number</b>
Description	Returns the 13-byte USN of the device.
Modes	U/L/P
Command	I<0x0D><0x0A> USN: nnnnnnnnnnnnnnnnnnnnnnnnnn<0x0D><0x0A>
Response: Success	None
Response: Failure	None

Table 25-14: S – Status

S – Status	Read Device Status
Description	<p>Returns the state of the loader and the application key and challenge key features. This changes during the same session when the command is executed, unlike the prompt, which only changes after reset:</p> <p>The Lock &lt;response&gt; is:          “Inactive” if the device is in the unlocked state.          “Active” if the device is in the locked or permanent lock state.</p> <p>The PLock &lt;response&gt; is:          “Inactive” if the device is in the unlocked or locked state.          “Active” if the device is in the permanent lock state.</p> <p>In addition, the SBL displays:</p> <p>The Application Length &lt;response&gt; is:          “Not Set” if the Write Code Length command has not previously loaded a non-zero value          “0xnnnnnnnn” is the previously entered value using the Write Code Length command.</p> <p>The Application Key &lt;response&gt; is:          “None Inactive” if no application key has been loaded using the LK command.          “Loaded Inactive” if the application key has been loaded, but the application key feature has not been activated by the AK command          “Loaded Active” If the application key has been loaded and the application key feature has been activated.</p> <p>The Challenge Key &lt;response&gt; is:          “None Inactive” if no challenge key has been loaded using the LK command.          “Loaded Inactive” if the challenge key has been loaded, but the challenge key feature has not been activated by the AK command          “Loaded Active” if the challenge key has been loaded and the challenge key feature has been activated.</p>
Modes	U
Command	<pre>S&lt;0xD&gt;&lt;0xA&gt; Status&lt;0xD&gt;&lt;0xA&gt; Lock: &lt;response&gt;&lt;0xD&gt;&lt;0xA&gt; PLock: &lt;response&gt;&lt;0xD&gt;&lt;0xA&gt; Application Length: &lt;response&gt;&lt;0xD&gt;&lt;0xA&gt; Application Key: &lt;response&gt;&lt;0xD&gt;&lt;0xA&gt; Challenge Key: &lt;response&gt;&lt;0xD&gt;&lt;0xA&gt;</pre>
Response: Success	None.
Response: Failure	None.

Table 25-15: Q – Quit

Q – Quit	Quit Bootloader Session
Description	Terminates the bootloader session and forces a reset of the device.
Modes	U/L/P
Command	Q<0xD><0xA>
Response: Success	None
Response: Failure	None

## 25.9 Secure Commands

Table 25-16: MAX78000 Secure Command Summary

Command
<i>LK – Load Application Key</i>
<i>LC – Load Challenge Key</i>
<i>VK – Verify Application Key</i>
<i>VC – Verify Challenge Key</i>
<i>AK – Activate Application Key</i>
<i>AC – Activate Challenge</i>
<i>WL – Write Code Length</i>

### 25.9.1 Secure Command Details

Table 25-17: LK – Load Application Key

LK – Load Application Key	Load Application HMAC-SHA256 Key
Description	Write 128-bit HMAC secret key to non-volatile memory.
Modes	U
Command	LK yyyyooooooooooooooooooooo <sub>128</sub> <0x0D><0x0A>
Response: Success	OK<0x0D><0x0A>
Response: Failure	Bad key input<0x0D><0x0A> or Key already loaded<0x0D><0x0A>

Table 25-18: LK – Load Challenge Key

LC – Load Challenge Key	Load Challenge Key
Description	Write 128-bit challenge key to non-volatile memory.
Modes	U
Command	LC yyyyooooooooooooooooooooo <sub>128</sub> <0x0D><0x0A>
Response: Success	OK<0x0D><0x0A>
Response: Failure	Bad key input<0x0D><0x0A> or Key already loaded<0x0D><0x0A>

Table 25-19: VK – Verify Application Key

VK – Verify Application Key	VK – Verify Application Key
Description	Verify the Application Key against a value provided by the host.
Modes	U
Command	VK yyyyooooooooooooooooooooo <sub>128</sub> <0x0D><0x0A>
Response: Success	OK<0x0D><0x0A>
Response: Failure	Bad key input<0x0D><0x0A> or Error, no key loaded<0x0D><0x0A> or Key Mismatch<0x0D><0x0A>

Table 25-20: VC – Verify Challenge Key

VC – Verify Challenge Key	VC – Verify Challenge Key
Description	Verify the Challenge Key against a value provided by the host.
Modes	U
Command	VC yyy<0x0D><0x0A>
Response: Success	OK<0x0D><0x0A>
Response: Failure	Bad key input<0x0D><0x0A> or Error, no key loaded<0x0D><0x0A> or Key Mismatch<0x0D><0x0A>

Table 25-21: AK – Activate Application Key

AK – Activate Application Key	Activate Application Key
Description	Activate application key. All application software loads must be encrypted with the application key. The UNLOCK command deactivates the application key.
Modes	U
Command	AK<0x0D><0x0A>
Response: Success	OK<0x0D><0x0A>
Response: Failure	Key activate failed<0x0D><0x0A> or Error, no key loaded<0x0D><0x0A>

Table 25-22: AC – Activate Challenge Key

AC – Activate Challenge Mode	Activate Challenge Mode
Description	Activate CHALLENGE mode. All subsequent bootloader sessions in LOCKED or PERMLOCKED states 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 re-enter the keys and activate the challenge mode again.
Modes	U
Command	AC<0x0D><0x0A>
Response: Success	OK<0x0D><0x0A>
Response: Failure	Key activate failed<0x0D><0x0A> or Error, no key loaded<0x0D><0x0A>

Table 25-23: WL – Write Code Length

WL – Write Code Length	Write Code Length
Description	Write the length of the application code in bytes as calculated in <a href="#">Creating the Motorola SREC File</a> . The “Write length failed” response indicates the WL command was previously performed. The host should use the SBL to re-enter the UNLOCKED state to clear the WL value and repeat the command.
Modes	U
Command	WL 0xnnnnnnnn<0x0D><0x0A>
Response: Success	Length set to: 0xnnnnnnnn<0x0D><0x0A>
Response: Failure	Bad length input<0x0D><0x0A> OR Write length failed<0x0D><0x0A>

## 25.10 Challenge/Response Commands

Table 25-24: MAX78000 Challenge/Response Command Summary

Register Name
<a href="#">GC – Get Challenge</a>
<a href="#">SR – Send Response</a>

### 25.10.1 Challenge/Response Command Details

Table 25-25: GC – Get Challenge

GC – Get Challenge	Get Challenge
Description	Bootloader generates a 16-byte hexadecimal ASCII challenge and transmits it to the host. The challenge is sent MSB first.
Modes	L/P
Command	GC<0x0D><0x0A>
Response: Success	0123456789ABCDEF0123456789ABCDEF<0x0D><0x0A>
Response: Failure	None

Table 25-26: SR – Send Response

SR – Send Response	Send Response
Description	The host calculates HMAC SHA-256 on the 16-byte challenge and sends the 32-byte hexadecimal ASCII response to SBL. The response is sent MSB first.
Modes	L/P
Command	SR 0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF<0x0D><0x0A>
Response: Success	OK<0x0D><0x0A>
Response: Failure	Bad response input<0x0D><0x0A> Or Verification failed<0x0D><0x0A> Or Error, request challenge<0x0D><0x0A>

## 26. Convolutional Neural Network (CNN)

### 26.1 Overview

The CNN accelerator consists of 64 parallel processors with 442KB weight memory, 512KB of data memory, and 384KB cache SRAM. Each processor includes a pooling unit and a convolutional engine with dedicated weight memory. Four processors share one data memory. These are further organized into groups of 16 processors that share common controls. A group of 16 processors operates as a slave to another group or independently. Data is read from SRAM associated with each processor and written to any data memory located within the accelerator. Any given processor has visibility of its dedicated weight memory, and to the data memory instance, it shares with the three others.

Maxim provides a complete tool set for the CNN including model training and synthesis on the Maxim Integrate AI github repository. See [https://github.com/MaximIntegratedAI/MaximAI\\_Documentation](https://github.com/MaximIntegratedAI/MaximAI_Documentation) for full suite of tools and training available for the CNN.

The features of the CNN accelerator include:

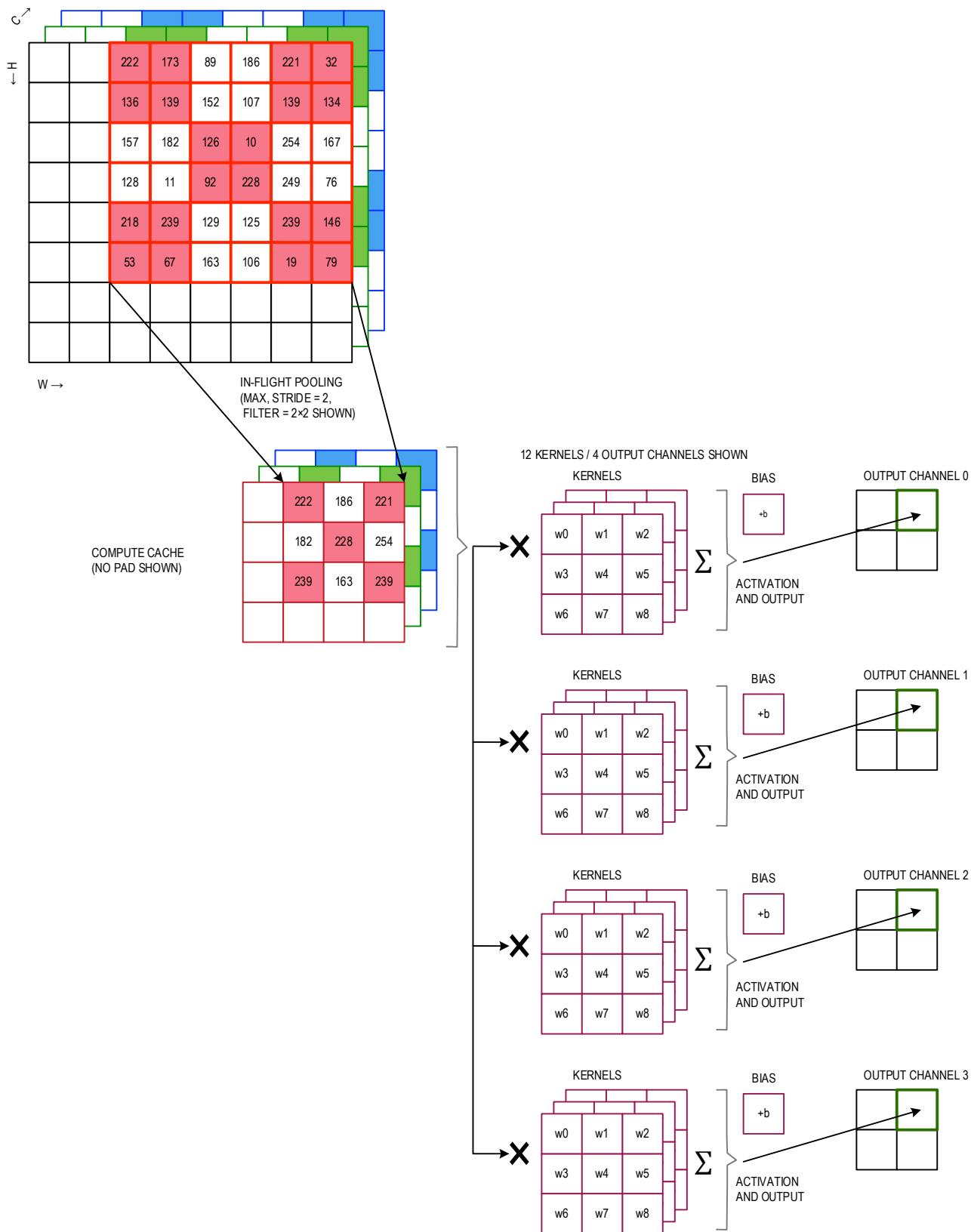
- 512KB SRAM data storage:
  - ◆ Configured as 8Kx8-bit integers x64 channels or 32Kx8-bit integers x4 channels for input layers.
  - ◆ Hardware load and unload assist.
- 64 parallel physical channel processors:
  - ◆ Organized as 4x16 processors.
  - ◆ 8-bit integer data path with option for 32-bit integers on the output layer.
  - ◆ Per-channel processor enable/disable.
  - ◆ Expandable to 1024 parallel logical channel processors.
- 1x1 or 3x3 2D kernel sizes.
- Configurable 1D kernel size to 1x9.
- Full resolution sum-of-product arithmetic for 1024 8-bit integer channels.
- Operating frequency up to 50MHz.
- Nominal 1 output channel per clock, maximum 4 output channels per clock (passthru).
- Configurable input layer image size:
  - ◆ 32K pixels, 16 channels, non-streaming.
  - ◆ 8K pixels, 4 channels, non-streaming.
  - ◆ 1024x1024 pixels, 4 channels, streaming.
- Hidden layers:
  - ◆ Up to 8K 8-bit integer data per channel, x64 channels, non-streaming.
  - ◆ 8K bytes can be split equally across 1 to 16 logical channels, non-streaming.
  - ◆ 1M 8-bit integer data per channel, x64 channels, streaming.
  - ◆ 1M bytes can be split equally across eight layers, streaming.

- Optional interrupt on CNN completion.
- User-accessible BIST on all SRAM storage.
- User-accessible zeroization of all SRAM storage.
- Single-step operation with full data SRAM access for CNN operation debug.
- Flexible power management:
  - ◆ Independent x16 processor supply enables.
  - ◆ Independent x16 processor mask retention enables.
  - ◆ Independent x16 data path clock enables.
  - ◆ Active Arm peripheral bus clock gating with per x16 processor override.
  - ◆ CNN clock frequency scaling (divide by 2, 4, 8, 16).
  - ◆ Chip-level voltage control for performance power optimization.
- Configurable weight storage:
  - ◆ SRAM-based weight storage with selectable data retention.
  - ◆ Configurable from 442K 8-bit integer weights to 3.456M 1-bit logical weights:
    - Organized as  $768 \times 9 \times 64$  8-bit integer weights to  $768 \times 72 \times 64$  1-bit logical weights.
    - Can be configured on a per-layer basis.
  - ◆ Programmable per x16 processor weight RAM start address, start pointer, and mask count.
  - ◆ Optional weight load hardware assist for packed weight storage.
- 32 independently configurable layer groups:
  - ◆ Each group can contain element-wise, and/or pooling, and/or convolution operations for a minimum of 32 and a maximum of 96 layers.
  - ◆ Processor and mask enables (16 channels).
  - ◆ Input data format.
  - ◆ Per-layer data streaming:
    - Stream start - relative to prior stream.
    - Dual-stream processing delay counters - 1 column, 1 row delta counter.
    - Data SRAM circular buffer size.
  - ◆ Input data size (row, column).
  - ◆ Row and column padding 0 to 4 bytes.
  - ◆ Number of input channels 1 to 1024.
  - ◆ Kernel bit width size (1, 2, 4, 8).
  - ◆ Kernel SRAM start pointer and count.
  - ◆ Inflight input image pooling:
    - Pool mode - none, maximum or average.
    - Pool size - 1x1 to 16x16.
  - ◆ Stride - 1 row, 1 column to 4 rows, 4 columns.
  - ◆ Data SRAM read pointer base address.
  - ◆ Data SRAM write pointer configuration:
    - Base address.
    - Independent offsets for output channel storage in SRAM.
    - Programmable stride increment offset.
  - ◆ Bias - 2048 8-bit integers with option for 512 32-bit integers.
  - ◆ Pre-activation output scaling from 0 to 8 bits.
  - ◆ Output activation - none, ReLU, absolute value.
  - ◆ Passthru - 8-bit or 32-bit integers.
  - ◆ Element-wise operations (add, subtract, xor, or) with optional convolution - up to 16 elements.
  - ◆ Deconvolution (upsampling).
  - ◆ Flattening for MLP processing.
  - ◆ 1x1 convolution.

A typical CNN operation consists of pooling followed by a convolution. While these are traditionally expressed as separate layers, pooling can be done “in-flight” on the MAX78000 for greater efficiency.

The accelerator is optimized for convolutions with in-flight pooling on a sequence of layers to minimize data movement. The MAX78000 also supports in-flight element-wise operations, pass-through layers, and 1-D convolutions without element-wise operations. [Figure 26-1](#) shows a high-level diagram of the MAX78000's convolutional neural network flow.

*Figure 26-1: CNN Overview*



Preliminary Draft 05/21/2021

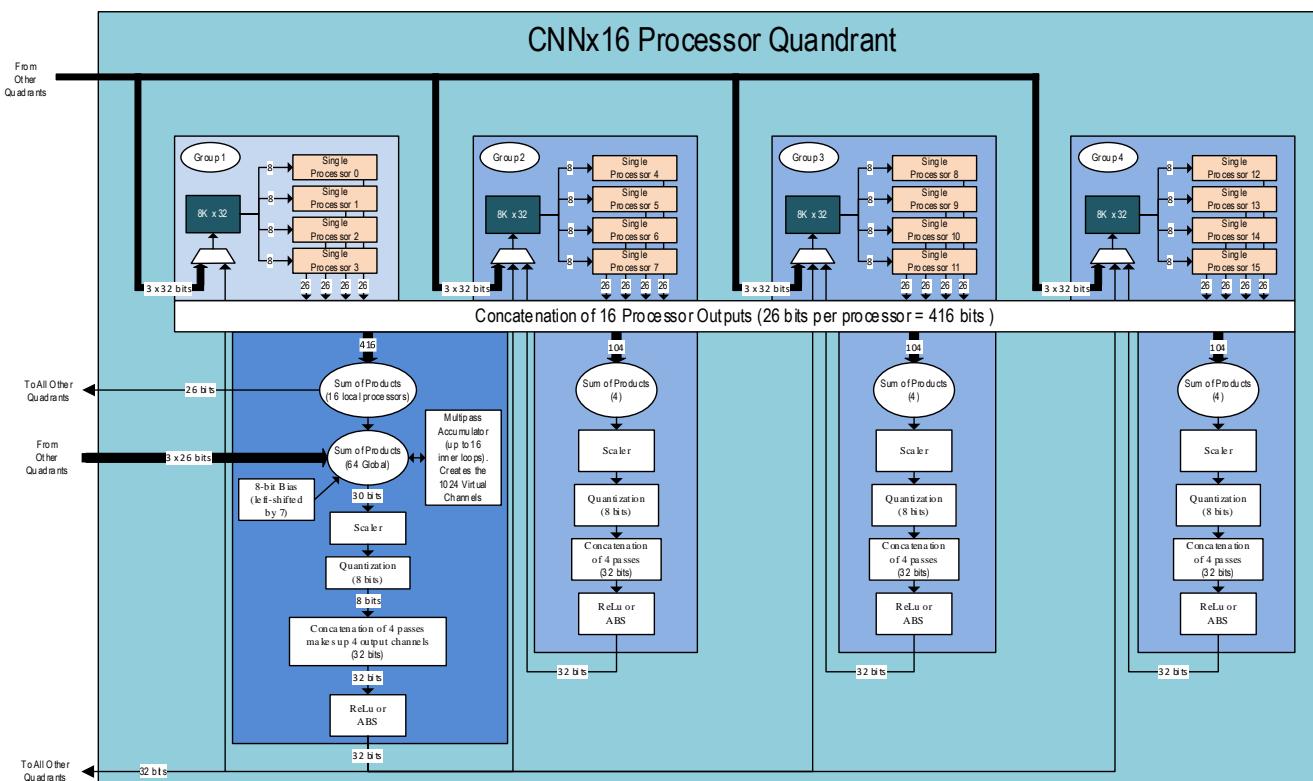
## 26.2 Instances

There is one instance of the CNN. The CNN contains four CNNx16 processor quadrants, generically referred to as CNNx16\_n. Each CNNx16\_n processor quadrant contains sixteen processors grouped in four groups of four. These groups are labeled group 1 to group 4 in the CNNx16\_processor quadrant block diagram and are referred to as CNNx16\_n\_q0 to CNNx16\_n\_q3 in the documentation.

### 26.2.1 Block Diagram

*Figure 26-2* shows a block diagram of a single CNNx16 Processor Quadrant. The MAX78000 includes four CNNx16 processor quadrants. The processor groups are labeled in *Figure 26-2* as Group 1 to Group 4.

*Figure 26-2: CNNx16\_n Processor Quadrant Block Diagram*



## 26.3 Memory Configuration

The CNN includes four CNNx16 processor arrays, and each processor array includes 128KB of SRAM, 192KB of mask RAM (MRAM), 32KB of bias RAM (BRAM), and 192KB of tornado RAM (TRAM). *Figure 26-3* shows the APB mapping for the CNN and the Quad CNNx16n processor arrays. *Table 26-1* shows the memory address range for each type of available memory in each CNNx16n processor array.

*Figure 26-3: CNN Global and Quad CNNx16n Processor Array APB Memory Map*

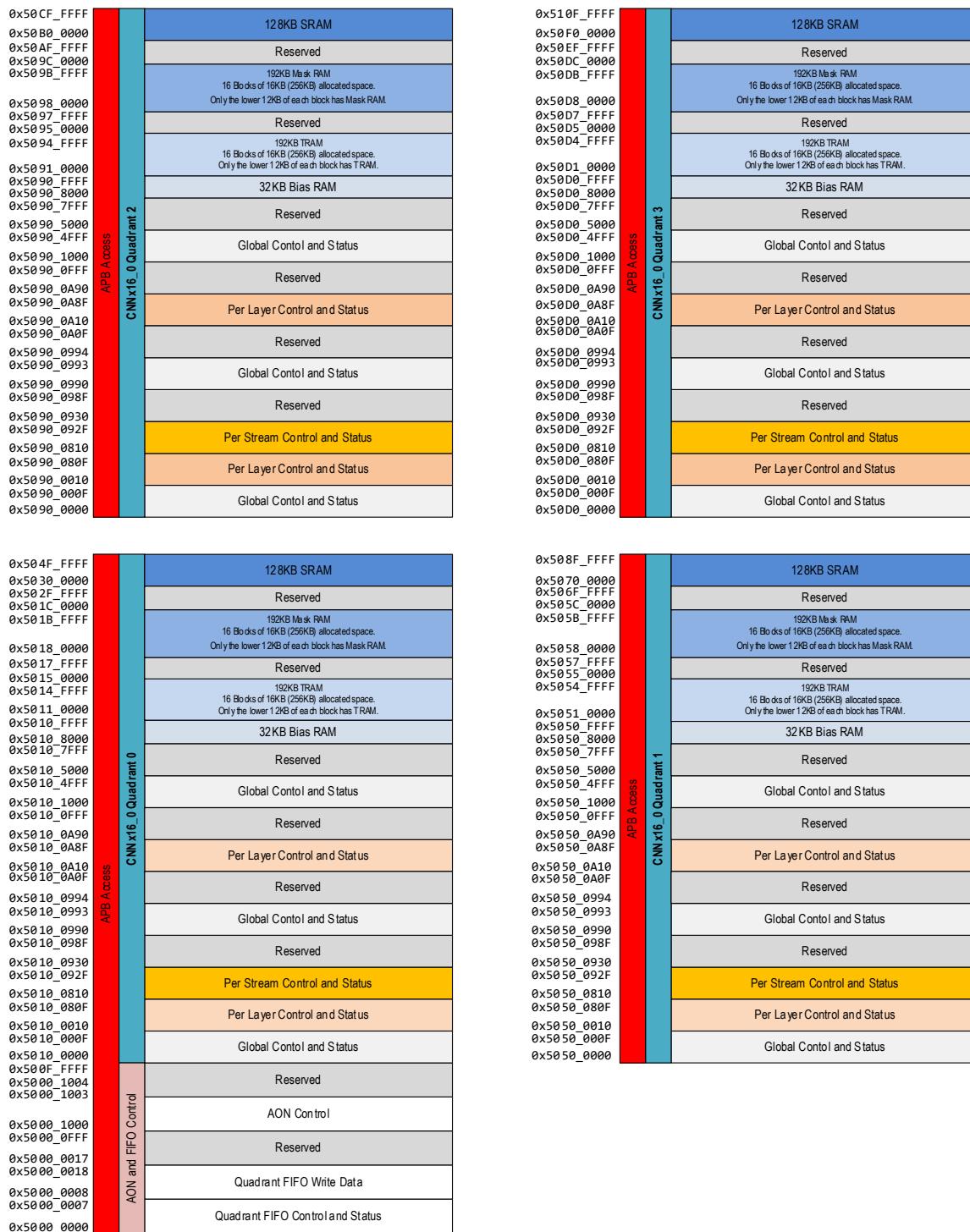


Table 26-1: CNN Memory Configuration (APB Accessible) for the Quad CNNx16n

CNN Quadrant#	Memory Type	Size	Start Address	End Address
CNNx16_0	BRAM	32KB	0x5010 8000	0x5010 FFFF
	TRAM	192KB	0x5011 0000	0x5014 FFFF
	MRAM	192KB	0x5018 0000	0x501B FFFF
	SRAM	128KB	0x5030 0000	0x5032 FFFF
CNNx16_1	BRAM	32KB	0x5050 8000	0x5050 FFFF
	TRAM	192KB	0x5051 0000	0x5054 FFFF
	MRAM	192KB	0x5058 0000	0x505B FFFF
	SRAM	128KB	0x5070 0000	0x5082 FFFF
CNNx16_2	BRAM	32KB	0x5090 8000	0x5090 FFFF
	TRAM	192KB	0x5091 0000	0x5094 FFFF
	MRAM	192KB	0x5098 0000	0x509B FFFF
	SRAM	128KB	0x50B0 0000	0x50B2 FFFF
CNNx16_3	BRAM	32KB	0x50D0 8000	0x50D0 FFFF
	TRAM	192KB	0x50D1 0000	0x50D4 FFFF
	MRAM	192KB	0x50D8 0000	0x50DB FFFF
	SRAM	128KB	0x50F0 0000	0x510F2 FFFF

### 26.3.1 CNNx16\_n TRAM Details

Each CNNx16 processor array includes 192KB of Tornado RAM (TRAM); however, the memory allocated to this region spans 256KB addressable memory. The TRAM is arranged as 16 blocks of 16KB of addressable memory space; however, only the first 12KB of each of these 16 blocks contains TRAM.

*Table 26-2*, *Table 26-3*, *Table 26-4*, and *Table 26-5* show each of the four CNNx16 Processor Array's TRAM start address and the valid TRAM end address for the 16 TRAM blocks. Memory addresses between the *Valid TRAM Block End Address* and the *Block End Address* are invalid memory addresses and must not be used.

Table 26-2: CNNx16 Processor Array 0 TRAM Mapping Details (APB Accessible)

CNN Quadrant#	TRAM Block	Size	Start Address	Valid TRAM Block End Address	Block End Address
CNNx16_0	0	12KB	0x5011 0000	0x5011 2FFF	0x5011 3FFF
	1	12KB	0x5011 4000	0x5011 6FFF	0x5011 7FFF
	2	12KB	0x5011 8000	0x5011 AFFF	0x5011 BFFF
	3	12KB	0x5011 C000	0x5011 EFFF	0x5011 FFFF
	4	12KB	0x5012 0000	0x5012 2FFF	0x5012 3FFF
	5	12KB	0x5012 4000	0x5012 6FFF	0x5012 7FFF
	6	12KB	0x5012 8000	0x5012 AFFF	0x5012 BFFF
	7	12KB	0x5012 C000	0x5012 EFFF	0x5012 FFFF
	8	12KB	0x5013 0000	0x5013 2FFF	0x5013 3FFF
	9	12KB	0x5013 4000	0x5013 6FFF	0x5013 7FFF
	10	12KB	0x5013 8000	0x5013 AFFF	0x5013 BFFF
	11	12KB	0x5013 C000	0x5013 EFFF	0x5013 FFFF
	12	12KB	0x5014 0000	0x5014 2FFF	0x5014 3FFF
	13	12KB	0x5014 4000	0x5014 6FFF	0x5014 7FFF

CNN Quadrant#	TRAM Block	Size	Start Address	Valid TRAM Block End Address	Block End Address
	14	12KB	0x5014 0000	0x5014 AFFF	0x5014 BFFF
	15	12KB	0x5014 0000	0x5014 EFFF	0x5014 FFFF

Table 26-3: CNNx16 Processor Array 1 TRAM Mapping Details (APB Accessible)

CNN Quadrant#	TRAM Block	Size	Start Address	Valid TRAM Block End Address	Block End Address
CNNx16_1	0	12KB	0x5051 0000	0x5051 2FFF	0x5051 3FFF
	1	12KB	0x5051 4000	0x5051 6FFF	0x5051 7FFF
	2	12KB	0x5051 8000	0x5051 AFFF	0x5051 BFFF
	3	12KB	0x5051 C000	0x5051 EFFF	0x5051 FFFF
	4	12KB	0x5052 0000	0x5052 2FFF	0x5052 3FFF
	5	12KB	0x5052 4000	0x5052 6FFF	0x5052 7FFF
	6	12KB	0x5052 8000	0x5052 AFFF	0x5052 BFFF
	7	12KB	0x5052 C000	0x5052 EFFF	0x5052 FFFF
	8	12KB	0x5053 0000	0x5053 2FFF	0x5053 3FFF
	9	12KB	0x5053 4000	0x5053 6FFF	0x5053 7FFF
	10	12KB	0x5053 8000	0x5053 AFFF	0x5053 BFFF
	11	12KB	0x5053 C000	0x5053 EFFF	0x5053 FFFF
	12	12KB	0x5054 0000	0x5054 2FFF	0x5054 3FFF
	13	12KB	0x5054 0000	0x5054 6FFF	0x5054 7FFF
	14	12KB	0x5054 0000	0x5054 AFFF	0x5054 BFFF
	15	12KB	0x5054 0000	0x5054 EFFF	0x5054 FFFF

Table 26-4: CNNx16 Processor Array 2 TRAM Mapping Details (APB Accessible)

CNN Quadrant#	TRAM Block	Size	Start Address	Valid TRAM Block End Address	Block End Address
CNNx16_2	0	12KB	0x5091 0000	0x5091 2FFF	0x5091 3FFF
	1	12KB	0x5091 4000	0x5091 6FFF	0x5091 7FFF
	2	12KB	0x5091 8000	0x5091 AFFF	0x5091 BFFF
	3	12KB	0x5091 C000	0x5091 EFFF	0x5091 FFFF
	4	12KB	0x5092 0000	0x5092 2FFF	0x5092 3FFF
	5	12KB	0x5092 4000	0x5092 6FFF	0x5092 6FFF
	6	12KB	0x5092 8000	0x5092 AFFF	0x5092 BFFF
	7	12KB	0x5092 C000	0x5092 EFFF	0x5092 FFFF
	8	12KB	0x5093 0000	0x5093 2FFF	0x5093 3FFF
	9	12KB	0x5093 4000	0x5093 6FFF	0x5093 7FFF
	10	12KB	0x5093 8000	0x5093 AFFF	0x5093 BFFF
	11	12KB	0x5093 C000	0x5093 EFFF	0x5093 FFFF
	12	12KB	0x5094 0000	0x5094 2FFF	0x5094 3FFF
	13	12KB	0x5094 0000	0x5094 6FFF	0x5094 7FFF
	14	12KB	0x5094 0000	0x5094 AFFF	0x5094 BFFF
	15	12KB	0x5094 0000	0x5094 EFFF	0x5094 FFFF

*Table 26-5: CNNx16 Processor Array 3 TRAM Mapping Details (APB Accessible)*

CNN Quadrant#	TRAM Block	Size	Start Address	Valid TRAM Block End Address	Block End Address
CNNx16_3	0	12KB	0x50D1 0000	0x50D1 2FFF	0x50D1 3FFF
	1	12KB	0x50D1 4000	0x50D1 6FFF	0x50D1 7FFF
	2	12KB	0x50D1 8000	0x50D1 AFFF	0x50D1 BFFF
	3	12KB	0x50D1 C000	0x50D1 EFFF	0x50D1 FFFF
	4	12KB	0x50D2 0000	0x50D2 2FFF	0x50D2 3FFF
	5	12KB	0x50D2 4000	0x50D2 6FFF	0x50D2 7FFF
	6	12KB	0x50D2 8000	0x50D2 AFFF	0x50D2 BFFF
	7	12KB	0x50D2 C000	0x50D2 EFFF	0x50D2 FFFF
	8	12KB	0x50D3 0000	0x50D3 2FFF	0x50D3 3FFF
	9	12KB	0x50D3 4000	0x50D3 6FFF	0x50D3 7FFF
	10	12KB	0x50D3 8000	0x50D3 AFFF	0x50D3 BFFF
	11	12KB	0x50D3 C000	0x50D3 EFFF	0x50D3 FFFF
	12	12KB	0x50D4 0000	0x50D4 2FFF	0x50D4 3FFF
	13	12KB	0x50D4 0000	0x50D4 6FFF	0x50D4 7FFF
	14	12KB	0x50D4 0000	0x50D4 AFFF	0x50D4 BFFF
	15	12KB	0x50D4 0000	0x50D4 EFFF	0x50D4 FFFF

### 26.3.2 CNNx16\_n MRAM Details

Each CNNx16 processor array includes 192KB of MRAM; however, the memory allocated to this region spans 256KB address space. The MRAM is arranged as 16 blocks of 16KB of addressable memory space; however, only the first 12KB of each of these 16 blocks contains usable MRAM.

The following tables below (*Table 26-6*, *Table 26-7*, *Table 26-8*, and *Table 26-9*) show each of the four CNNx16 Processor Array's MRAM start address and the valid MRAM end address for the 16 MRAM blocks. Memory addresses between the *Valid MRAM Block End Address* and the *Block End Address* are invalid memory addresses and cannot be used.

*Table 26-6: CNNx16 Processor Array 0 MRAM Mapping Details (APB Accessible)*

CNN Quadrant#	MRAM Block	Size	MRAM Block Start Address	Valid MRAM Block End Address	Block End Address
CNNx16_0	0	12KB	0x5018 0000	0x5018 2FFF	0x5018 3FFF
	1	12KB	0x5018 4000	0x5018 6FFF	0x5018 7FFF
	2	12KB	0x5018 8000	0x5018 AFFF	0x5018 BFFF
	3	12KB	0x5018 C000	0x5018 EFFF	0x5018 FFFF
	4	12KB	0x5019 0000	0x5019 2FFF	0x5019 3FFF
	5	12KB	0x5019 4000	0x5019 6FFF	0x5019 7FFF
	6	12KB	0x5019 8000	0x5019 AFFF	0x5019 BFFF
	7	12KB	0x5019 C000	0x5019 EFFF	0x5019 FFFF
	8	12KB	0x501A 0000	0x501A 2FFF	0x501A 3FFF
	9	12KB	0x501A 4000	0x501A 6FFF	0x501A 7FFF
	10	12KB	0x501A 8000	0x501A AFFF	0x501A BFFF
	11	12KB	0x501A C000	0x501A EFFF	0x501A FFFF
	12	12KB	0x501B 0000	0x501B 2FFF	0x501B 3FFF
	13	12KB	0x501B 0000	0x501B 6FFF	0x501B 7FFF

CNN Quadrant#	MRAM Block	Size	MRAM Block Start Address	Valid MRAM Block End Address	Block End Address
	14	12KB	0x501B 0000	0x501B AFFF	0x501B BFFF
	15	12KB	0x501B 0000	0x501B EFFF	0x501B FFFF

Table 26-7: CNNx16 Processor Array 1 MRAM Mapping Details (APB Accessible)

CNN Quadrant#	MRAM Block	Size	MRAM Block Start Address	Valid MRAM Block End Address	Block End Address
CNNx16_1	0	12KB	0x5058 0000	0x50D8 2FFF	0x5058 3FFF
	1	12KB	0x5058 4000	0x50D8 6FFF	0x5058 7FFF
	2	12KB	0x5058 8000	0x50D8 AFFF	0x5058 BFFF
	3	12KB	0x5058 C000	0x50D8 EFFF	0x5058 FFFF
	4	12KB	0x5059 0000	0x50D9 2FFF	0x5059 3FFF
	5	12KB	0x5059 4000	0x50D9 6FFF	0x5059 7FFF
	6	12KB	0x5059 8000	0x50D9 AFFF	0x5059 BFFF
	7	12KB	0x5059 C000	0x50D9 EFFF	0x5059 FFFF
	8	12KB	0x505A 0000	0x50DA 2FFF	0x505A 3FFF
	9	12KB	0x505A 4000	0x50DA 6FFF	0x505A 7FFF
	10	12KB	0x505A 8000	0x50DA AFFF	0x505A BFFF
	11	12KB	0x505A C000	0x50DA EFFF	0x505A FFFF
	12	12KB	0x505B 0000	0x50DB 2FFF	0x50DB 3FFF
	13	12KB	0x505B 4000	0x50DB 6FFF	0x50DB 7FFF
	14	12KB	0x505B 8000	0x50DB AFFF	0x50DB BFFF
	15	12KB	0x505B C000	0x50DB EFFF	0x50DB FFFF

Table 26-8: CNNx16 Processor Array 2 MRAM Mapping Details (APB Accessible)

CNN Quadrant#	MRAM Block	Size	Start Address	Valid MRAM Block End Address	Block End Address
CNNx16_2	0	12KB	0x5098 0000	0x5098 2FFF	0x5098 3FFF
	1	12KB	0x5098 4000	0x5098 6FFF	0x5098 7FFF
	2	12KB	0x5098 8000	0x5098 AFFF	0x5098 CFFF
	3	12KB	0x5098 C000	0x5098 EFFF	0x5098 FFFF
	4	12KB	0x5099 0000	0x5099 2FFF	0x5099 3FFF
	5	12KB	0x5099 4000	0x5099 6FFF	0x5099 7FFF
	6	12KB	0x5099 8000	0x5099 AFFF	0x5099 BFFF
	7	12KB	0x5099 C000	0x5099 EFFF	0x5099 FFFF
	8	12KB	0x509A 0000	0x509A 2FFF	0x509A 3FFF
	9	12KB	0x509A 4000	0x509A 6FFF	0x509A 7FFF
	10	12KB	0x509A 8000	0x509A AFFF	0x509A BFFF
	11	12KB	0x509A C000	0x509A 3FFF	0x509A FFFF
	12	12KB	0x509B 0000	0x509B 2FFF	0x509B 3FFF
	13	12KB	0x509B 4000	0x509B 6FFF	0x509B 7FFF
	14	12KB	0x509B 8000	0x509B AFFF	0x509B BFFF
	15	12KB	0x509B C000	0x509B EFFF	0x509B FFFF

Table 26-9: CNNx16 Processor Array3 MRAM Mapping Details (APB Accessible)

CNN Quadrant#	MRAM Block	Size	Memory Start Address	Valid MRAM Block End Address	Block End Address
CNNx16_3	0	12KB	0x50D8 0000	0x50D8 2FFF	0x50D8 3FFF
	1	12KB	0x50D8 4000	0x50D8 6FFF	0x50D8 7FFF
	2	12KB	0x50D8 8000	0x50D8 AFFF	0x50D8 CFFF
	3	12KB	0x50D8 C000	0x50D8 EFFF	0x50D8 FFFF
	4	12KB	0x50D9 0000	0x50D9 2FFF	0x50D9 3FFF
	5	12KB	0x50D9 4000	0x50D9 6FFF	0x50D9 7FFF
	6	12KB	0x50D9 8000	0x50D9 AFFF	0x50D9 BFFF
	7	12KB	0x50D9 C000	0x50D9 EFFF	0x50D9 FFFF
	8	12KB	0x50DA 0000	0x50DA 2FFF	0x50DA 3FFF
	9	12KB	0x50DA 4000	0x50DA 6FFF	0x50DA 7FFF
	10	12KB	0x50DA 8000	0x50DA AFFF	0x50DA BFFF
	11	12KB	0x50DA C000	0x50DA EFFF	0x50DA FFFF
	12	12KB	0x50DB 0000	0x50DB 2FFF	0x50DB 3FFF
	13	12KB	0x50DB 0000	0x50DB 6FFF	0x50DB 7FFF
	14	12KB	0x50DB 0000	0x50DB AFFF	0x50DB BFFF
	15	12KB	0x50DB 0000	0x50DB EFFF	0x50DB FFFF

## 26.4 CNN Global Registers (CNN)

See [Table 2-4](#) for the base address of this peripheral/module. There is one instance of the CNN in the MAX78000. The global CNN registers are shown in [Table 26-10](#). See [Table 2-1](#) 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 26-10: Global CNN Register Summary

Offset	Register	Description
[0x0000]	<a href="#">CNN_FIFO_CTRL</a>	CNN FIFO Control Register
[0x0004]	<a href="#">CNN_FIFO_STAT</a>	CNN FIFO Status Register
[0x0008]	<a href="#">CNN_FIFO_WRO</a>	CNN FIFO 0 Write Register
[0x000C]	<a href="#">CNN_FIFO_WR1</a>	CNN FIFO 1 Write Register
[0x0010]	<a href="#">CNN_FIFO_WR2</a>	CNN Fast FIFO 2 Write Register
[0x0014]	<a href="#">CNN_FIFO_WR3</a>	CNN FIFO 3 Write Register
[0x1000]	<a href="#">CNN_AOD_CTRL</a>	CNN AoD Control Register

### 26.4.1 Global CNN Register Details

Table 26-11: CNN FIFO Control Register

CNN FIFO Control		CNN_FIFO_CTRL		[0x0000]
Bits	Field	Access	Reset	Description
31:28	almost_empty_int_en	R/W	0	<b>FIFO Almost Empty Interrupt Enable</b> Each bit of this field maps to one of the CNNx16_n quadrants. Bit 3 maps to CNNx16_n_q3, bit 2 maps to CNNx16_n_q2, bit 1 maps to CNNx16_n_q1, and bit 0 maps to CNNx16_n_q0. Set this field to 1 to enable an interrupt event based on the <a href="#">CNN_FIFO_CTRL.empty_thresh</a> flag. Enable the interrupt for all quadrants by setting this field to 0b1111.

CNN FIFO Control			CNN_FIFO_CTRL		[0x0000]
Bits	Field	Access	Reset	Description	
27:24	almost_full_int_en	R/W	0	<b>FIFO Almost Full Interrupt Enable</b> Each bit of this field maps to one of the CNNx16_n quadrants. Bit 3 maps to CNNx16_n_q3, bit 2 maps to CNNx16_n_q2, bit 1 maps to CNNx16_n_q1, and bit 0 maps to CNNx16_n_q0. Set this field to 1 to enable an IRQ event based on the FIFO almost full threshold flag, <a href="#">CNN_FIFO_CTRL.full_thresh</a> . Enable the IRQ for all quadrants by setting this field to 0b1111.	
23:20	empty_int_en	R/W	0	<b>FIFO Empty Interrupt Enable</b> Each bit of this field maps to one of the CNNx16_n quadrants. Bit 3 maps to CNNx16_n_q3, bit 2 maps to CNNx16_n_q2, bit 1 maps to CNNx16_n_q1, and bit 0 maps to CNNx16_n_q0. Set this field to 1 to enable an IRQ event based on the FIFO empty flag. Enable the IRQ for all quadrants by setting this field to 0b1111	
19:16	full_int_en	R/W	0	<b>FIFO Full Interrupt Enable</b> Each bit of this field maps to one of the CNNx16_n quadrants. Bit 3 maps to CNNx16_n_q3, bit 2 maps to CNNx16_n_q2, bit 1 maps to CNNx16_n_q1, and bit 0 maps to CNNx16_n_q0. Set this field to 1 to enable an IRQ event based on the FIFO full flag. Enable the IRQ for all quadrants by setting this field to 0b1111.	
15:12	fifo_en	R/W	0	<b>Per FIFO Enable</b> Set this field to 1 to enable the corresponding FIFO for the specified CNNx16_n quadrant. Each bit of this field maps to one of the CNNx16_n quadrants. Bit 3 maps to CNNx16_n_q3, Bit 2 maps to CNNx16_n_q2, Bit1 maps to CNNx16_n_q1, and Bit 0 maps to CNNx16_n_q0. <i>Note: Unused FIFOs must be disabled.</i>	
11	fifo_cpl	R/W	0	<b>FIFO Coupling</b> Setting this bit to 1 forces the FIFOs to operate in lockstep. Data available status is dependent on all FIFOs having identical write pointer values. 0: FIFOs are not coupled 1: FIFOs are coupled and operate in lockstep	
10	-	RO	0	<b>Reserved</b>	
9:7	empty_thresh	R/W	0	<b>FIFO Almost Empty Threshold</b> If the difference between the write and read pointer (read_pointer - write_pointer) falls below this number of bytes, the almost empty flag is set. 0 to 7: Sets the FIFO Almost Empty Threshold to the value written.	
6:5	-	RO	0	<b>Reserved</b>	
4:2	full_thresh	R/W	0	<b>FIFO Almost Full Threshold</b> This flag is set if the difference between the write and read pointer (read_pointer - write_pointer) exceeds this number of bytes, the almost full flag is set. 0 to 7: Sets the FIFO Almost Full Threshold to the value written.	

CNN FIFO Control			CNN_FIFO_CTRL		[0x0000]
Bits	Field	Access	Reset	Description	
1:0	rdy_sel	R/W	b11	<b>APB Wait State Selection</b> This field determines the number of wait states added during an APB access. 0b00: 0 wait states 0b01: 1 wait state 0b10: 2 wait states 0b11: 3 wait states <i>Note: Write operations load the data one clock before the end of the cycle.</i>	

Table 26-12: CNN FIFO Status Register

CNN FIFO Status			CNN_FIFO_STAT		[0x0004]
Bits	Field	Access	Reset	Description	
31:22	-	RO	0	<b>Reserved</b>	
21	rptr_eq	R	0	<b>FIFO Read Pointers Equal</b> This bit is set when all active FIFO read pointers are equal	
20	wptr_eq	R	0	<b>FIFO Write Pointers Equal</b> This bit is set when all active FIFO write pointers are equal	
19	fifos_almost_empty	R	0	<b>Aggregate FIFO Almost Empty Status</b> This field is a logical OR of individual FIFO almost empty statuses.	
18	fifos_almost_full	R	0	<b>Aggregate FIFO Almost Full Status</b> This field is a logical AND of individual FIFO almost full statuses.	
17	fifos_empty	R	0	<b>Aggregate FIFO Empty Status</b> This field is a logical OR of individual FIFO empty statuses.	
16	fifos_full	R	0	<b>Aggregate FIFO Full Status</b> This field is a logical AND of individual FIFO full statuses.	
15:12	almost_empty	R	0	<b>Per FIFO Almost Empty Status</b> If a bit in this field reads 1, the corresponding FIFO is almost empty. Each bit corresponds to a FIFO with almost_empty[0] mapped to CNNx16_n FIFO0. This status is persistent until the condition changes or until application firmware disables the FIFO.	
11:8	almost_full	R	0	<b>Per FIFO Almost Full Status</b> If a bit in this field reads 1, the corresponding FIFO is almost full. Each bit corresponds to a FIFO with almost_full[0] mapped to CNNx16_n FIFO0. This status is persistent until the condition changes or until application firmware disables the FIFO.	
7:4	empty	R	0	<b>Per FIFO Empty Status</b> If a bit in this field reads 1, the corresponding FIFO is empty. Each bit corresponds to a FIFO with empty[0] mapped to CNNx16_n FIFO0. This status is persistent until the condition changes or until application firmware disables the FIFO.	

CNN FIFO Status			CNN_FIFO_STAT		[0x0004]
Bits	Field	Access	Reset	Description	
3:0	full	R	0	<b>Per FIFO Full Status</b> If a bit in this field reads 1, the corresponding FIFO is full. Each bit corresponds to a FIFO with full[0] mapped to CNNx16_n FIFO0. The status is persistent until the condition changes, or until application firmware disables the FIFO.	

Table 26-13: CNN FIFO 0 Write Register

CNN FIFO 0 Write			CNN_FIFO_WR0		[0x0008]
Bits	Field	Access	Reset	Description	
31:0	data	R/W	0	<b>FIFO 0 Data</b> CNNx16_n FIFO 0 data register.	

Table 26-14: CNN FIFO 1 Write Register

CNN FIFO 1 Write			CNN_FIFO_WR1		[0x000C]
Bits	Field	Access	Reset	Description	
31:0	data	R/W	0	<b>FIFO 1 Data</b> CNNx16_n FIFO 2 data register.	

Table 26-15: CNN FIFO 2 Write Register

CNN FIFO 2 Write			CNN_FIFO_WR2		[0x0010]
Bits	Field	Access	Reset	Description	
31:0	data	R/W	0	<b>FIFO 2 Data</b> CNNx16_n FIFO 2 data register.	

Table 26-16: CNN FIFO 3 Write Register

CNN FIFO 3 Write			CNN_FIFO_WR3		[0x0014]
Bits	Field	Access	Reset	Description	
31:0	data	R/W	0	<b>FIFO 3 Data</b> CNNx16_n FIFO 3 data register.	

Table 26-17: CNN Always On Domain Control Register

CNN Always On Domain Control			CNN_AOD_CTRL		[0x1000]
Bits	Field	Access	Reset	Description	
31:20	-	RO	0	<b>Reserved</b>	
19:16	rm	R/W	0	<b>MRAM Read Margin MSB</b> Each bit of this field maps to one of the four CNNx16_n quadrants. Bit 0 maps to CNNx16_n_q0, Bit 1 maps to CNNx16_n_q1, bit 2 maps to CNNx16_n_q2 and bit 3 maps to CNNx16_n_q3.	

CNN Always On Domain Control			CNN_AOD_CTRL		[0x1000]
Bits	Field	Access	Reset	Description	
15:12	pd	R/W	0	<b>MRAM Power Down Enable</b> Each bit of this field maps to a CNNx16_n quadrant. Setting the quadrant's bit position to 1 shuts down power to the MRAM in the CNNx16_n quadrant. Write 0b1111 to this field to power down all CNNx16_n quadrants' MRAM. When a CNNx16_n quadrant's MRAM is put into power down mode, the contents of the MRAM are <i>not</i> maintained. 0: In any bit position, the corresponding CNNx16_n's MRAM is not in a power down state. 1: In any bit position, the corresponding CNNx16_n's MRAM is in a power down state. <i>Note: This field is independent of the Low Power Mode settings. Refer to <a href="#">Operating Modes</a> for more information on the system low power modes and their effect on the CNN and CNNx16_n quadrant memories.</i>	
11:8	dsleep	R/W	0	<b>MRAM Deep Sleep Enable</b> Each bit of this field maps to a CNNx16_n quadrant. Setting the quadrant's bit position to 1 puts the specified CNNx16_n's MRAM in a deep sleep state. Write 0b1111 to this field to put all four of the CNNx16_n quadrant's MRAM in deep sleep mode. When a CNNx16_n quadrant's MRAM is in deep sleep, the contents of the MRAM are retained, but the memory cannot be accessed. 0: In any bit position, the corresponding CNNx16_n's MRAM is not in deep sleep. 1: In any bit position, the corresponding CNNx16_n's MRAM is in deep sleep. <i>Note: This field is independent of the Low Power Mode settings. Refer to <a href="#">Operating Modes</a> for more information on the system's low-power modes and their effect on the CNN and CNNx16_n quadrant memories.</i>	
7:2	-	RO	0	<b>Reserved</b>	
1:0	rdy_sel	R/W	b11	<b>APB Wait State Selection</b> This field determines the number of wait states added during an APB access. 0: 0 wait states 1: 1 wait state 2: 2 wait states 3: 3 wait states <i>Note: Write operations load the data one clock before the end of the cycle.</i>	

## 26.5 CNNx16 Processor Array (CNNx16\_n) Registers

The CNN includes four  $\times 16$  Processor Arrays, referred to as CNNx16\_n. Each processor array includes a dedicated FIFO interface allowing configuration, status, and write access to each of the four CNNx16\_n's individual memory spaces. The table below shows the Base Address for each of the four CNNx16\_n processor arrays.

*Table 26-18: CNNx16\_n Instances and Base Offset Address*

Base Offset Address	Processor Array	Description
[0x0010 0000]	CNNx16_0	CNN $\times 16$ Processor Array 0
[0x0050 0000]	CNNx16_1	CNN $\times 16$ Processor Array 1
[0x0090 0000]	CNNx16_2	CNN $\times 16$ Processor Array 2
[0x00D0 0000]	CNNx16_3	CNN $\times 16$ Processor Array 3

### 26.5.1 CNNx16\_n Instances and Base Offset Addresses

*Table 26-18* shows the base address for each of the four CNNx16 processor arrays. Each CNNx16 processor array has its own independent set of registers shown in *Table 26-19*. The base address for a specific CNNx16\_n processor array is calculated by adding the CNNx16\_n base offset address to the global CNN base address shown in *Table 2-4*. Register names for a specific instance are defined by replacing “n” with the instance number. As an example, a register PERIPHERALn\_CTRL resolves to PERIPHERAL0\_CTRL and PERIPHERAL1\_CTRL for instances 0 and 1, respectively.

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

*Table 26-19: CNNx16\_n Processor Array Registers*

Offset	Register	Description
[0x0000]	<i>CNNx16_n_CTRL</i>	CNNx16_n Control Register
[0x0004]	<i>CNNx16_n_SRAM</i>	CNNx16_n SRAM Control Register
[0x0008]	<i>CNNx16_n_LCNT_MAX</i>	CNNx16_n Layer Count Maximum Register
[0x000C]	<i>CNNx16_n_TEST</i>	CNNx16_n SRAM Test Register
[0x0990]	<i>CNNx16_n_IFRM</i>	CNNx16_n Input FIFO Frame Size Register
[0x1000]	<i>CNNx16_n_MLAT</i>	CNNx16_n Muchselator Data Register
[0x0010]-[0x008C]	<i>CNNx16_n_Ly_RCNT</i>	CNNx16_n_Ly Row Count Register
[0x0090]-[0x010C]	<i>CNNx16_n_Ly_CCNT</i>	CNNx16_n_Ly Column Count Register
[0x0110]-[0x018C]	<i>CNNx16_n_Ly_ONED</i>	CNNx16_n_Ly One Dimensional Control Register
[0x0190]-[0x020C]	<i>CNNx16_n_Ly_PRCNT</i>	CNNx16_n_Ly Pool Row Count Register
[0x0210]-[0x028C]	<i>CNNx16_n_Ly_PCCNT</i>	CNNx16_n_Ly Pool Column Count Register
[0x0290]-[0x030C]	<i>CNNx16_n_Ly_STRIDE</i>	CNNx16_n_Ly Stride Count Register
[0x0310]-[0x038C]	<i>CNNx16_n_Ly_WPTR_BASE</i>	CNNx16_n_Ly Write Pointer Base Address Register
[0x0390]-[0x040C]	<i>CNNx16_n_Ly_WPTR_TOFF</i>	CNNx16_n_Ly Write Pointer Timeslot Offset Register
[0x0410]-[0x048C]	<i>CNNx16_n_Ly_WPTR_MOFF</i>	CNNx16_n_Ly Write Pointer Mask Offset Register
[0x0490]-[0x050C]	<i>CNNx16_n_Ly_WPTR_CHOFF</i>	CNNx16_n_Ly Write Pointer Multi-Pass Channel Offset Register
[0x0510]-[0x058C]	<i>CNNx16_n_Ly_RPTR_BASE</i>	CNNx16_n_Ly Read Pointer Base Address Register
[0x0590]-[0x060C]	<i>CNNx16_n_Ly_LCTRL0</i>	CNNx16_n_Ly Layer Control 0 Register
[0x0610]-[0x068C]	<i>CNNx16_n_Ly_MCNT</i>	CNNx16_n_Ly Mask Count Register
[0x0690]-[0x070C]	<i>CNNx16_n_Ly_TPTR</i>	CNNx16_n_Ly TRAM Pointer Register
[0x0710]-[0x078C]	<i>CNNx16_n_Ly_EN</i>	CNNx16_n_Ly Enable Register
[0x0790]-[0x080F]	<i>CNNx16_n_Ly_POST</i>	CNNx16_n_Ly Post Processing Register
[0x0A10]-[0x0A8C]	<i>CNNx16_n_Ly_LCTRL1</i>	CNNx16_n_Ly Layer Control 1 Register

Offset	Register	Description
[0x0810]-[0x082C]	<a href="#">CNNx16_n_Sz_STRMO</a>	CNNx16_n_Sz Stream Control 0 Register
[0x0890]-[0x08AC]	<a href="#">CNNx16_n_Sz_STRM1</a>	CNNx16_n_Sz Stream Control 1 Register
[0x0910]-[0x092C]	<a href="#">CNNx16_n_Sz_FBUF</a>	CNNx16_n_Sz Stream Frame Buffer Size Register

### 26.5.2 CNN Per x16 Processor Register Details

Table 26-20: CNNx16\_n Control Register

CNNx16_n Control			CNNx16_n_CTRL		[0x0000]
Bits	Field	Access	Reset	Description	
31	qupac	R/W	0	<b>QuPac Mode</b> The quad processors pack bit enables parallel processing of the same data by each of the 16 processors in the CNNx16_n. 0: Normal processing mode 1: x16 parallel processing mode <i>Note: FIFO mode must be enabled, CNNx16_n_CTRL.ffifoen set to 1, prior to enabling QuPac mode.</i>	
30	timeshft	R/W	0	<b>Pooling Stage Time Shift</b> When set, one wait state is added to the pooling stage to allow design time closure at a higher clock frequency.	
29:24	fclk_dly	R/W	0	<b>FIFO Clock Delay</b> This field selects the clock delay of the FIFO clock relative to the primary CNN clock. A setting of 0 adds in the minimum delay, and a value of 0x3F adds the maximum delay.	
23	fifo grp	R/W	0	<b>FIFO Group Output</b> Enables sending all "little data" channels to the first 4 processors. When this bit is not set, each byte of FIFO data is directed to the first little data channel of each CNNx16_n processor.	
22	ffifo_en	R/W	0	<b>Fast FIFO Enable</b> This field enables the tightly coupled data path between the MAX78000's CNN_TX fast FIFO and the CNN data SRAMs. The CNN_TX_FIFO is 32 bits wide, with 8 bits being dedicated to each of 4 channels. Channel routing is controlled by the state of the CNNx16_n_CTRL.ffifogrp control bit.	
21	simple1b	R/W	0	<b>Simple 1-Bit Weights</b> Enable simple logic for 1-bit weights. Setting this bit disables the wide accumulators used to calculate the convolution products and activates simple one-bit logic functions.	
20	mexpress	R/W	0	<b>Mask Memory Packed Memory Enable</b> Enable loading of the mask memories using packed data. With this bit set, a change in the state of the two least significant bits of the MRAM address triggers a reload of the address counter.	
19	lilbuf	R/W	0	<b>Stream Mode Circular Buffer Enable</b> Setting this bit restricts the associated read buffer's bounds to a circular buffer starting at the CNNx16_n_Ly_RPTR_BASE address and terminating at the CNNx16_n_Sz_FBUF address. When set, the circular buffer is used on all streaming layers.	

CNNx16_n Control			CNNx16_n_CTRL		[0x0000]
Bits	Field	Access	Reset	Description	
18:17	mlatch_sel	R/W	0	<b>SRAM Packed Channel Select</b> Selects which of the four channels in the SRAM read data is packed. 0: Selects channel 0, SRAM data bits 7:0 1: Selects channel 1 SRAM data bits 15:8 2: Selects channel 2 SRAM data bits 23:16 3: Select channel 3 SRAM data bits 31:24	
16	mlat_id	R/W	0	<b>Muchselator Load Data</b> Writing the bit from a 0 to a 1 forces the <a href="#">CNNx16_n_Ly_WPTR_BASE</a> address to be loaded into the Muchselator address counter and selects the counter as the SRAM address source. SRAM reads are only possible when <a href="#">CNNx16_n_CTRL.cnn_en</a> is reset to 0.	
15	fifo_en	R/W	0	<b>CNNx16_n_FIFO Enable</b> When set, data for the first (input) layer is taken from the CNN_FIFO_WRn FIFO register. One 4 byte-wide FIFO is dedicated for each of the four processor arrays. The FIFOs are accessed through the APB memory map, and each can be used in a single byte-wide channel mode, a single 4 byte-wide channel mode, or 4 single byte-wide channel mode. The mode is determined by the data configuration written to the FIFO through the APB, the <a href="#">CNNx16_n_CTRL.bigdata/CNNx16_n_LCTRLO.parallel</a> configuration, and the channel enables.	
14	stream_en	R/W	0	<b>Streaming Mode Enable</b> When set, the streaming mode is enabled for the CNNx16_n processor array. Streaming behavior is defined by the <a href="#">CNNx16_n Processor Stream Registers</a> Refer to <a href="#">Streaming Mode Configuration</a> for additional information. <i>Note: Unexpected behavior is likely when all four CNNx16_n processor arrays are not configured identically for streaming/non-streaming operation. Each CNN_x16_n processor array should be configured identically for streaming or non-streaming operation. Refer to <a href="#">Streaming Mode Configuration</a> for further details.</i>	
13	poolrnd	R/W	0	<b>Average Pooling Enable</b> When this bit is set, and average pooling is enabled, pooled values are rounded up for remainders greater or equal to 0.5 and down for remainders less than 0.5. 0: Average Pooling Disabled 1: Average Pooling Enabled	
12	cnn_irq	R/W	0	<b>CNN Interrupt Enable</b> This read/write bit indicates when the CNN interrupt request is active. It can be written to zero to reset the interrupt request. This interrupt signals the completion of CNN processing and should be masked in the MCU interrupt control logic if not required. It can also be written to 1 to force an interrupt. 0: CNN IRQ not active, write 0 to clear the CNN IRQ 1: CNN IRQ active, write 1 to generate a CNN IRQ	
11:9	ext_sync	R/W	0	<b>CNNx16_n External Sync Select</b> Each of these bits enable the external sync input from one of the CNN_x16_n processors. These bits allow the CNNx16_n processors to optionally operate in synchronization with one of the other CNNx16_n processors. In the general case, when all 64 processors are operating on a single convolution, CNNx16_0 processor 0 is selected by all four of the CNNx16_n's as the master byte setting ext_sync = 0b001. Combinations of processors can be configured as long as the groups are made up of sequential processors, without gaps.	

CNNx16_n Control				CNNx16_n_CTRL	[0x0000]
Bits	Field	Access	Reset	Description	
8	oneshot	R/W	0	<b>One Shot Layer Mode</b> With this bit set, only one layer is processed when the <a href="#">CNNx16_n_CTRL.cnn_en</a> bit is set. To advance to the next layer, the <a href="#">CNNx16_n_CTRL.cnn_en</a> bit must be reset to 0 and then set to 1. The low to high transition causes the CNN state machine to advance through the next layer. Memories can be interrogated between layers when <a href="#">CNNx16_n_CTRL.cnn_en</a> is 0. 0: One-shot layer mode disabled 1: One-shot layer mode enabled	
7	apbclk_en	R/W	0	<b>APB Clock Always On</b> Setting this bit forces the APB clock always on. When this bit is set to 0, clocks are only generated to the APB registers during write operations. 0: APB clocks are only on when APB register writes occur 1: APB clocks to the CNN APB registers is always on	
6	bigdata	R/W	0	<b>Big Data Enable</b> This bit globally selects the input data format that uses four data bytes per read for the groups of 4 processors. In other words, the four bytes allocated for a group of four processors is multiplexed to the first processor of the group.	
5	pool_en	R/W	0	<b>Pooling Enable</b> This bit globally enables pooling for all layers. 0: Global pooling disabled 1: Global pooling enabled for all layers. <i>Note: If this bit is set and the <a href="#">CNNx16_n_CTRL.calcmax</a> bit is not set, the per-layer <a href="#">CNNx16_n_Ly_LCTRL0.maxpl_en</a> bits are in effect.</i>	
4	calcmax	R/W	0	<b>Max Pooling Enable</b> This bit globally enables max pooling for all layers when the <a href="#">CNNx16_n_CTRL.pool_en</a> bit is set. <i>Note that this bit will be in effect, per layer, when the global <a href="#">CNNx16_n_CTRL.pool_en</a> bit is 0, and the per-layer <a href="#">CNNx16_n_Ly_LCTRL0.pool_en</a> bits are set.</i>	
3	clk_en	R/W	0	<b>Data Processing Clock Enable</b> Setting this bit enables the clocks to the . This field does not affect the clocks to the APB registers. Refer to the <a href="#">CNNx16_n_CTRL.apbclken</a> bit for the description of the APB clock behavior. 0: Clocks disabled to the Data Processing registers 1: Clocks enabled to the Data Processing registers	
2:1	rdy_sel	R/W	b11	<b>APB Wait State Selection</b> This field determines the number of wait states added during an APB access. 0b00: 0 wait states 0b01: 1 wait state 0b10: 2 wait states 0b11: 3 wait states <i>Note: Write operations load the data one clock before the end of the cycle.</i>	

CNNx16_n Control			CNNx16_n_CTRL		[0x0000]
Bits	Field	Access	Reset	Description	
0	cnn_en	R/W	0	<b>CNN Enable</b> Setting this bit to 1 initiates CNN processing. Processing is triggered by this field to 1. Firmware must set this field to 0 and back to 1 to perform subsequent CNN processing operations. 0: CNN processing stopped 1: Start CNN processing <i>Note: This field must be written to 0 before writing it to 1 for subsequent CNN processing to start.</i>	

Table 26-21: CNNx16\_n SRAM Control Register

CNNx16_n SRAM Control			CNNx16_n_SRAM		[0x0004]
Bits	Field	Access	Reset	Description	
22	lsram	R/W	0	<b>Bias Memory Light Sleep</b> Setting this bit forces the bias memory into light sleep when the bias memory is not selected by the CNN system or the APB.	
21	lstram	R/W	0	<b>TRAM Light Sleep</b> Setting this bit forces the TRAMs into light sleep when the TRAM is not selected by the CNN system or the APB.	
20	lsmram	R/W	0	<b>MRAM Light Sleep</b> Setting this bit forces the MRAMs into light sleep when the MRAM is not selected by the CNN system or the APB.	
19	lsdram	R/W	0	<b>Data RAM Light Sleep</b> Setting this bit forces the Data RAMs into light sleep when the SRAM is not selected by the CNN system or the APB.	
18:17	-	RO	0	<b>Reserved</b>	
16	pd	R/W	0	<b>CNNx16_n Memory Power Down Enable</b> Set this field to 1 to put the CNNx16_n's SRAM, TRAM, MRAM, and bias memory into a power-down mode. All memory contents are lost in power downstate. 0: CNNx16 memories are not powered down 1: Power down the CNNx16_n's memories	
15	ds	R/W	0	<b>CNNx16_n Memory Deep Sleep Enable</b> Set this field to 1 to put the CNNx16_n's SRAM, TRAM, MRAM, and bias memory into a deep sleep state. In deep sleep, the memories contents are retained, but peripheral logic is powered down, and the memory cannot be accessed. All memory outputs are set to output 0. 0: Memories are not in deep sleep state. 1: Put the CNNx16_n's memories into a deep sleep state. <i>Note: This field has no effect If the CNNx16_n_SRAM.pd field is set to 1 (memories powered down).</i>	
14	-	RO	0	<b>Reserved</b>	

CNNx16_n SRAM Control			CNNx16_n_SRAM		[0x0004]
Bits	Field	Access	Reset	Description	
13:11	wpulse	R/W	0	<b>Write Pulse Width</b> This field determines the bit line pulse width applied to the memory during writes. 0b000: Use the minimum bit line pulse width 0b111: Use the maximum bit line pulse width <i>Note: Values of wpulse between 0 and 7 incrementally set the bit line pulse width between the minimum and maximum values.</i>	
10	wneg_en			<b>Write Negative Voltage Enable</b> This bit enables the <a href="#">CNNx16_n_SRAM.wneg_vol</a> . If this field is 0, the system controls the negative voltage applied to the bit lines. 0: Write negative voltage time is controlled by the system. 1: Write negative voltage time is controlled by the setting in the <a href="#">CNNx16_n_SRAM.wneg_vol</a> field.	
9:8	wneg_vol	R/W	0	<b>Write Negative Voltage</b> This field sets the SRAM negative voltage level applied to the bit lines. This field is only used when the <a href="#">CNNx16_n_SRAM.wneg_en</a> field is set to 1. 0: V <sub>DD</sub> – 80mV 1: V <sub>DD</sub> – 120mV 2: V <sub>DD</sub> – 180mV 3: V <sub>DD</sub> – 220mV	
7:6	ra	R/W	0	<b>Read Assist Voltage</b> This field controls the Read Assist value for the SRAM bit lines. 0: V <sub>DD</sub> 1: V <sub>DD</sub> – 20mV 2: V <sub>DD</sub> – 40mV 3: V <sub>DD</sub> – 60mV These bits determine the WL underdrive (Read Assist) value. ra[1:0] = 00 limits the WL voltage to VDD, ra[1:0] = 01 limits the WL to VDD-20mV, ra[1:0] = 10 limits WL to VDD-40mV, and ra[1:0] = 11 limits the WL voltage to VDD-60mV.	
5:2	rmargin	R/W	b0011	<b>SRAM Read Margin</b> When <a href="#">CNNx16_n_SRAM.rm_en</a> is set, this field determines the length of the memory access time. 0b0000: Slowest SRAM access time 0b0001: 0b0010: 0b0011: Fastest SRAM access time (Reset Default) 0b0100-0b1111: Reserved <i>Note: The value of this field has no effect unless the <a href="#">CNNx16_n_CTRL.rm_en</a> field is set to 1.</i>	
1	rmargin_en	R/W	b1	<b>SRAM Read Margin Enable</b> Set this field to 1 to use the SRAM Access Time setting in the <a href="#">CNNx16_n_CTRL.ram_acc_time</a> field. 0: SRAM access time is set by the hardware 1: SRAM access time is controlled using the <a href="#">CNNx16_n_CTRL.ram_acc_time</a> field.	

CNNx16_n SRAM Control			CNNx16_n_SRAM		[0x0004]
Bits	Field	Access	Reset	Description	
0	extacc	R/W	0	<b>SRAM Extended Access Time Enable</b> Set this bit to 1 to enable SRAM access time maximum. Enabling longer SRAM access time increases the power consumption of the SRAM. 0: SRAM Power Optimized, reduced performance 1: SRAM Extended Access, higher power <i>Note, this setting can be used to extend access time but force the memories to consume additional power when active. This bit applies to all SRAMs in the CNNx16_n processor.</i>	

Table 26-22: CNNx16\_n Layer Count Maximum Register

CNNx16_n Layer Count Maximum			CNNx16_n_LCNT_MAX		[0x0008]
Bits	Field	Access	Reset	Description	
31:5	-	RO	0	<b>Reserved</b>	
4:0	lcnt	R/W	0	<b>Layer Count Maximum</b> Set this field to the maximum layer number for processing by the CNNx16_n. When the CNNx16_n is enabled, processing starts at layer 0 and completes processing at the layer number set by this field. 0-31: Set to the last layer for processing by the CNNx16_n <i>Note: The CNNx16_n must be inactive(<a href="#">CNNx16_n_CTRL.cnn_en=0</a>) when setting this field.</i>	

Table 26-23: CNNx16\_n SRAM Test Register

CNNx16_n SRAM Test			CNNx16_n_TEST		[0x000C]
Bits	Field	Access	Reset	Description	
28	zerodone	R	0	<b>BIST Zeroization Complete</b> This field is set to 1 by hardware when any of the zero run bits are set to 1 by firmware, and the hardware completes the zeroization. This bit is read only. Clear this bit by setting each of the zero run bits to 0. 1: BIST zeroization complete	
27	bistdone	R	0	<b>BIST Run Complete</b> This field is set to 1 by hardware when any of the BIST run bits are set to 1 by firmware, and the hardware completes the BIST operation. This bit is read only. Clear this bit by setting each of the BIST run bits to 0. 1: BIST operation complete	
26	bistfail	R	0	<b>BIST Run Failure Detected</b> This field is set to 1 by hardware if a BIST run operation was run and a BIST failure occurred. This bit is read only. Clear this bit by setting each of the BIST run bits to 0. 0: If the <a href="#">CNNx16_n_TEST.bistdone</a> bit reads 1, this bit indicates no BIST failures were detected. 1: BIST failure detected	

CNNx16_n SRAM Test			CNNx16_n_TEST		[0x000C]
Bits	Field	Access	Reset	Description	
25	ballzdone	R/W	0	<b>BRAM Zeroization Complete</b> This field indicates an SRAM zeroization is completed. This field is reset by hardware when firmware writes the <a href="#">CNNx16_n_TEST.bzerorun</a> field to 0. 1: BRAM zeroization complete	
24	tallzdone	R/W	0	<b>TRAM Zeroization Complete</b> This field indicates a TRAM zeroization is completed. This field is reset by hardware when firmware writes the <a href="#">CNNx16_n_TEST.tzerorun</a> field to 0. 1: TRAM zeroization complete	
23	mallzdone	R/W	0	<b>MRAM Zeroization Complete</b> This field indicates a MRAM zeroization is completed. This field is reset by hardware when firmware writes the <a href="#">CNNx16_n_TEST.mzerorun</a> field to 0. 1: MRAM zeroization complete	
22	sallzdone	R/W	0	<b>SRAM Zeroization Complete</b> This field indicates an SRAM zeroization is completed. This field is reset by hardware when firmware writes the <a href="#">CNNx16_n_TEST.szerorun</a> field to 0. 1: SRAM zeroization complete	
21	ballbdone	R/W	0	<b>BRAM BIST Complete</b> This field indicates a BRAM BIST run is completed. This field is reset by hardware when firmware writes the <a href="#">CNNx16_n_TEST.bbistrun</a> field to 0. 1: BRAM BIST complete	
20	tallbdone	R/W	0	<b>TRAM BIST Complete</b> This field indicates a TRAM BIST run is completed. This field is reset by hardware when firmware writes the <a href="#">CNNx16_n_TEST.tbistrun</a> field to 0. 1: TRAM BIST complete	
19	mallbdone	R/W	0	<b>MRAM BIST Complete</b> This field indicates a MRAM BIST run is completed. This field is reset by hardware when firmware writes the <a href="#">CNNx16_n_TEST.mbistrun</a> field to 0. 1: MRAM BIST complete	
18	sallbdone	R	0	<b>SRAM BIST Complete</b> This field indicates an SRAM BIST run is completed. This field is reset by hardware when firmware writes the <a href="#">CNNx16_n_TEST.sbistrun</a> field to 0. 1: SRAM BIST complete	
17	ballbfail	R	0	<b>BRAM BIST Result</b> When a BRAM BIST operation was started by setting <a href="#">CNNx16_n_TEST.bbistrun</a> bit to 1, and the operation is completed by hardware ( <a href="#">CNNx16_n_TEST.ballbdone</a> is set to 1 by hardware), this field indicates the result of the BRAM BIST operation. Reset this field by writing a 0 to the <a href="#">CNNx16_n_TEST.bbistrun</a> field. 0: BRAM BIST Passed 1: BRAM BIST Failed, indicating an error occurred in one of the BRAMs. <i>Note: This field is only valid after a BRAM BIST operation has started and completed (<a href="#">CNNx16_n_TEST.ballbdone</a> = 1).</i>	

CNNx16_n SRAM Test			CNNx16_n_TEST		[0x000C]
Bits	Field	Access	Reset	Description	
16	tallbfail	R/W	0	<b>SRAM BIST Result</b> When a TRAM BIST operation was started by setting <a href="#">CNNx16_n_TEST.tbistrun</a> bit to 1, and the operation is completed by hardware ( <a href="#">CNNx16_n_TEST.tallbdone</a> is set to 1 by hardware), this field indicates the result of the TRAM BIST operation. Reset this field by writing a 0 to the <a href="#">CNNx16_n_TEST.tbistrun</a> field. 0: TRAM BIST Passed 1: TRAM BIST Failed, indicating an error occurred in one of the four SRAMs. <i>Note: This field is only valid after a TRAM BIST operation has started and completes (<a href="#">CNNx16_n_TEST.tallbdone</a> = 1).</i>	
15	mallbfail	RO	0	<b>MRAM BIST Result</b> When a MRAM BIST operation was started by setting <a href="#">CNNx16_n_TEST.mbistrun</a> to one, and the operation is completed by hardware ( <a href="#">CNNx16_n_TEST.mallbdone</a> is set by hardware to 1), this field indicates the result of the SRAM BIST operation. Reset this field by writing a 0 to the <a href="#">CNNx16_n_TEST.mbistrun</a> field. 0: MRAM BIST Passed 1: MRAM BIST Failed, indicating an error occurred in one of the 16 MRAMs. <i>Note: This field is only valid after a MRAM BIST operation has started and completed (<a href="#">CNNx16_n_TEST.mallbdone</a> = 1).</i>	
14	sallbfail	RO	0	<b>SRAM BIST Result</b> When an SRAM BIST operation was started by setting <a href="#">CNNx16_n_TEST.sbistrun</a> to one, and the operation is completed by hardware ( <a href="#">CNNx16_n_TEST.sallbdone</a> is set by hardware to 1), this field indicates the result of the SRAM BIST operation. Reset this field by writing a 0 to the <a href="#">CNNx16_n_TEST.sbistrun</a> field 0: SRAM BIST Passed 1: SRAM BIST Failed, indicating an error occurred in one of the four SRAMs. <i>Note: This field is only valid after an SRAM BIST operation has started and completed (<a href="#">CNNx16_n_TEST.sallbdone</a> = 1).</i>	
13:8	bistsel	R/W	0	<b>BIST Controller Status Selection</b> The bits select an individual BIST controller status to be reported in the associated 32 bits of the memory read data bus. <a href="#">CNNx16_n_TEST.bistsel[5]</a> selects the SRAM or bias memory BIST group controller statuses, with: <a href="#">CNNx16_n_TEST.bistsel[2:0]</a> selecting the individual SRAM/bias memory instance with <a href="#">CNNx16_n_TEST.bistsel[2:0]</a> = 100 selecting the bias memory instance. Control bit <a href="#">CNNx16_n_TEST.bistsel[4]</a> selects the MRAM BIST controller statuses, with <a href="#">CNNx16_n_TEST.bistsel[3:0]</a> selecting the individual RAM instance. Control bit <a href="#">CNNx16_n_TEST.bistsel[3]</a> selects the TRAM BIST controller statuses, with <a href="#">CNNx16_n_TEST.bistsel[3:0]</a> selecting the individual RAM instance.	
7	bramz	R/W	0	<b>BIAS Memory Zeroize</b> Setting this bit to 1 will force the BIST to initialize all Bias memory locations to 0. Completion status can be found in the: <ul style="list-style-type: none"> <li>• <a href="#">CNNx16_n_TEST.sallzdone</a> and</li> <li>• <a href="#">CNNx16_n_TEST.zerodone</a></li> </ul> This bit must be written to 0 to reset the operation prior to writing it to 1. Writing 1 consecutively does not run the memory zeroization again.	

CNNx16_n SRAM Test			CNNx16_n_TEST		[0x000C]
Bits	Field	Access	Reset	Description	
6	bbistrun	R/W	0	<b>Run Bias Memory BIST</b> Setting this bit to 1 will run the BIST for all bias memory instances in the CNNx16_n processor. The BIST will run to completion, and the status is reported in: <ul style="list-style-type: none"> <li>• <a href="#">CNNx16_n_TEST.ballbdone</a></li> <li>• <a href="#">CNNx16_n_TEST.ballbfail</a></li> <li>• <a href="#">CNNx16_n_TEST.bistdone</a></li> <li>• <a href="#">CNNx16_n_TEST.bistfail</a></li> </ul> If more detailed status is required from the BIST execution, the <a href="#">CNNx16_n_TEST.bistsel</a> field can be used to extract status from an individual BIST controller. This bit must be written to 0 to reset the BIST operation prior to writing it to 1. Writing 1 consecutively does not run the BIST again.	
5	tramz	R/W	0	<b>TRAM Zeroize</b> Setting this bit to 1 will force the BIST to initialize all TRAM memory locations to 0. Completion status can be found in the <a href="#">CNNx16_n_TEST.tallzdone</a> and <a href="#">CNNx16_n_TEST.zerodone</a> status bit. This bit is edge-triggered and must be toggled from zero to one to run the BIST.	
4	tbistrun	R/W	0	<b>TRAM BIST Run</b> Setting this bit to 1 will run the BIST for all TRAM instances in the CNNx16_n processor. The BIST will run to completion and the status reported in: <ul style="list-style-type: none"> <li>• <a href="#">CNNx16_n_TEST.tallbdone</a></li> <li>• <a href="#">CNNx16_n_TEST.tallbfail</a></li> <li>• <a href="#">CNNx16_n_TEST.bistdone</a></li> <li>• <a href="#">CNNx16_n_TEST.bistfail</a></li> </ul> If more detailed status is required from the BIST execution, the <a href="#">CNNx16_n_TEST.bistsel</a> field can be used to extract status from an individual BIST controller. This bit must be written to 0 to reset the BIST operation prior to writing it to 1. Writing 1 consecutively does not run the BIST again.	
3	mramz	R/W	0	<b>MRAM Zeroize</b> Setting this bit to 1 will force the BIST to initialize all MRAM memory locations to 0. Completion status can be found in the: <ul style="list-style-type: none"> <li>• <a href="#">CNNx16_n_TEST.mallzdone</a> and</li> <li>• <a href="#">CNNx16_n_TEST.zerodon</a></li> </ul> This bit must be written to 0 to reset the operation prior to writing it to 1. Writing 1 consecutively does not run the memory zeroization again.	

CNNx16_n SRAM Test			CNNx16_n_TEST		[0x000C]
Bits	Field	Access	Reset	Description	
2	mbistrun	R/W	0	<b>MRAM BIST Run</b> Setting this bit to 1 will run the BIST for all MRAM instances in the CNNx16_n processor. The BIST will run to completion, and the status is reported in: <ul style="list-style-type: none"> <li>• <a href="#">CNNx16_n_TEST.mallbdone</a></li> <li>• <a href="#">CNNx16_n_TEST.mallbfail</a></li> <li>• <a href="#">CNNx16_n_TEST.bistdone</a></li> <li>• <a href="#">CNNx16_n_TEST.bistfail</a></li> </ul> If more detailed status is required from the BIST execution, the <a href="#">CNNx16_n_TEST.bistsel</a> field can be used to extract status from an individual BIST controller. This bit must be written to 0 to reset the BIST operation prior to writing it to 1. Writing 1 consecutively does not run the BIST again.	
1	sramz	R/W	0	<b>SRAM Zeroize</b> Setting this bit to 1 will force the BIST to initialize all SRAM memory locations to 0. Completion status can be found in the <ul style="list-style-type: none"> <li>• <a href="#">CNNx16_n_TEST.sallzdone</a> and</li> <li>• <a href="#">CNNx16_n_TEST.zerodone</a></li> </ul> This bit must be written to 0 to reset the operation prior to writing it to 1. Writing 1 consecutively does not run the memory zeroization again.	
0	sbistrun	R/W	0	<b>SRAM BIST Run</b> Setting this bit to 1 will run the BIST for all SRAM instances in the CNNx16_n processor. The BIST will run to completion, and the status is reported in: <ul style="list-style-type: none"> <li>• <a href="#">CNNx16_n_TEST.sallbdone</a></li> <li>• <a href="#">CNNx16_n_TEST.sallbfail</a></li> <li>• <a href="#">CNNx16_n_TEST.bistdone</a></li> <li>• <a href="#">CNNx16_n_TEST.bistfail</a></li> </ul> If more detailed status is required from the BIST execution, the <a href="#">CNNx16_n_TEST.bistsel</a> field can be used to extract status from an individual BIST controller. This bit must be written to 0 to reset the BIST operation prior to writing it to 1. Writing 1 consecutively does not run the BIST again.	

### 26.5.2.1 CNNx16\_n Per Layer Registers (CNNx16\_n\_Ly)

Each of the four CNNx16 Processor Arrays supports up to 32 layers. Each layer is controlled using the CNNx16\_n's Layer registers. Each layer includes one instance of each layer register. Each layer register is 32 bits wide (4 bytes). Register names for a given layer are defined by replacing "y" with the layer number ranging from 0 to 31 (32 layers maximum). The offset address of each layer's specific register is determined by the layer's base offset address and adding 4 times the layer number in hex. For example, for the CNNx16\_n\_Ly Row Count Register, the base offset address is [0x0010]. For layer 21, register CNNx16\_n\_L21\_RCNT, the address offset is [0x0010] + (4 × 0x15) = [0x0064].

Table 26-24: CNNx16\_n\_Ly Row Count Register

CNNx16_n_Ly Row Count			CNNx16_n_Ly_RCNT		[0x0010]-[0x008C]
Bits	Field	Access	Reset	Description	
31:18	-	RO	0	Reserved	

CNNx16_n_Ly Row Count			CNNx16_n_Ly_RCNT		[0x0010]-[0x008C]
Bits	Field	Access	Reset	Description	
17:16	rcnt_pad	R/W	0	<b>Pad Rows</b> This field sets the number of pad rows included at the beginning and end of the frame. <i>Note: The CNNx16_n_Ly_RCNT.rcnt_max is inclusive of two times this value, as the same number of pad rows are included at the beginning and end of the frame. This R/W register is accessible through the CNN APB interface.</i>	
15:10	-	RO	0	<b>Reserved</b>	
9:0	rcnt_max	R/W	0	<b>Layer Row Count Maximum</b> This field sets the maximum row count to be processed. Processing begins with row 0 and completes when the processing of the row determined by this field is complete. Set this field to include two times the CNNx16_n_Ly_RCNT.rcnt_pad value. $rcnt\_max = (2 \times rcnt\_pad) + \text{number of rows}$ <i>Note: The value programmed into this field is the effective image row value, including pad, but excluding rows not processed due to stride restrictions.</i>	

Table 26-25: CNNx16\_n\_Ly Column Count Register

CNNx16_n_Ly Column Count			CNNx16_n_Ly_CCNT		[0x0090]-[0x010C]
Bits	Field	Access	Reset	Description	
31:18	-	RO	0	<b>Reserved</b>	
17:16	ccnt_pad	R/W	0	<b>Layer Column Pad Count</b> This field determines the number of pad columns included at the beginning and end of the frame. Note that the CNNx16_n_Ly_CCNT.ccnt_max is inclusive of two times this value, as the same number of pad columns are included at the beginning and end of the row. This R/W register is accessible through the CNN APB interface.	
15:10	-	RO	0	<b>Reserved</b>	
9:0	ccnt_max	R/W	0	<b>Layer Column Count Maximum</b> When the CNN is enabled, these bits determine the maximum per layer column count to be processed. Processing begins with column 0 and completes when the processing of the column determined by CNNx16_n_Ly_CCNT.ccnt_max is complete. The value programmed into this register, through the CNN APB interface, is the effective image column value including pad, but excluding columns not processed due to stride restrictions.	

Table 26-26: CNNx16\_n\_Ly One Dimensional Control Register

CNNx16_n_Ly One Dimensional Control			CNNx16_n_Ly_ONED		[0x0110]-[0x018C]
Bits	Field	Access	Reset	Description	
31:22	-	RO	0	<b>Reserved</b>	

CNNx16_n_Ly One Dimensional Control			CNNx16_n_Ly_ONED		[0x0110]-[0x018C]
Bits	Field	Access	Reset	Description	
21:18	ewise_cnt	R/W	0	<b>Element-Wise Channel Count</b> Determines the number of element-wise channels to be processed. 0: 1 channel 1: 2 channels 2: 3 channels ... 14: 15 channels 15: 16 channels	
17	2d_conv	R/W	0	<b>2D Convolution of Element-Wise Result</b> Set this field to 1 to enable 2D convolution of the element-wise result. Standard 2D processing applies. 0: 2D convolution disabled 1: Enable 2D convolution	
16	prepool	R/W	0	<b>Pre-Pooling of Input Data</b> Set this field to 1 to enable pre-pooling of the input data prior to the element-wise operation selected with <a href="#">CNNx16_n_Ly_ONED.ewise_func</a> . 0: Input data is not pre-pooled prior to the element-wise function. 1: Pre-pooling of input data prior to element-wise operation enabled	
15:14	ewise_func	R/W	0	<b>Element-Wise Function Select</b> Selects the element-wise function performed on the input data if . Ob00: Subtract Ob01: Add Ob10: Bitwise OR Ob11: Bitwise XOR	
13	ewise_en	R/W	0	<b>Element-Wise Enable</b> Set this field to 1 to enable the element-wise operations. Prior to enabling element-wise operations, both the <a href="#">CNNx16_n_Ly_ONED.ts cnt_max</a> field and the <a href="#">CNNx16_n_Ly_POST.ts_en</a> fields must be set. 0: Element-wise operation disabled 1: Enable element-wise operation	
12	oned_en	R/W	0	<b>One Dimensional Processing Enable</b> Enables 1D input data processing. If the <a href="#">CNNx16_n_Ly_RCNT.rcnt_max</a> field is non-zero, the row count is used to index the input image; otherwise, the column count, <a href="#">CNNx16_n_Ly_CCNT.ccnt_max</a> , value is used.	
11:8	oned_width	R/W	0	<b>One Dimensional Convolution Mask Width</b> One dimensional convolution mask width (0-9 are valid values). One based value with a width > 0 enabling 1D convolution operation.	
7:4	oned_sad	R/W	0	<b>One Dimensional Convolution Start Mask Address</b> One dimensional convolution start mask address (offset within 9-byte mask width) used in conjunction with the mask start address ( <a href="#">CNNx16_n_Ly_MCNT.mc nt_sad</a> ) to determine that 1D convolution mask starting address.	
3:0	tscnt_max	R/W	0	<b>Maximum Time Slot Count</b> Maximum timeslot count register. When <a href="#">CNNx16_n_Ly_POST.tsen</a> is set, this value determines the number of timeslots required to output data that has been generated in parallel by the CNNx16_n processors. This count is used for pass-thru, 1x1, elementwise, and mask sharing ( <a href="#">CNNx16_n_Ly_LCTRL0.m slave</a> and <a href="#">CNNx16_n_Ly_LCTRL0.s slave</a> ) operations.	

Table 26-27: CNNx16\_n\_Ly Pool Row Count Register

CNNx16_n_Ly Pool Row Count			CNNx16_n_Ly_PRCNT		[0x0190]-[0x020C]
Bits	Field	Access	Reset	Description	
31:4	-	RO	0	<b>Reserved</b>	
3:0	prcnt_max	R/W	0	<b>Pool Row Count Max</b> When the CNN is enabled with one of the <a href="#">CNNx16_n_CTRL.pool_en</a> (global) or <a href="#">CNNx16_n_Ly_LCTRL0.pool_en</a> (per layer) bits set, this field determines the per layer pool row count to be processed. Processing begins with pool row 0 and completes when the processing of the pooling row determined by <i>prcnt_max</i> is complete, or the effective pool row count exceeds the image row count specified in <a href="#">CNNx16_n_Ly_RCNT.rcnt_max</a> . These count values are added to the row count to determine the effective address of the pooled data. This R/W register is accessible through the CNN APB interface.	

Table 26-28: CNNx16\_n\_Ly Pool Column Count Register

CNNx16_n_Ly Pool Column Count			CNNx16_n_Ly_PCCNT		[0x0210]-[0x028C]
Bits	Field	Access	Reset	Description	
31:4	-	RO	0	<b>Reserved</b>	
3:0	pccnt_max	R/W	0	<b>Pool Column Count Max</b> When the CNN is enabled with one of the <a href="#">CNNx16_n_CTRL.pool_en</a> (global) or <a href="#">CNNx16_n_Ly_LCTRL0.pool_en</a> (per layer) bits set, these bits determine the per layer pool column count to be processed. Processing begins with pool column 0 and completes when the processing of the pooling column determined by <i>pccnt_max</i> is complete, or the effective pool column count exceeds the image column count specified in <a href="#">CNNx16_n_Ly_CCNT.ccnt_max</a> . These count values are added to the column count to determine the effective address of the pooled data. This R/W register is accessible through the CNN APB interface.	

Table 26-29: CNNx16\_n\_Ly Stride Count Register

CNNx16_n_Ly Stride Count			CNNx16_n_Ly_STRIDE		[0x0290]-[0x030C]
Bits	Field	Access	Reset	Description	
31:2	-	RO	0	<b>Reserved</b>	
1:0	stride	R/W	0	<b>Stride</b> This field determines the stride length across and down the image. Processing begins with row 0 and column 0. A stride of one is applied until the top row or left column of the receptive field lands in the unpadded image. At that point, the stride value programmed through the APB interface is applied to the column and/or row of the field in the unpadded image. The stride is applied as long as the field remains within the bounds of the padded image.	

Table 26-30: CNNx16\_n\_Ly Write Pointer Base Address Register

CNNx16_n_Ly Write Pointer Base Address			CNNx16_n_Ly_WPTR_BASE		[0x0310]-[0x038C]
Bits	Field	Access	Reset	Description	
31:17	-	RO	0	<b>Reserved</b>	

CNNx16_n_Ly Write Pointer Base Address			CNNx16_n_Ly_WPTR_BASE		[0x0310]-[0x038C]
Bits	Field	Access	Reset	Description	
16:0	wptr_base	R/W	0	<b>Write Pointer Base</b> This per layer register sets the CNN convolution result SRAM write pointer base address. The base address can be set to any location in any data SRAM in the CNN. Bit 16 allows the write pointer to not point to any SRAM. This R/W register is accessible through the CNN APB interface.	

Table 26-31: CNNx16\_n\_Ly Write Pointer Timeslot Offset Register

CNNx16_n_Ly Write Pointer Timeslot Offset			CNNx16_n_Ly_WPTR_TOFF		[0x0390]-[0x040C]
Bits	Field	Access	Reset	Description	
31:17	-	RO	0	<b>Reserved</b>	
16:0	wptr_toff	R/W	0	<b>Write Pointer Timeslot Offset</b> When the CNN is enabled with the <a href="#">CNNx16_n_Ly_POST.ts_en</a> bit set and a non-zero time slot count value loaded into <a href="#">CNNx16_n_Ly_ONED.tscnt_max</a> field, this per layer register sets the CNN convolution result SRAM write pointer timeslot offset value. During a convolution result, SRAM data write, this timeslot offset value is multiplied by the timeslot counter and added to the SRAM write address pointer. This offset can be used to determine SRAM write addresses based on the timeslot. The offset can be set to any location in any data SRAM in the CNN. This R/W register is accessible through the CNN APB interface.	

Table 26-32: CNNx16\_n\_Ly Write Pointer Mask Offset Register

CNNx16_n_Ly Write Pointer Mask Offset			CNNx16_n_Ly_WPTR_MOFF		[0x0410]-[0x048C]
Bits	Field	Access	Reset	Description	
31:17	-	RO	0	<b>Reserved</b>	
16:0	wptr_moff	R/W	0	<b>Write Pointer Mask Offset</b> When the CNN is enabled with the <a href="#">CNNx16_n_Ly_EN.mask_en</a> bit set and a mask count value loaded into the <a href="#">CNNx16_n_Ly_MCNT.mcnt_max</a> register that is greater than the <a href="#">CNNx16_n_Ly_MCNT.mcnt_sad</a> value, this per layer register sets the CNN convolution result SRAM write pointer mask count offset value. During a convolution result, SRAM data write, this mask count offset value is multiplied by the mask counter and added to the SRAM write address pointer. This offset can be used to determine SRAM write addresses based on a mask count. The offset can be set to any location in any data SRAM in the CNN. This R/W register is accessible through the CNN APB interface.	

Table 26-33: CNNx16\_n\_Ly Write Pointer Multi-Pass Channel Offset Register

CNNx16_n_Ly Write Pointer Multi-Pass Channel Offset			CNNx16_n_Ly_WPTR_CHOFF		[0x0490]-[0x050C]
Bits	Field	Access	Reset	Description	
31:17	-	RO	0	<b>Reserved</b>	

CNNx16_n_Ly Write Pointer Multi-Pass Channel Offset				CNNx16_n_Ly_WPTR_CHOFF	[0x0490]-[0x050C]
Bits	Field	Access	Reset	Description	
16:0	wptr_choff	R/W	0	<b>Write Pointer Multi-Pass Offset</b> When the CNN is enabled with the <a href="#">CNNx16_n_Ly_EN.mask_en</a> bit set and a programmed maximum mask counter value ( <a href="#">CNNx16_n_Ly_MCNT.mcnt_max</a> - <a href="#">CNNx16_n_Ly_MCNT.mcnt_sad</a> ) that is greater than the maximum number of available processors programmed into the <a href="#">CNNx16_n_Ly_LCTRL1.xpch_max</a> register (output channel multi-pass is enabled), the rounded mask counter value is divided by the <a href="#">CNNx16_n_Ly_LCTRL1.xpch_max</a> value and multiplied by the <a href="#">CNNx16_n_Ly_WPTR_CHOFF.wptr_choff</a> to create a multi-pass channel offset value. During a convolution result SRAM data write, this offset can be used to determine SRAM write addresses based on a multi-pass count value. The offset can be set to any location in any data SRAM in the CNN. This R/W register is accessible through the CNN APB interface.	

Table 26-34: CNNx16\_n\_Ly Read Pointer Base Address Register

CNNx16_n_Ly Read Pointer Base Address			CNNx16_n_Ly_RPTR_BASE		[0x0510]-[0x058C]
Bits	Field	Access	Reset	Description	
31:17	-	RO	0	<b>Reserved</b>	
16:0	rptr_base	R/W	0	<b>Read Pointer Base Address</b> When the CNN is enabled, this per layer register sets the CNN convolution result SRAM read pointer base address. The base address can be set to any location in the SRAM dedicated to a specific CNN input channel processor. <i>Note: This field is limited to the 8192 Bytes of SRAM in the MAX78000. Do not write values greater than the memory available.</i> This R/W register is accessible through the CNN APB interface.	

Table 26-35: CNNx16\_n\_Ly Layer Control 0 Register

CNNx16_n_Ly Layer Control 0			CNNx16_n_Ly_LCTRL0		[0x0590]-[0x060C]
Bits	Field	Access	Reset	Description	
31:17	-	RO	0	<b>Reserved</b>	
16	bigdwrt	R/W	0	<b>Big Data Write</b> Enables writing out the current output channel in 32-bit accumulator form. This bit allows the full resolution of the output layer to be written to assist with software-defined SoftMax operation.	

CNNx16_n_Ly Layer Control 0				CNNx16_n_Ly_LCTRL0	[0x0590]-[0x060C]
Bits	Field	Access	Reset	Description	
15:12	cnnsi_en	R/W	0	<b>CNN 26-Bit Non-Scaled Non-quantized Sum of Products Feed</b> When set, this field enables the associated CNNx16_n 26-bit non-scaled, non-quantized sum of products data into the output accumulator, allowing sums of 16, 32, 48, or 64 products. Each bit is associated with one of the remaining 3 CNNx16_n processors. In processor 0, bit 1 is associated with CNNx16_n processor 1, bit 2 with CNNx16_n processor 2, and bit 3 with CNNx16_n processor 3. In processor 1, bit 1 is associated with CNNx16_n processor 2, bit 2 with CNNx16_n processor 3, and bit 3 with CNNx16_n processor 0, and so on. When these bits are set to zero, the internal state of the 26-bit data bus is forced to zero. 0b0000: 26-bit data bus set to 0 0b0001-0b1111: See description	
11	sramlsrc	R/W	0	<b>SRAM CNNx16_n SRAM Global Write Source Data</b> When set, SRAM data is sourced from the global data busses. The device supports 4 global data busses, one from each of the CNNx16_n processors. When all four CNNx16_n processors are used together to form a single sum-of-products value, all four CNNx16_n outputs an identical address, and based on the natural priority of the decode, processor 0 sources the SRAM write data.	
10	cpad_only	R/W	0	<b>Input Frame Column Pad</b> When set, padding is applied only to the input frame columns. In this case, row padding is ignored. This bit is intended to be used for parallel processing. <i>Note: When this field is set, row padding is ignored.</i>	
9	act_en	R/W	0	<b>ReLU Activation Eanble</b> When set, ReLU activation is enabled for each output channel. Activation is applied to the scaled and quantized data. 0: ReLU not enabled 1: ReLU activation enabled for each output channel and is applied to the scaled and quantized data.	
8	maxpl_en	R/W	0	<b>Max Pooling Enable</b> When set to 1, Max Pooling is selected as the pooling mode. In Max Pooling mode, the maximum value in the pool is selected for the field and written into the TRAM. When this field is set to 0, average pooling mode is selected. In Average Pooling Mode, the average of all pooled values is calculated and selected for use in the field. 0: Average pooling mode selected. 1: Max pooling mode selected <i>Note: This field is only used if the <a href="#">CNNx16_n_Ly_LCTRL0.pool_en</a> field is set to 1.</i>	
7	pool_en	R/W	0	<b>Enable Pooling</b> Setting this bit enables pooling for the associated layer. Pool dimensions are determined by the pool row and column count maximums ( <a href="#">CNNx16_n_Ly_PRCNT.prcnt_max</a> and <a href="#">CNNx16_n_Ly_PCCNT.pccnt_max</a> ). 0: Disabled 1: Enabled <i>Note: The type of pooling performed is determined by the <a href="#">CNNx16_n_Ly_LCTRL0.maxpl_en</a> field as either average pooling or max pooling.</i>	

CNNx16_n_Ly Layer Control 0			CNNx16_n_Ly_LCTRL0		[0x0590]-[0x060C]
Bits	Field	Access	Reset	Description	
6	parallel	R/W	0	<b>Parallel Mode Enable</b> Setting this bit to one enables a single input channel's data to use 4 bytes of memory instead of one. In parallel mode, data is read in byte order from byte 0 to byte 3 and then by memory depth. The purpose of the mode is to allow additional memory to be used for the input layer data to support larger images. When set, the receptive field for the data will be generated in the first processor in each group of 4 processors, provided the processor is enabled.	
5	master	R/W	0	<b>CNNx16_n Processor Group Master Select</b> Enables a CNNx16_n group of processors to independently calculate a sum-of-products result for all adjacent ascending CNNx16_n processor groups not configured for master operation.	
4	mslave	R/W	0	<b>CNNx16_n Processor 0 Mask Leader</b> When this bit is set, all 16 processors within the CNNx16_n share the receptive field with processor 0. This allows the generation of additional output channels using the mask values distributed across the 16 processors. Each processor applies a 3x3 mask of the selected width to be applied to the processor 0 field. <i>Note: Use of timeslots is required to write the parallel generated output channels to memory.</i>	
3:0	sslave	R/W	0	<b>CNNx16_n Processor Group Slave Mode</b> Within a CNNx16_n, this field enables each of the 4 groups of four processors to share the receptive field with the first processor of the x4 group (channels 0,4,8,12). When set, the receptive field of the first processor in the associated x4 group to be passed to the remaining 3 processors in the x4 group. Masks associated with the slaved group of three processors can be applied to the field of the first processor and additional output channels generated. Use of the timeslot counter is required to write the additional output channels to memory.	

Table 26-36: CNNx16\_n\_Ly Layer Mask Count Register

CNNx16_n_Ly Mask Count			CNNx16_n_Ly_MCNT		[0x0610]-[0x068C]
Bits	Field	Access	Reset	Description	
31:16	mcnt_sad	R/W	0	<b>Layer Mask SRAM start Address</b> Mask SRAM address counter increments sequentially from this word and bit address during layer processing. Counter restarts each stride and increments once per output channel: <ul style="list-style-type: none"> <li>• <math>mcnt\_sad[15:4]</math>→SRAM word (72bits) address</li> <li>• <math>mcnt\_sad[2:0]</math>→bit address</li> </ul>	
15:0	mcnt_max	R/W	0	<b>Mask SRAM Layer Maximum Address</b> Mask SRAM address counter increments sequentially by word and bit address during layer processing to this address. Counter restarts each stride and increments once per output channel: <ul style="list-style-type: none"> <li>• <math>mcnt\_max[15:4]</math>→SRAM word (72bits) address</li> <li>• <math>mcnt\_max[2:0]</math>→bit address.</li> </ul>	

Table 26-37: CNNx16\_n\_Ly TRAM Pointer Register

CNNx16_n_Ly TRAM Pointer			CNNx16_n_Ly_TPTR		[0x0690]-[0x070C]
Bits	Field	Access	Reset	Description	
31:27	-	RO	0	<b>Reserved</b>	
26:16	tptr_sad	R/W	0	<b>TRAM Start Address</b> This field determines the per layer TRAM pointer start address for initial processing and rollover events.	
15:11	-	RO	0	<b>Reserved</b>	
10:0	tptr_max	R/W	0	<b>TRAM Max Address</b> This field determines the rollover point of the TRAM address pointer. This field, <i>tptr_max</i> , is used together with the TRAM address pointer start address value ( <i>CNNx16_n_Ly_TPTR.tptr_sad</i> ) to reflect the usable input image row size, including pad.	

Table 26-38: CNNx16\_n\_Ly Enable Register

CNNx16_n_Ly Enable			CNNx16_n_Ly_EN		[0x0710]-[0x078C]
Bits	Field	Access	Reset	Description	
31:16	mask_en	R/W	0	<b>CNNx16_n Processor Mask Enable</b> Each bit in this per layer field enables the state of one processor's kernel logic in the CNNx16_n. Bit 0 controls processor 0's mask application (the master processor), and bit 15 controls processor 15's mask application.	
15:0	pro_en	R/W	0	<b>CNNx16_n Processor Enable</b> Each bit in this per layer field controls the enable state of one processor in the CNNx16_n. Bit 0 controls processor 0's (the master processor) enable and bit 15 controls processor 15's enable.	

Table 26-39: CNNx16\_n\_Ly Post Processing Register

CNNx16_n_Ly Post Processing			CNNx16_n_Ly_POST		[0x0790]-[0x080F]
Bits	Field	Access	Reset	Description	
31: 29	-	RO	0	<b>Reserved</b>	
28	deconv_en	R/W	0	<b>Deconvolution Enable</b> Virtually expands the input image size by adding a 0 byte after each actual input data byte is shifted into the TRAM, and a row of 0s following each row of column expanded input data is shifted into the TRAM. The receptive field remains at 3x3 scanned across the expanded data. 0: Deconvolution disabled 1: Deconvolution enabled	
27	flatten_en	R/W	0	<b>Flatten Enable</b> Enables flattening all of the input channel data supporting a series of 1x1 convolutions emulating a fully connected network. Setting this bit forces the use of an extended multi-pass count allowing for up to 256 neurons. This bit is used in conjunction with the <i>CNNx16_n_Ly_LCTRL1.inpcexp</i> , <i>CNNx16_n_Ly_POST.xpmp_cnt</i> , and <i>CNNx16_n_Ly_POST.onexone_en</i> fields to enable Multi-Layer Processing.	

CNNx16_n_Ly Post Processing			CNNx16_n_Ly_POST		[0x0790]-[0x080F]
Bits	Field	Access	Reset	Description	
26	out_abs	R/W	0	<b>Absolute Value Output Enable</b> Convert the scaled, quantized convolution output to an absolute value. This bit, along with the activation enable bit, <a href="#">CNNx16_n_Ly_LCTRL0.act_en</a> ) determine the final processing operation prior to writing the out channel to memory. 0: Output is not converted to an absolute value 1: Output is converted to an absolute value <i>Note: The <a href="#">CNNx16_n_Ly_POST.out_abs</a> has priority over the activation enable bit, <a href="#">CNNx16_n_Ly_LCTRL0.act_en</a>.</i>	
25	onexone_en	R/W	0	<b>Pass-Thru/1×1 Convolution Mode Enable</b> This bit forces all sixteen processors in the CNNx16_n to either directly pass-thru the result of the pooling logic or compute a 1 data byte by 1 byte (1,2,4,8 bit) weight product. Control of a pass-thru or 1×1 convolution is made for each of the 16 processors using the <a href="#">CNNx16_n_Ly_EN.mask_en</a> control field. 0: Pass-Thru Enabled 1: 1×1 Enabled	
24	ts_en	R/W	0	<b>Timeslot Mode Enable</b> This bit is used to enable the timeslot counter. When enabled, the number of timeslots programmed into the timeslot counter, <a href="#">CNNx16_n_Ly_ONED.tscount_max</a> , are added to each output channel slot. The timeslot counter allows pass-thru, 1×1, and elementwise values calculated in parallel to be written sequentially to memory. 0: Timeslot Mode Disabled 1: Timeslot Mode Enabled	
23:22	mask_size	R/W	0	<b>Mask Size Selection</b> This field determines the mask size multiplied with each 8-bit data value in the convolution. 0b00: 8-bits 0b01: 1-bit 0b10: 2-bits 0b11: 4-bits	
21:18	xpmp_cnt	R/W	0	<b>Expanded Multi-Pass (MP) Count</b> Adds 4 additional bits to the MP counter for flattening (MLP) operations. This field is only used when the <a href="#">CNNx16_n_Ly_POST.flatten_en</a> bit is set to 1. This field is appended to the <a href="#">CNNx16_n_Ly_LCTRL1.inpcchexp</a> bits and makes up the 4 most significant bits of the count.	
17	scale_shft	R/W	0	<b>Scale Shift Control</b> This bit selects the shift direction of the pre-activation sum-of-products result of the convolution. 0: Left Shift 1: Right Shift	
16:13	scale_reg	R/W	0	<b>Scale Shift Number</b> This field sets the number of bits to shift the pre-activation sum-of-products result of the convolution. Valid values are 0 to 16-bits, and the direction of the shift is controlled by <a href="#">CNNx16_n_Ly_POST.scale_shift</a> .	

CNNx16_n_Ly Post Processing			CNNx16_n_Ly_POST		[0x0790]-[0x080F]
Bits	Field	Access	Reset	Description	
12	bptr_en	R/W	0	<b>Bias Enable</b> This field enables the addition of a scaled bias, stored in each bias location, to the result of the convolution. Bias values are automatically scaled by a shift left of seven bits in hardware.	
11:0	bptr_sad	R/W	0	<b>Bias Pointer Start Address</b> Bias register file address byte counter increments once each mask count increment. <i>Note: The x16 Bias values can be enabled and used independently across the four output processors.</i>	

Table 26-40: CNNx16\_n\_Ly Layer Control 1 Register

CNNx16_n_Ly Layer Control 1			CNNx16_n_Ly_LCTRL1		[0x0A10]-[0x0A8C]
Bits	Field	Access	Reset	Description	
31:17	-	RO	0	<b>Reserved</b>	
16:8	xpch_max	R/W	0	<b>Expansion Mode Maximum Processors</b> This field selects the maximum channel processor number used in channel expansion mode. This allows for fewer than 64 processors to be used in a multi-pass or channel expansion configuration. <i>Note: Processor management was shown to be important for mask management when input channel processing draws on a relatively small number of channels for a potentially large number of masks.</i> 0-64: Selects the number of processors to use for channel expansion mode <i>Note: This field contains 9 bits; the upper 3 bits are reserved. Only values up to 64 are supported.</i>	
7:4	wptr_inc	R/W	0	<b>Write Pointer Increment</b> This field determines the increment for the write pointer counter after all output channels within a given stride are written to memory. Non-zero values in this field are used for multi-pass operations. <i>Note: The Write Pointer is always incremented by 1 or by 4 in parallel mode. The value in this field is added to the internal write pointer increment.</i>	
3:0	inpchexp	R/W	0	<b>Multi-Pass Input Channel Expansion Count</b> This field determines the number of sequential memory locations associated with a single convolution. For example, if a value of 15 (0xf) is programmed into this field, 16 channels are assumed to be sequentially stored in memory (channel then byte order) for each of the 64 processors. This allows for up to 1024 input channels to be processed in a single convolution. Similarly, if this field is set to 1, 2 channels are assumed to be sequentially stored in memory (channel then byte order) as channels for the convolution, totaling 128 channels if all 64 processors are used. A value of zero disables the multi-pass feature allowing for a single channel for each processor.	

Table 26-41: CNNx16\_n Muchselator Data Register

CNNx16_n Muchselator Data			CNNx16_n_MLAT		[0x1000]
Bits	Field	Access	Reset	Description	
31:0	mlatdat	RO	0	<b>Packed Channel Data</b> This register accumulates four bytes of output channel data to be read by the host MCU. Channel data is usually stored in the memory depth of a single byte of the SRAM data word, and four channels make up the memory data word. The Muchselator automatically fetches four bytes in the memory depth to generate a packed 4-byte word for efficient reading by the MCU. The target channel is selected using the <a href="#">CNNx16_n_CTRL.mlatch_sel</a> bits, and the target SRAM address is determined by the <a href="#">CNNx16_n_Ly_WPTR_BASE</a> register. Setting the <a href="#">CNNx16_n_CTRL.mlat_Id</a> bit to 1 loads the <a href="#">CNNx16_n_Ly_WPTR_BASE.wptr_base</a> value into the address counter and initiates the read of the first 4 bytes of channel data. When the current 4 bytes are accumulated in the <a href="#">CNNx16_n_MLAT.mlatdat</a> is read through the APB interface, the next 4 bytes are read in sequence. Reading continues until halted by the MCU. Four clock cycles are required after the completion of the last read or setting of the <a href="#">CNNx16_n_CTRL.mlat_Id</a> bit to 1 to accumulate the 4 bytes of data. It is up to the MCU firmware to ensure there is adequate time between reads to accumulate the 4 bytes of data.	

### 26.5.2.2 CNNx16\_n Processor Stream Registers

Each of the four CNNx16 Processor arrays supports up to 8 simultaneous streams for the first 8 layers. Each stream is controlled using the CNNx16\_n's Stream registers. Each stream includes one instance of each Stream register. Each Stream register is 32 bits wide (4 bytes). Register names for a given stream are defined by replacing "z" with the stream number ranging from 0 to 7 (8 streams maximum). The offset address of each stream's specific register is determined by the stream's base offset address and adding 4 times the stream number in hex. For example, for the CNNx16\_n\_Sz Stream Control 0 Register, the base offset address is [0x0810]. For Stream 5, register CNNx16\_n\_S5\_STRM0, the address offset is [0x0810] + (4 × 0x05) = [0x0824].

Table 26-42: CNNx16\_n\_Sz Stream Control 0 Register

CNNx16_n_Sz Stream Control 0			CNNx16_n_Sz_STRM0		[0x0810]-[0x082C]
Bits	Field	Access	Reset	Description	
31:14	-	RO	0	<b>Reserved</b>	
13:0	strm_isval	R/W	0	<b>Per-Stream Start Count</b> The per-stream start count is based on the previous layer's .tptr_inc count. When streaming is enabled ( <a href="#">CNNx16_n_CTRL.stream_en</a> = 1), each time a byte in the prior or input layer is written into the TRAM, the internal stream start counter is incremented. When the counter reaches this field's value, <a href="#">CNNx16_n_Sz_STRM0.strm_isval</a> , processing of the current layer is enabled. This mechanism allows adequate receptive field data to be accumulated before convolution processing in a stream layer begins. Once the counter value reaches this field's value, <a href="#">CNNx16_n_Sz_STRM0.strm_isval</a> , counting is halted, and the terminal count value remains in the counter. The R/W register is accessible through the APB interface.	

Table 26-43: CNNx16\_n\_Sz Stream Control 1 Register

CNNx16_n_Sz Stream Control 1			CNNx16_n_Sz_STRM1		[0x0890]-[0x08AC]
Bits	Field	Access	Reset	Description	
31:28	-	RO	0	<b>Reserved</b>	

CNNx16_n_Sz Stream Control 1			CNNx16_n_Sz_STRM1		[0x0890]-[0x08AC]
Bits	Field	Access	Reset	Description	
27:16	strm_dsval2	R/W	0	<b>Per Stream Multi-Row Delta Count</b> This field is based on the previous layer's <a href="#">CNNx16_n_Ly_TPTR</a> count. When streaming is enabled ( <a href="#">CNNx16_n_CTRL.stream_en = 1</a> ), this field determines the number of bytes written into the TRAM of the prior layer between active processing of the current layer. This APB accessible R/W register is used to set the processing cadence across an image row boundary. When the internal delta count counter reaches the value stored in this field, <a href="#">CNNx16_n_Sz_STRM1.strm_dsval2</a> , processing of the current layer is enabled for one stride (input pooling plus output channel generation), and the delta counter is reset to zero.	
15:9	-	RO	0	<b>Reserved</b>	
8:4	strm_dsval1	R/W	0	<b>Per Stream In-Row Delta Count</b> This field is based on the previous layer's tptr_inc count. When streaming is enabled ( <a href="#">CNNx16_n_CTRL.stream_en = 1</a> ), this field determines the number of bytes written into the TRAM of the prior layer between active processing of the current layer. This APB accessible R/W register is used to set the processing cadence of each column across an image row. When the internal delta count counter reaches the value stored in this field, <a href="#">CNNx16_n_Sz_STRM1.strm_dsval1</a> , processing of the current layer is enabled for one stride (input pooling plus output channel generation), and the delta counter is reset to zero.	
3:0	strm_invol	R/W	0	<b>Per Stream Input Volume Offset</b> This field is based on the stream count. When streaming is enabled ( <a href="#">CNNx16_n_CTRL.stream_en = 1</a> ), this field determines the input volume offset applied to each stream. The value programmed into this field is multiplied by the stream count to calculate the stream count input volume offset.  The CNN supports up to 16 independent input volumes. The input volumes are split between multi-pass and streaming. The streaming input volume offset allows the streaming input volume selection to "skip over" the input volumes used for multi-pass processing.	

Table 26-44: CNNx16\_n\_Sz Stream Frame Buffer Size Register

CNNx16_n_Sz Stream Frame Buffer Size			CNNx16_n_Sz_FBUF		[0x0910]-[0x092C]
Bits	Field	Access	Reset	Description	
31:17	-	RO	0	<b>Reserved</b>	
16:0	fbuf_max	R/W	0	<b>Per Stream Circular Buffer Max Count</b> When streaming is enabled ( <a href="#">CNNx16_n_CTRL.stream_en = 1</a> ), the per-layer SRAM read pointer base register value ( <a href="#">CNNx16_n_Ly_RPTR_BASE.rptr_base</a> ) is subtracted from the internally generated read pointer counter and compared to the value stored in this field. When the adjusted read pointer is equal to this field, the rollover point is detected, and the <a href="#">CNN_FIFO_STAT.rptr_base</a> value is loaded into the read pointer counter. The R/W register is accessible through the APB interface.	

Table 26-45: CNNx16\_n Input FIFO Frame Size Register

CNNx16_n Input FIFO Frame Size			CNNx16_n_IFRM		[0x0990]
Bits	Field	Access	Reset	Description	
31:17	-	RO	0	Reserved	
16:0	ifrm_reg	R/W	0	<b>Streaming Input Frame Size Byte Count</b> When streaming is enabled ( <a href="#">CNNx16_n_CTRL.stream_en = 1</a> ), this field determines the number of bytes read from the data FIFOs. An internal counter counts the number of input frame bytes read from the input FIFOs and compares the count to the value in this register. Once the value in this field is reached, the terminal count value is retained, incrementing of this counter is inhibited, and processing of the input data is terminated. This R/W register is accessible through the APB interface.	

## 27. Revision History

Table 27-1: Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
0	03/09/2021	Initial release	-
0.1	05/21/2021	Updated DMA chapter: Added PCIF RX channel Updated DMA Interrupt Enable and Interrupt Flag registers Updated Figure 9-1 to address typos and incorrect decision labels in last decision tree. Updated section 9.5 to show the proper register for configuring a channel for chaining ( <a href="#">DMA_CHn_DSTRLD</a> ). Changed all DMAm to DMA as there is only one DMA engine in this device.	133-141

Preliminary Draft 05/21/2021

©2021 by Maxim Integrated Products, Inc. All rights reserved. Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. MAXIM INTEGRATED PRODUCTS, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. MAXIM ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering or registered trademarks of Maxim Integrated Products, Inc. All other product or service names are the property of their respective owners.