Abstract: This user guide provides application developers information on how to use the memory and peripherals of the MAX32665–MAX32668 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.
# Table of Contents

1. Overview
   1.1 Block Diagram

2. Resource Protection Unit (RPU)
   2.1 Instances
   2.2 Usage
      2.2.1 Reset State
      2.2.2 MPU Implementation
      2.2.3 MPU Protection Fault
      2.2.4 RPU Protection Fault
      2.2.5 RPU Fault Handler
   2.3 Registers
   2.4 Register Details

3. Memory, Register Mapping, and Access
   3.1 Memory, Register Mapping, and Access Overview
   3.2 Standard Memory Regions
      3.2.1 Code Space
      3.2.2 SRAM Space
      3.2.3 Peripheral Space
      3.2.4 External RAM Space
      3.2.5 External Device Space
      3.2.6 System Area (Private Peripheral Bus)
      3.2.7 System Area (Vendor Defined)
   3.3 Device Memory Instances
      3.3.1 Main Program Flash Memory
      3.3.2 Cache Memories
      3.3.3 External Memory Cache Controller (EMCC)
      3.3.4 Information Block Flash Memory
      3.3.5 System SRAM
      3.3.6 AES Key and Working Space Memory
      3.3.7 MAA Key and Working Space Memory
      3.3.8 TPU Memory
   3.4 AHB Interfaces
3.4.1 Core AHB Interfaces
3.4.2 AHB Masters
3.5 Peripheral Register Map
3.5.1 APB Peripheral Base Address Map
3.5.2 AHB Peripheral Base Address Map
3.6 Error Correction Coding (ECC) Module
3.6.1 SRAM
3.6.2 FLASH
3.6.3 Cache
3.6.4 Limitations
4. System, Power, Clocks, Reset
4.1 Oscillator Sources and Clock Switching
4.1.1 Oscillator Implementation
4.1.2 96MHz Internal Main High-Speed Oscillator
4.1.3 60MHz Low Power Internal Oscillator
4.1.4 32MHz Bluetooth Radio Oscillator
4.1.5 7.3728MHz Internal Oscillator
4.1.6 32.768kHz External Crystal Oscillator
4.1.7 8kHz Ultra-Low Power Nano-Ring Internal Oscillator
4.2 Operating Modes
4.2.1 ACTIVE Mode
4.2.2 SLEEP Low Power Mode
4.2.3 DEEPSLEEP Low Power Mode
4.2.4 BACKUP Low Power Mode
4.3 Device Resets
4.3.1 Peripheral Reset
4.3.2 Soft Reset
4.3.3 System Reset
4.3.4 Power-On Reset
4.4 Cache
4.5 Instruction Cache Controller
4.5.1 Enabling ICC0/ICC1/SFCC
4.5.2 Flushing the ICC0/ICC1/SFCC Cache
4.5.3 Flushing SRCC Cache
4.6 Instruction Cache Controller Registers
4.7 External RAM SPIXR Cache Controller (SRCC)
4.8 RAM Memory Management ................................................................. 67
  4.8.1 RAM Zeroization ........................................................................... 67
  4.8.2 RAM Low Power Modes .............................................................. 67
4.9 Miscellaneous Control Registers .................................................. 68
4.10 Miscellaneous Control Registers Details .................................. 68
4.11 Single Inductor Multiple Output (SIMO) Power Supply ............ 71
  4.11.1 Power Supply Monitor ............................................................... 71
4.12 Single Inductor Multiple Output (SIMO) Registers .................. 72
4.13 Single Inductor Multiple Output (SIMO) Registers Details ....... 73
4.14 Power Sequencer and Always-On Domain Registers ............. 78
4.15 Power Sequencer and Always-On Domain Register Details .... 79
4.16 Global Control Registers (GCR) ...................................................... 85
4.17 Global Control Register Details (GCR) ......................................... 86
4.18 Function Control Registers ............................................................. 113
4.19 Function Control Register Details ................................................ 113
4.20 AES Key Registers ........................................................................ 114
4.21 AES Key Register Details ............................................................... 114

5. Interrupts and Exceptions ............................................................... 115
  5.1 Features ......................................................................................... 115
  5.2 Interrupt Vector Table ................................................................. 115

6. General-Purpose I/O and Alternate Function Pins (GPIO) .......... 119
  6.1 Instances ..................................................................................... 119
  6.2 Usage ......................................................................................... 123
    6.2.1 Reset State ............................................................................ 123
    6.2.2 Input Mode Configuration .................................................... 123
    6.2.3 Output Mode Configuration .................................................. 123
    6.2.4 Alternate Function Configuration ........................................ 123
  6.3 Configuring GPIO (External) Interrupts .................................... 123
    6.3.1 GPIO Interrupt Handling ...................................................... 124
    6.3.2 Using GPIO for Wakeup from Low Power Modes ............... 124
  6.4 Registers ................................................................................... 125
  6.5 Register Details ........................................................................... 126

7. Flash Controller (FLC) ................................................................. 134
  7.1 Instances ................................................................................... 134
  7.2 Usage ......................................................................................... 134
7.2.1 Clock Configuration

7.2.2 Lock Protection

7.2.3 Flash Write Width

7.2.4 Flash Write

7.2.5 Page Erase

7.2.6 Mass Erase

7.3 Flash Error Correction Coding

7.4 Flash Controller Registers

7.5 Flash Controller Register Details

8. External Memory

8.1 Overview

8.2 SPI Execute-in-Place Flash (SPIXF)

8.2.1 SPIXF Master Controller

8.2.2 SPIXF Master

8.3 SPI Execute-in-Place RAM (SPIXR)

8.3.1 SPIXR Master Controller Registers

8.3.2 SPIXR Register Details

8.4 SPIXR Cache Controller (SRCC)

8.4.1 Features

8.4.2 Enabling the SRCC

8.4.3 Disabling the SRCC

8.4.4 SRCC Registers

8.4.5 SRCC Register Details

8.5 Secure Digital Host Controller

8.5.1 Instances

8.5.2 SDHC Peripheral Clock Selection

8.5.3 Usage

8.5.4 SD Command Generation

8.5.5 SDHC Registers

8.5.6 SDHC Register Details

9. Standard DMA (DMAC)

9.1 Instances

9.2 DMA Channel Operation (DMACH)

9.2.1 DMA Channel Arbitration and DMA Bursts

9.2.2 DMA Source and Destination Addressing

Data Movement From Source to DMA
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.2.3</td>
<td>Data Movement From the DMA to Destination</td>
<td>226</td>
</tr>
<tr>
<td>9.3</td>
<td>Usage</td>
<td>227</td>
</tr>
<tr>
<td>9.4</td>
<td>Count-To-Zero (CTZ) Condition</td>
<td>227</td>
</tr>
<tr>
<td>9.5</td>
<td>Chaining Buffers</td>
<td>228</td>
</tr>
<tr>
<td>9.6</td>
<td>DMA Interrupts</td>
<td>230</td>
</tr>
<tr>
<td>9.7</td>
<td>Channel Timeout Detect</td>
<td>230</td>
</tr>
<tr>
<td>9.8</td>
<td>Memory-to-Memory DMA</td>
<td>231</td>
</tr>
<tr>
<td>9.9</td>
<td>DMAC Registers</td>
<td>231</td>
</tr>
<tr>
<td>9.10</td>
<td>DMAC Register Details</td>
<td>231</td>
</tr>
<tr>
<td>9.11</td>
<td>DMA Channel Registers</td>
<td>232</td>
</tr>
<tr>
<td>9.12</td>
<td>DMAC Channel Registers</td>
<td>232</td>
</tr>
<tr>
<td>9.13</td>
<td>DMA Channel Register Details</td>
<td>233</td>
</tr>
<tr>
<td>10.</td>
<td>Cyclic Redundancy Check Engine (CRC)</td>
<td>238</td>
</tr>
<tr>
<td>10.1</td>
<td>Instances</td>
<td>239</td>
</tr>
<tr>
<td>10.2</td>
<td>Linear Feedback Shift Register (LFSR)</td>
<td>239</td>
</tr>
<tr>
<td>10.3</td>
<td>Registers</td>
<td>240</td>
</tr>
<tr>
<td>10.4</td>
<td>Register Details</td>
<td>241</td>
</tr>
<tr>
<td>11.</td>
<td>Analog to Digital Converter and Comparators (ADC)</td>
<td>246</td>
</tr>
<tr>
<td>11.1</td>
<td>Features</td>
<td>246</td>
</tr>
<tr>
<td>11.2</td>
<td>Instances</td>
<td>246</td>
</tr>
<tr>
<td>11.3</td>
<td>Architecture</td>
<td>247</td>
</tr>
<tr>
<td>11.4</td>
<td>Clock Configuration</td>
<td>249</td>
</tr>
<tr>
<td>11.5</td>
<td>Power-Up Sequence</td>
<td>250</td>
</tr>
<tr>
<td>11.6</td>
<td>Conversion</td>
<td>250</td>
</tr>
<tr>
<td>11.7</td>
<td>Reference Scaling and Input Scaling</td>
<td>250</td>
</tr>
<tr>
<td>11.7.1</td>
<td>AIN0 – AIN7 Scale Limitations</td>
<td>251</td>
</tr>
<tr>
<td>11.7.2</td>
<td>Scale Limitations for All Other Input Channels</td>
<td>251</td>
</tr>
<tr>
<td>11.7.3</td>
<td>Data Conversion Output Alignment</td>
<td>251</td>
</tr>
<tr>
<td>11.7.4</td>
<td>Data Conversion Value Equations</td>
<td>252</td>
</tr>
<tr>
<td>11.7.5</td>
<td>Data Limits and Out of Range Interrupts</td>
<td>253</td>
</tr>
<tr>
<td>11.7.6</td>
<td>Power-Down Sequence</td>
<td>254</td>
</tr>
<tr>
<td>11.8</td>
<td>Comparator Operation</td>
<td>254</td>
</tr>
<tr>
<td>11.9</td>
<td>Registers</td>
<td>255</td>
</tr>
<tr>
<td>11.10</td>
<td>Register Details</td>
<td>255</td>
</tr>
<tr>
<td>12.</td>
<td>UART</td>
<td>260</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>12.1 Instances</td>
<td>260</td>
<td></td>
</tr>
<tr>
<td>12.2 UART Frame</td>
<td>260</td>
<td></td>
</tr>
<tr>
<td>12.3 UART Interrupts</td>
<td>261</td>
<td></td>
</tr>
<tr>
<td>12.4 UART Baud Rate Clock Source</td>
<td>261</td>
<td></td>
</tr>
<tr>
<td>12.5 UART Baud Rate Calculation</td>
<td>261</td>
<td></td>
</tr>
<tr>
<td>12.6 UART Configuration and Operation</td>
<td>263</td>
<td></td>
</tr>
<tr>
<td>12.7 Wakeup Time</td>
<td>263</td>
<td></td>
</tr>
<tr>
<td>12.8 Hardware Flow Control</td>
<td>263</td>
<td></td>
</tr>
<tr>
<td>12.9 Registers</td>
<td>263</td>
<td></td>
</tr>
<tr>
<td>12.10 Register Details</td>
<td>264</td>
<td></td>
</tr>
<tr>
<td>13. I²C Master/Slave Serial Communications Peripheral (I2C)</td>
<td>272</td>
<td></td>
</tr>
<tr>
<td>13.1 I²C Master/Slave Features</td>
<td>274</td>
<td></td>
</tr>
<tr>
<td>13.2 Instances</td>
<td>274</td>
<td></td>
</tr>
<tr>
<td>13.3 I²C Overview</td>
<td>275</td>
<td></td>
</tr>
<tr>
<td>13.3.1 I²C Bus Terminology</td>
<td>275</td>
<td></td>
</tr>
<tr>
<td>13.3.2 I²C Transfer Protocol Operation</td>
<td>275</td>
<td></td>
</tr>
<tr>
<td>13.3.3 START and STOP Conditions</td>
<td>275</td>
<td></td>
</tr>
<tr>
<td>13.3.4 Master Operation</td>
<td>275</td>
<td></td>
</tr>
<tr>
<td>13.3.5 Acknowledge and Not Acknowledge</td>
<td>275</td>
<td></td>
</tr>
<tr>
<td>13.3.6 Bit Transfer Process</td>
<td>276</td>
<td></td>
</tr>
<tr>
<td>13.4 I²C Configuration and Usage</td>
<td>277</td>
<td></td>
</tr>
<tr>
<td>13.4.1 SCL and SDA Bus Drivers</td>
<td>277</td>
<td></td>
</tr>
<tr>
<td>13.4.2 SCL Clock Configurations</td>
<td>277</td>
<td></td>
</tr>
<tr>
<td>13.4.3 SCL Clock Generation for Standard, Fast and Fast-Plus Modes</td>
<td>277</td>
<td></td>
</tr>
<tr>
<td>13.4.4 SCL Clock Generation for Hs-mode</td>
<td>278</td>
<td></td>
</tr>
<tr>
<td>13.4.5 I²C Addressing</td>
<td>279</td>
<td></td>
</tr>
<tr>
<td>13.4.6 I²C Master Mode Operation</td>
<td>280</td>
<td></td>
</tr>
<tr>
<td>13.4.7 I²C Slave Mode Operation</td>
<td>283</td>
<td></td>
</tr>
<tr>
<td>13.4.8 I²C Interrupt Sources</td>
<td>288</td>
<td></td>
</tr>
<tr>
<td>13.4.9 TX FIFO and RX FIFO</td>
<td>288</td>
<td></td>
</tr>
<tr>
<td>13.4.10 TX FIFO Preloading</td>
<td>289</td>
<td></td>
</tr>
<tr>
<td>13.4.11 Interactive Receive Mode (IRXM)</td>
<td>290</td>
<td></td>
</tr>
<tr>
<td>13.4.12 Clock Stretching</td>
<td>291</td>
<td></td>
</tr>
<tr>
<td>13.4.13 I²C Bus Timeout</td>
<td>291</td>
<td></td>
</tr>
<tr>
<td>13.4.14 I²C DMA Control</td>
<td>292</td>
<td></td>
</tr>
<tr>
<td>13.5 Registers</td>
<td>292</td>
<td></td>
</tr>
</tbody>
</table>
13.6 Register Details

14. Quad Serial Peripheral Interface (SPI)

14.1 Instances

14.2 SPI Formats

14.2.1 Four-Wire SPI

14.2.2 Three-Wire SPI

14.3 Pin Configuration

14.3.1 QSPln Alternate Function Mapping

14.3.2 Four-wire Format Configuration

14.3.3 Three-Wire Format Configuration

14.3.4 Dual Mode Format Configuration

14.3.5 Quad Mode Format Pin Configuration

14.4 QSPln Clock Configuration

14.4.1 Serial Clock

14.4.2 SPI Peripheral Clock

14.4.3 Master Mode Serial Clock Generation

14.4.4 Clock Phase and Polarity Control

14.4.5 QSPln FIFOs

14.4.6 QSPln Interrupts and Wakeups

14.5 Registers

14.6 Register Details

15. HTimer (HT)

15.1 Overview

15.2 Alarm Functions

15.2.1 Long-Interval Alarm

15.2.2 Short-Interval Alarm

15.3 Register Access Control

15.3.1 Register Write Protection

15.3.2 Register Read Protection

15.3.3 Count Register Access

15.3.4 Alarm Register Access

15.4 Registers

15.5 Register Details

16. Timers

16.1 Features

16.2 Basic Operation
16.3 Timer Pin Functionality
16.4 One-Shot Mode (000b)
  16.4.1 One-Shot Mode Timer Period
  16.4.2 One-Shot Mode Configuration
16.5 Continuous Mode (001b)
  16.5.1 Continuous Mode Timer Period
  16.5.2 Continuous Mode Configuration
16.6 Counter Mode (010b)
  16.6.1 Counter Mode Timer Period
  16.6.2 Counter Mode Configuration
16.7 PWM Mode (011b)
  16.7.1 PWM Mode Timer Period
  16.7.2 PWM Mode Configuration
16.8 Capture Mode (100b)
  16.8.1 Capture Mode Timer Period
  16.8.2 Capture Mode Configuration
16.9 Compare Mode (101b)
  16.9.1 Compare Mode Timer Period
  16.9.2 Compare Mode Configuration
16.10 Gated Mode (110b)
  16.10.1 Gated Mode Timer Period
  16.10.2 Gated Mode Configuration
16.11 Capture/Compare Mode (111b)
  16.11.1 Capture/Compare Timer Period
  16.11.2 Capture/Compare Configuration
16.12 Timer Registers

17. Pulse Train Engine (PT)
  17.1 Instances
  17.2 Pulse Train Engine Features
  17.3 Engine
  17.3.1 Pulse Train Output Modes
  17.4 Enabling and Disabling a Pulse Train Output
  17.5 Atomic Pulse Train Output Enable and Disable
    17.5.1 Pulse Train Atomic Enable
    17.5.2 Pulse Train Atomic Disable
  17.6 Pulse Train Halt and Disable
17.7 Pulse Train Interrupts---------------------------------------------------------------352
17.8 Registers------------------------------------------------------------------------352
17.9 Register Details------------------------------------------------------------------354
17.9.1 Pulse Train Engine Safe Enable Register----------------------------------------363
17.9.2 Pulse Train Engine Safe Disable Register---------------------------------------364

18. Real-Time Clock (RTC)-------------------------------------------------------------369
18.1 Overview------------------------------------------------------------------------369
18.2 Instances------------------------------------------------------------------------370
18.3 Register Access Control-----------------------------------------------------------370
18.3.1 RTC_SEC and RTC_SSEC Read Access Control--------------------------------------370
18.3.2 RTC Write Access Control------------------------------------------------------371
18.4 RTC Alarm Functions--------------------------------------------------------------371
18.4.1 Time-of-Day Alarm-------------------------------------------------------------371
18.4.2 Sub-Second Alarm--------------------------------------------------------------372
18.4.3 RTC Interrupt and Wakeup Configuration-----------------------------------------373
18.4.4 Square Wave Output------------------------------------------------------------373
18.5 RTC Calibration----------------------------------------------------------------374
18.6 Registers------------------------------------------------------------------------376
18.7 Register Details----------------------------------------------------------------376

19. Watchdog Timer (WDT)-------------------------------------------------------------380
19.1 Features-------------------------------------------------------------------------381
19.2 Usage-----------------------------------------------------------------------------381
19.3 Interrupt and Reset Period Timeout Configuration--------------------------------382
19.4 Timed Access Protection----------------------------------------------------------382
19.5 Enabling the Watchdog Timer------------------------------------------------------383
19.5.1 Enable sequence---------------------------------------------------------------383
19.6 Disabling the Watchdog Timer------------------------------------------------------383
19.6.1 Manual Disable----------------------------------------------------------------383
19.6.2 Automatic Disable-------------------------------------------------------------383
19.7 Resetting the Watchdog Timer------------------------------------------------------383
19.7.1 Reset Sequence----------------------------------------------------------------383
19.8 Detection of a Watchdog Reset Event----------------------------------------------383
19.9 Registers------------------------------------------------------------------------383
19.10 Register Details----------------------------------------------------------------384

20. 1-Wire Master (OWM)---------------------------------------------------------------387
20.1 Instances

20.2 Pins and Configuration

20.2.1 Pin Configuration

20.2.2 1-Wire I/O (OWM_IO)

20.2.3 Pullup Enable (OWM_PE)

20.2.4 Clock Configuration

20.3 1-Wire Protocol

20.3.1 Networking Layers

20.3.2 Read ROM Command

20.3.3 Skip ROM and Overdrive Skip ROM Commands

20.3.4 Match ROM and Overdrive Match ROM Commands

20.3.5 Search ROM Command

20.3.6 Search ROM Accelerator Operation

20.3.7 Resume Communication Command

20.4 1-Wire Operation

20.4.1Resetting the OWM

20.5 1-Wire Data Reads

20.5.1 Reading a Single Bit Value from the 1-Wire Bus

20.5.2 Reading an 8-Bit Value from the 1-Wire Bus

20.6 Registers

20.7 Register Details

21. USB 2.0 High-Speed (USBHS) Host Interface with PHY

21.1 Instances

21.2 USBHS Bus Signals

21.3 USBHS Device Endpoints

21.4 USBHS Reset and Clock

21.5 USBHS SUSPEND Mode and RESUME States

21.6 Packet Size

21.7 Endpoint 0 Control Transactions

21.7.1 Endpoint 0 Error Handling

21.8 Bulk Endpoints Operation and Options

21.8.1 Bulk IN Endpoints

21.8.2 Bulk OUT Endpoints

21.9 Interrupt Endpoints

21.9.1 Interrupt IN Endpoints

21.9.2 Interrupt OUT Endpoints
21.10 Isochronous Endpoints

21.10.1 Isochronous IN Endpoints

21.10.2 Isochronous OUT Endpoints

21.11 USBHS Device Registers

21.12 USBHS Device Register Details

21.12.2 Endpoint Register Access Control

21.12.3 USBHS IN Endpoint Maximum Packet Size Registers

21.12.4 USBHS IN Endpoint Lower Control and Status Registers

21.12.5 USBHS Endpoint 0 Control Status Register

21.12.6 USBHS IN Endpoint Upper Control Registers

22. Bluetooth 5 Low Energy (LE) Radio

22.1 Power-Efficient Design

22.2 Bluetooth Hardware Accelerator

22.3 Arm Cordio®-B50 Software Stack

22.4 Pins

22.5 Configuration

22.6 Documentation

23. Trust Protection Unit (TPU)

23.1 Dedicated Cryptographic DMA Engine (CDMA)

23.1.1 FIFOs

23.1.2 Direct FIFO Access

23.1.3 Cache Security

23.2 Block Cipher Accelerator

23.2.1 Cipher Key Storage and Initialization

23.2.2 Operation

23.3 Hash Function Accelerator

23.3.1 Last Message Block Padding

23.4 CRC Engine (Galois Field Accelerator)

23.5 Hamming Code Accelerator

23.6 Modular Arithmetic Accelerator

23.6.1 Operation

23.6.2 MAA Memory

23.7 True Random Number Generation

23.8 Registers

23.8.1 Write Access

23.8.2 Read Access
List of Figures

Figure 1-1: MAX32665—MAX32668 Block Diagram .................................................. 25
Figure 3-1: Code Memory Mapping ........................................................................... 35
Figure 3-2: Data Memory Mapping .......................................................................... 36
Figure 4-1: Clock Block Diagram ............................................................................. 51
Figure 4-2: Example 32MHz Crystal Capacitor Determination .............................. 53
Figure 4-3: SLEEP Mode Clock Control ................................................................ 55
Figure 4-4: DEEPSLEEP Clock Control ................................................................ 57
Figure 4-5: BACKUP Mode Clock Control ............................................................... 59
Figure 4-6: MAX32665—MAX32668 Cache Controllers Control ......................... 63
Figure 8-1. Simplified SPIXF Block Diagram ......................................................... 143
Figure 8-2. Simplified Block Diagram ..................................................................... 144
Figure 8-3. SPIXF Transaction Delay .................................................................... 149
Figure 8-4. Supported SPI configuration ................................................................ 159
Figure 8-5. SPIXF Delay Configuration ................................................................ 160
Figure 8-6. Simplified SPIXR Block Diagram ......................................................... 168
Figure 8-7: SDHC Block Diagram .......................................................................... 182
Figure 8-8: SD Bus Protocol - No Response and No Data Operations ................. 183
Figure 8-9: SD Bus Protocol - Multi-Block Read Operation ................................... 184
Figure 8-10: SD Bus Protocol - Multi Block Write Operation ................................. 184
Figure 9-1: DMA Block-Chaining Flowchart ......................................................... 229
Figure 10-1: Galois Field CRC and LFSR Architecture .......................................... 240
Figure 11-1: Analog to Digital Converter Block Diagram ...................................... 248
Figure 11-2: ADC Limit Engine ............................................................................. 253
Figure 12-1: UART Frame Diagram ........................................................................ 260
Figure 13-1: I2C Block Diagram ............................................................................ 273
Figure 13-2: I2C Write Data Transfer ..................................................................... 276
Figure 13-3: I2C SCL Timing for Standard, Fast, and Fast-Plus Modes .............. 278
Figure 14-1: QSPI Block Diagram ........................................................................ 307
Figure 14-2: 4-Wire-SPI Connection Diagram ...................................................... 309
Figure 14-3: Generic 3-Wire SPI Master to Slave Connection .............................. 310
Figure 14-4: Dual Mode SPI Connection Diagram ................................................. 311
Figure 14-5: SCK Clock Rate Control .................................................................... 312
Figure 14-6: SPI Clock Polarity .............................................................................. 313
Figure 16-1: One-Shot Mode Diagram ................................................................... 330
Figure 16-2: Continuous Mode Diagram ................................................................. 332
Figure 16-3: Counter Mode Diagram ...................................................................... 334
Figure 16-4: Capture Mode Diagram ...................................................................... 338
Figure 16-5: Counter Mode Diagram ...................................................................... 340
Figure 16-6: Gated Mode Diagram ........................................................................ 342
Figure 18-1. MAX32665—MAX32668 RTC Block Diagram (12-bit Sub-Second Counter) ........................................................................................................... 369
Figure 18-2. RTC Busy/Ready Signal Timing ............................................................ 371
Figure 18-3. RTC Interrupt/Wakeup Diagram Wakeup Function ......................... 373
Figure 18-4. Internal Implementation of Digital Trim, 4kHz ................................ 375
Figure 19-1: Watchdog Timer Block Diagram ....................................................... 381
Figure 20-1: 1-Wire Signal Interface ...................................................................... 389
List of Tables

Table 2-1: Dual Mapped APB Peripherals ................................................................. 26
Table 2-2: MAX32665—MAX32668 Master Permission Bits ................................. 26
Table 2-3: MAX32665—MAX32668 AHB Slaves .................................................... 27
Table 2-4: MAX32665—MAX32668 AHB Master/Slave Interconnect Matrix ....... 27
Table 2-5: RPU APB Register Offsets, Names, Access, and Descriptions .......... 29
Table 2-6: RPU AHB Slave Register Addresses, Names, Access, and Descriptions 30
Table 2-7: RPU APB Slave Permission Registers .................................................... 31
Table 2-8: RPU AHB SlavePermission Register .................................................... 33
Table 3-1: APB Peripheral Base Address Map ....................................................... 41
Table 3-2: AHB Peripheral Base Address Map ....................................................... 43
Table 3-3: Error Correction Coding (ECC) Enable Register ................................ 44
Table 3-4: Error Correction Coding (ECC) Error Register .................................... 44
Table 3-5: Correctable Error Detected Register ..................................................... 46
Table 3-6: Error Correction Coding (ECC) Interrupt Enable Register ............... 47
Table 3-7: Error Correction Coding (ECC) Address Register ............................... 48
Table 4-1: Wakeup Sources .................................................................................... 54
Table 4-2: Reset and Low Power Mode Effects ..................................................... 61
Table 4-3: Instruction Cache Controller Register Summary ............................... 64
Table 4-4: SPIXF Cache Controller Register Summary ........................................ 64
Table 4-5: ICCn Cache ID Register ....................................................................... 64
Table 4-6: ICCn Memory Size Register ................................................................. 65
Table 4-7: ICCn Cache Control Register ............................................................... 65
Table 4-8: ICCn Invalidate Register .................................................................... 65
Table 4-9: SFCC Cache ID Register .................................................................... 65
Table 4-10: SFCC Memory Size Register .............................................................. 66
Table 4-11: SFCC Cache Control Register ............................................................ 66
Table 4-12: SFCC Invalidate Register .................................................................. 66
Table 4-13: RAM Block Size and Base Address .................................................... 67
Table 4-14: Miscellaneous Control Register Summary ......................................... 68
Table 4-15: Error Correction Coding (ECC) Enable Register ......................... 68
Table 4-16: SQWOUT and PDOWN Output Enable Register ............................. 69
Table 4-17: Comparator Enable Register ............................................................. 69
Table 4-18: Control Register ................................................................................ 70
Table 4-19: SIMO Power Supply Device Pin Connectivity .................................... 71
Table 4-20: SIMO Controller Register Summary ................................................ 72
Table 4-21: Buck Voltage Regulator A Control Register .................................... 73
Table 4-22: Buck Voltage Regulator B Control Register .................................... 73
Table 4-23: Buck Voltage Regulator C Control Register .................................... 74
Table 4-24: Buck Voltage Regulator D Control Register .................................... 74
Table 4-25: High Side FET Peak Current VREGO_A VREGO_B Register .......... 74
Table 4-26: High Side FET Peak Current VREGO_C VREGO_D Register .......... 75
Table 4-27: Maximum High Side FET Time On Register .................................... 75
Table 4-28: Buck Cycle Count VREGO_A Register ............................................. 75
Table 4-29: Buck Cycle Count VREGO_B Register ............................................. 75
Table 4-30: Buck Cycle Count VREGO_C Register ............................................. 76
Table 4-31: Buck Cycle Count VREGO_D Register ............................................. 76
Table 4-32: Buck Cycle Count Alert VREGO_A Register .................................... 76
Table 4-33: Buck Cycle Count Alert VREGO_B Register .................................... 76
Table 4-34: Buck Cycle Count Alert VREGO_C Register .................................... 76
Table 4-35: Buck Cycle Count Alert VREGO_D Register .................................... 77
Table 4-36: Buck Regulator Output Ready Register ................................................................. 77
Table 4-37: Zero Cross Calibration VREGO_A Register .......................................................... 77
Table 4-38: Zero Cross Calibration VREGO_B Register .......................................................... 78
Table 4-39: Zero Cross Calibration VREGO_C Register .......................................................... 78
Table 4-40: Zero Cross Calibration VREGO_D Register .......................................................... 78
Table 4-41: Power Sequencer and Always-On Domain Register Summary ............................... 78
Table 4-42: Low Power Control Register .................................................................................. 79
Table 4-43: GPIO00 Low Power Wakeup Status Flags ............................................................... 80
Table 4-44: GPIO00 Low Power Wakeup Enable Registers ....................................................... 80
Table 4-45: GPIO1 Low Power Wakeup Status Flags ................................................................. 80
Table 4-46: GPIO1 Low Power Wakeup Enable Registers ......................................................... 80
Table 4-47: Peripheral Low Power Wakeup Status Flags ........................................................... 81
Table 4-48: Peripheral Low Power Wakeup Enable Register .................................................... 82
Table 4-49: RAM Shutdown Control Register .......................................................................... 83
Table 4-50: Low Power VDD Power Down Register .................................................................. 84
Table 4-51: BACKUP Return Vector Register .......................................................................... 85
Table 4-52: BACKUP AoD Register ......................................................................................... 85
Table 4-53: Global Control Register Summary ........................................................................ 85
Table 4-54: System Control Register ...................................................................................... 86
Table 4-55: Reset Register 0 .................................................................................................... 87
Table 4-56: System Clock Control Register .............................................................................. 90
Table 4-57: Power Management Register ............................................................................... 92
Table 4-58: Peripheral Clock Divisor Register ........................................................................ 93
Table 4-59: Peripheral Clock Disable Register 0 .................................................................. 94
Table 4-60: Memory Clock Control Register .......................................................................... 96
Table 4-61: Memory Zeroization Control Register .................................................................. 98
Table 4-62: System Status Flag Register .................................................................................. 100
Table 4-63: Reset Register 1 .................................................................................................. 100
Table 4-64: Peripheral Clock Disable Register 1 ................................................................... 102
Table 4-65: Event Enable Register .......................................................................................... 105
Table 4-66: Revision Register .................................................................................................. 106
Table 4-67: System Status Interrupt Enable Register ............................................................. 106
Table 4-68: Error Correction Coding Error Register .............................................................. 106
Table 4-69: Error Correction Not Double Error Detected Register ......................................... 107
Table 4-70: Error Correction Coding Interrupt Enable Register ........................................... 109
Table 4-71: Error Correction Coding Address Register .......................................................... 110
Table 4-72: Bluetooth LDO Control Register ......................................................................... 110
Table 4-73: Bluetooth LDO Delay Count Register ................................................................. 112
Table 4-74: General Purpose 0 Register .................................................................................. 112
Table 4-75: Arm Peripheral Bus Asynchronous Bridge Select Register .................................... 112
Table 4-76: Function Control Register Summary ................................................................. 113
Table 4-77: Function Control Register 0 .................................................................................. 113
Table 4-78: AES Key Register Summary ................................................................................. 114
Table 4-79: AES Key 0 and 1 Registers .................................................................................... 114
Table 4-80: AES Key 2 and 3 Registers .................................................................................... 114
Table 5-1: MAX32665—MAX32668 Interrupt Vector Table .................................................. 115
Table 6-1: MAX32665—MAX32668 GPIO Pin Count .............................................................. 119
Table 6-2: MAX32665—MAX32668 GPIO and Alternate Function Matrix, 140 WLP ............ 120
Table 6-3: MAX32665—MAX32668 GPIO Pin Configuration .................................................. 121
Table 6-4: MAX32665—MAX32668 Input Mode Configuration ............................................... 121
Table 6-5: MAX32665—MAX32668 Output Mode Configuration ............................................ 122
Table 6-6: MAX32665—MAX32668 GPIO Port Interrupt Vector Mapping ............................. 122
Table 6-7: MAX32665—MAX32668 GPIO Wakeup Interrupt Vector ......................................... 124
Table 6-8: GPIO Register Summary
Table 6-9: GPIO Port n Configuration Enable Bit 0 Register
Table 6-10: GPIO Port n Configuration Enable Atomic Set Bit 0 Register
Table 6-11: GPIO Port n Configuration Enable Atomic Set Bit 0 Register
Table 6-12: GPIO Port n Output Enable Register
Table 6-13: GPIO Port n Output Enable Atomic Set Register
Table 6-14: GPIO Port n Output Enable Atomic Clear Register
Table 6-15: GPIO Port n Output Register
Table 6-16: GPIO Port n Output Atomic Set Register
Table 6-17: GPIO Port n Output Atomic Clear Register
Table 6-18: GPIO Port n Input Register
Table 6-19: GPIO Port n Interrupt Mode Register
Table 6-20: GPIO Port n Interrupt Polarity Register
Table 6-21: GPIO Port n Input Enable Register
Table 6-22: GPIO Port n Interrupt Enable Register
Table 6-23: GPIO Port n Interrupt Enable Atomic Set Register
Table 6-24: GPIO Port n Interrupt Enable Atomic Clear Register
Table 6-25: GPIO Port n Interrupt Status Register
Table 6-26: GPIO Port n Interrupt Clear Register
Table 6-27: GPIO Port n Wakeup Enable Register
Table 6-28: GPIO Port n Wakeup Enable Atomic Set Register
Table 6-29: GPIO Port n Wakeup Enable Clear Register
Table 6-30: GPIO Port n Interrupt Dual Edge Mode Register
Table 6-31: GPIO Port n Pullup Pulldown Selection 0 Register
Table 6-32: GPIO Port n Pullup Pulldown Selection 1 Register
Table 6-33: GPIO Port n Configuration Enable Bit 1 Register
Table 6-34: GPIO Port n Configuration Enable Atomic Set, Bit 1 Register
Table 6-35: GPIO Port n Configuration Enable Atomic Clear, Bit 1 Register
Table 6-36: GPIO Port n Configuration Enable Bit 2 Register
Table 6-37: GPIO Port n Configuration Enable Atomic Set Bit 2 Register
Table 6-38: GPIO Port n Configuration Enable Atomic Clear Bit 2 Register
Table 6-39: GPIO Port n Output Drive Strength Bit 0 Register
Table 6-40: GPIO Port n Output Drive Strength Bit 0 Register
Table 6-41: GPIO Pullup/Pullown Strength Select Register
Table 6-42: GPIO Supply Voltage Select Register
Table 7-1: MAX32665—MAX32668 Internal Flash Memory Organization
Table 7-2: Valid Addresses Flash Writes
Table 7-3: Flash Controller Registers
Table 7-4: Flash Controller Address Pointer Register
Table 7-5: Flash Controller Clock Divisor Register
Table 7-6: Flash Controller Control Register
Table 7-7: Flash Controller Interrupt Register
Table 7-8: Flash Controller ECC Data Register
Table 7-9: Flash Controller Data Register 0
Table 7-10: Flash Controller Data Register 1
Table 7-11: Flash Controller Data Register 2
Table 7-12: Flash Controller Data Register 3
Table 8-1: SPI Header Format
Table 8-2: Clock Polarity and Phase Combinations
Table 8-3: Encrypted Data Write Order to SPIX Flash Memory
Table 8-4: SPIXF Master Controller Register Offsets, Names, Access and Description
Table 8-5: SPIXF Controller Configuration Register
Table 8-6: SPIXF Controller Slave Select Polarity Register
Table 8-7. SPIXF Controller General Control Register ................................................................. 153
Table 8-8. SPIXF Controller FIFO Control and Status Register .............................................. 154
Table 8-9. SPIXF Controller Special Control Register ................................................................. 155
Table 8-10. SPIXF Controller Interrupt Status Register .............................................................. 156
Table 8-11. SPIXF Controller Interrupt Enable Register ............................................................. 157
Table 8-12. SPIXF Master Controller FIFO Register Offsets, Names, Access and Description ................................................................. 158
Table 8-13. SPIXF Master Controller TX FIFO Register ............................................................... 158
Table 8-14. SPIXF Master Controller TX FIFO Register ............................................................... 158
Table 8-15. SPIXF Master Register Offsets, Names, Access and Description ................................................................. 162
Table 8-16. SPIXF Configuration Register .................................................................................. 162
Table 8-17. SPIXF Fetch Control Register ..................................................................................... 163
Table 8-18. SPIXF Mode Control Register .................................................................................... 164
Table 8-19. SPIXF Mode Data Register ....................................................................................... 165
Table 8-20. SPIXF SCK Feedback Control Register ...................................................................... 165
Table 8-21. SPIXF I/O Control Register ........................................................................................ 165
Table 8-22. SPIXF Memory Security Control Register ................................................................. 166
Table 8-23. SPIXF Bus Idle Detection .......................................................................................... 166
Table 8-24. SPIXR Master Controller Register Offsets, Names, Access and Descriptions .......... 168
Table 8-25. SPIXR FIFO Data Register ....................................................................................... 169
Table 8-26. SPIXR Master Signals Control Register ...................................................................... 169
Table 8-27. SPIXR Transmit Packet Size Register ....................................................................... 170
Table 8-28. SPIXR Static Configuration Register ......................................................................... 170
Table 8-29. SPIXR Slave Select Timing Register ......................................................................... 171
Table 8-30. SPIXR Master Baud Rate Generator ......................................................................... 172
Table 8-31. SPIXR DMA Control Register ................................................................................... 173
Table 8-32. SPIXR Interrupt Status Flag Register ......................................................................... 174
Table 8-33. SPIXR Interrupt Enable Register ............................................................................. 175
Table 8-34. SPIXR Wakeup Flag Register .................................................................................... 176
Table 8-35. SPIXR Wakeup Enable Register ............................................................................... 177
Table 8-36. SPIXR Active Status Register .................................................................................. 177
Table 8-37. SPIXR External Memory Control Register ................................................................. 177
Table 8-38. External Memory Cache Controller Register Addresses and Descriptions ........... 179
Table 8-39: SRCC Cache ID Register ........................................................................................... 179
Table 8-40: SRCC Memory Size Register ..................................................................................... 179
Table 8-41: SRCC Cache Control Register .................................................................................. 179
Table 8-42: SRCC Invalidate Register ......................................................................................... 180
Table 8-43: MAX32665—MAX32668 SDHC Alternate Function Mapping to SDHC Specification Pin Names ......................................................................................... 182
Table 8-44: Registers Used to Generate SD Commands ................................................................. 184
Table 8-45: SDHC Register Offsets, Names and Descriptions ....................................................... 185
Table 8-46: SDHC SDMA System Address / Argument Register ................................................ 186
Table 8-47: SDHC SDMA Block Size Register .............................................................................. 187
Table 8-48: SDHC SDMA Block Count Register ......................................................................... 188
Table 8-49: SDHC SDMA Argument 1 Register .......................................................................... 188
Table 8-50: SDHC SDMA Transfer Mode Register ...................................................................... 188
Table 8-51: Summary of how register settings determine type of data transfer ......................... 190
Table 8-52: SDHC Command Register ....................................................................................... 190
Table 8-53: Relationship between Parameters and the Name of Response Type ....................... 191
Table 8-54: SDHC Response 0 Register ...................................................................................... 191
Table 8-55: SDHC Response 1 Register ...................................................................................... 191
Table 8-56: SDHC Response 2 Register ...................................................................................... 191
Table 8-57: SDHC Response 3 Register ...................................................................................... 192
Table 8-58: SDHC Response 4 Register ...................................................................................... 192
Table 8-59: SDHC Response 5 Register ...................................................................................... 192
Table 8-60: SDHC Response 6 Register ............................................................... 192
Table 8-61: SDHC Response 7 Register ............................................................... 193
Table 8-62: SDHC Response Register Mapping to SD Host Controller Response Register Convention ......................................................... 193
Table 8-63: Kind of SD Card Response Mapping to SDHC Response Registers ................................................................. 193
Table 8-64: SDHC Buffer Data Port Register ......................................................... 193
Table 8-65: SDHC Present State Register ............................................................. 193
Table 8-66: SDHC Host Control 1 Register .......................................................... 196
Table 8-67: SDHC Power Control Register .......................................................... 197
Table 8-68: SDHC Block Gap Control Register .................................................... 197
Table 8-69: SDHC Wakeup Control Register ....................................................... 199
Table 8-70: SDHC Clock Control Register ........................................................... 199
Table 8-71: SDHC Timeout Control Register ....................................................... 201
Table 8-72: SDHC Software Reset Register ......................................................... 202
Table 8-73: SDHC Normal Interrupt Status Register .......................................... 203
Table 8-74: Transfer Complete and Data Timeout Error Priority and Status ............... 205
Table 8-75: Command Complete and Command Timeout Error Priority and Status ........ 205
Table 8-76: SDHC Error Interrupt Status Register .............................................. 205
Table 8-77: SDHC Normal Interrupt Status Register .......................................... 207
Table 8-78: SDHC Error Interrupt Status Enable Register .................................. 208
Table 8-79: SDHC Normal Interrupt Signal Enable Register .................................. 209
Table 8-80: SDHC Error Interrupt Signal Enable Register .................................. 210
Table 8-81: SDHC Auto CMD Error Status Register ........................................... 211
Table 8-82: SDHC Host Control 2 Register .......................................................... 212
Table 8-83: SDHC Capabilities Register 0 ............................................................ 213
Table 8-84: SDHC Capabilities Register 1 ............................................................ 215
Table 8-85: SDHC Maximum Current Capabilities Register ............................... 216
Table 8-86: SDHC Force Event Register for Auto CMD Error Status Register .......... 216
Table 8-87: SDHC Force Event Register for Error Interrupt Status ....................... 216
Table 8-88: SDHC ADMA Error Status Register ................................................. 217
Table 8-89: SDHC ADMA System Address Register 0 ....................................... 218
Table 8-90: SDHC ADMA System Address Register 1 ....................................... 219
Table 8-91: Preset Value Register Example ......................................................... 219
Table 8-92: Preset Value Register Selection Conditions ..................................... 219
Table 8-93: SDHC Preset Value 0 to Preset Value 7 Registers ............................... 220
Table 8-94: SDHC Slot Interrupt Status Register .............................................. 220
Table 8-95: SDHC Host Controller Version Register .......................................... 221
Table 9-1: MAX32665—MAX32668 DMAC and Channel Instances .................... 223
Table 9-2: MAX32665—MAX32668 DMAC Source and Destination by Peripheral ...................................................................................................... 224
Table 9-3: Data Movement from Source to DMA FIFO ....................................... 226
Table 9-4: Data Movement from the DMA FIFO to Destination ............................. 226
Table 9-5: DMA Channel Timeout Configuration ............................................... 230
Table 9-6: DMAC Register Summary ................................................................. 231
Table 9-7: DMACn Control Register ................................................................. 231
Table 9-8: DMACn Interrupt Register ............................................................... 232
Table 9-9: Standard DMA Channel 0 to Channel 7 Register Summary .................... 232
Table 9-10: DMACH Channel Registers Summary ............................................. 232
Table 9-11: DMACHn Configuration Register ..................................................... 233
Table 9-12: DMA Status Register ................................................................. 234
Table 9-13: DMACHn Source Register ............................................................. 235
Table 9-14: DMA Channel n Destination Register ............................................. 236
Table 9-15: DMA Channel n Count Register ..................................................... 236
Table 9-16: DMA Channel n Source Reload Register ........................................ 236
Table 9-17: DMA Channel n Destination Reload Register ................................... 236
Table 9-18: DMA Channel n Count Reload Register ................................................................. 237
Table 10-1: Common CRC Polynomials ..................................................................................... 239
Table 10-2: CRC Register Summary .......................................................................................... 240
Table 10-3: CRC Control Register ............................................................................................ 243
Table 10-4: CRC DMA Source Register .................................................................................... 243
Table 10-5: CRC DMA Destination Register .......................................................................... 243
Table 10-6: CRC DMA Count Register ..................................................................................... 243
Table 10-7: CRC Data Input Registers ....................................................................................... 244
Table 10-8: CRC Data Output Registers ................................................................................... 244
Table 10-9: CRC Polynomial Register ..................................................................................... 244
Table 10-10: CRC Value Register ............................................................................................ 245
Table 10-11: CRC Pseudo-Random Number Generator Register ........................................ 245
Table 11-1: MAX32665—MAX32668 ADC Peripheral Pins ..................................................... 246
Table 11-2: ADC Clock Frequency and ADC Conversion Time (\( f_{SYSCLK} = 96\, \text{MHz} \), \( f_{PCLK} = 48\, \text{MHz} \)) ................................................................................................................ 249
Table 11-3: Input and Reference Scale Support by ADC Input Channel ................................ 251
Table 11-4: ADC Data Register Alignment Options ................................................................. 252
Table 11-5: ADC Registers Summary ....................................................................................... 255
Table 11-6: ADC Control Register .......................................................................................... 255
Table 11-7: ADC Status Register ............................................................................................ 257
Table 11-8: ADC Data Register ............................................................................................... 257
Table 11-9: ADC Interrupt Control Register .......................................................................... 257
Table 11-10: ADC Limit 0 to 3 Registers ................................................................................. 258
Table 12-1: UART Interrupt Conditions ................................................................................... 261
Table 12-2: Example Baud Rate Calculation Results, Target Bit Rate = 1.8Mbps .............. 262
Table 12-3: UART Register Summary ...................................................................................... 264
Table 12-4: UART Control 0 Register ...................................................................................... 264
Table 12-5: UART Control 1 Register ...................................................................................... 266
Table 12-6: UART Status Register .......................................................................................... 266
Table 12-7: UART Interrupt Enable Register ......................................................................... 267
Table 12-8: UART Interrupt Flags Register ............................................................................ 268
Table 12-9: UART Rate Integer Register ................................................................................. 270
Table 12-10: UART Baud Rate Decimal Register ................................................................. 270
Table 12-11: UART FIFO Register .......................................................................................... 270
Table 12-12: UART DMA Configuration Register ................................................................. 270
Table 12-13: UART Transmit FIFO Data Output Register .................................................... 271
Table 13-1: MAX32665 – MAX32668 I²C Peripheral Pins ...................................................... 274
Table 13-2: I²C Bus Terminology ........................................................................................... 275
Table 13-3: Calculated I²C Bus Clock Frequencies .............................................................. 279
Table 13-4: I²C Slave Address Format .................................................................................... 279
Table 13-5: I²C Registers ........................................................................................................ 292
Table 13-6: I²C Control 0 Register ........................................................................................... 293
Table 13-7: I²C Status Register ............................................................................................... 294
Table 13-8: I²C Interrupt Flag 0 Register ................................................................................ 295
Table 13-9: I²C Interrupt Enable 0 Register .......................................................................... 297
Table 13-10: I²C Interrupt Flag 1 Register ............................................................................. 299
Table 13-11: I²C Interrupt Enable 1 Register ........................................................................ 299
Table 13-12: I²C FIFO Length Register .................................................................................. 299
Table 13-13: I²C Receive Control 0 Register .......................................................................... 299
Table 13-14: I²C Receive Control 1 Register .......................................................................... 300
Table 13-15: I²C Transmit Control 0 Register ....................................................................... 301
Table 13-16: I²C Transmit Control 1 Register ....................................................................... 302
Table 13-17: I²C Data Register .............................................................................................. 302
Table 13-18: I²C Master Mode Control Register .................................................................... 303
Table 18-7. RTC Time-of-Day Alarm Register ................................................................. 377
Table 18-8. RTC Sub-Second Alarm Register ................................................................. 377
Table 18-9. RTC Control Register ................................................................................. 377
Table 18-10. RTC 32kHz Oscillator Digital Trim Register ........................................... 377
Table 18-11. RTC 32kHz Oscillator Control Register .................................................. 379
Table 19-1: Watchdog Timer Interrupt Period fSYS_CLK = 96MHz and fPCLK = 48MHz ......................... 382
Table 19-2: Watchdog Timer Register Offsets, Names and Descriptions ...................... 383
Table 19-3: Watchdog Timer Control Register .............................................................. 384
Table 19-4: Watchdog Timer Reset Register .................................................................. 386
Table 20-1: OWM Pin to Alternate Function Mapping ................................................... 388
Table 20-2: 1-Wire ROM Commands ........................................................................... 392
Table 20-3: 1-Wire Slave Device ROM ID Field ........................................................... 393
Table 20-4: OWM Register Summary ............................................................................ 397
Table 20-5: OWM Configuration Register ..................................................................... 398
Table 20-6: OWM Clock Divisor Register ..................................................................... 399
Table 20-7: OWM Control/Status Register ................................................................. 399
Table 20-8: OWM Data Register .................................................................................... 400
Table 20-9: OWM Interrupt Flag Register ..................................................................... 400
Table 20-10: OWM Interrupt Enable Register ............................................................. 401
Table 21-1: USB Bus States Indicated by the Differential Pair (D+, D-) .............................. 403
Table 21-2: USB Bulk IN Endpoints Options ............................................................... 405
Table 21-3: USB Isochronous IN Endpoint Options ...................................................... 407
Table 21-4: USB Isochronous OUT Endpoint Options .................................................. 408
Table 21-5: USBHS Device Register Offsets, Names, Access, and Descriptions .......... 409
Table 21-6: USBHS Device Address Register ............................................................... 410
Table 21-7: USBHS Power Management Register ....................................................... 410
Table 21-8: USBHS IN Endpoint Interrupt Flags Register ............................................ 411
Table 21-9: USBHS OUT Endpoint Interrupt Flags Register ......................................... 412
Table 21-10: USBHS IN Endpoint Interrupt Enable Register ........................................ 412
Table 21-11: USBHS OUT Endpoint Interrupt Enable Register ................................... 414
Table 21-12: USBHS Signaling Interrupt Status Flag Register ....................................... 415
Table 21-13: USBHS Signaling Interrupt Enable Register ............................................. 415
Table 21-14: USBHS Frame Number Register .............................................................. 415
Table 21-15: USBHS Register Index Select Register ...................................................... 416
Table 21-16: USBHS Test Mode Register ....................................................................... 416
Table 21-17: USB Memory Mapped Register Access for Endpoints 1 to 11 .................... 416
Table 21-18: USBHS IN Endpoint Maximum Packet Size Register ............................. 417
Table 21-19: USBHS IN Endpoint Lower Control and Status Register ......................... 417
Table 21-20: USBHS Endpoint 0 Control Status Register .............................................. 418
Table 21-21: USBHS IN Endpoint Upper Control Register ........................................... 419
Table 21-22: USBHS OUT Endpoint Maximum Packet Size Register ........................ 420
Table 21-23: USBHS OUT Endpoint Lower Control Status Register .......................... 421
Table 21-24: USBHS OUT Endpoint Upper Control Status Register ........................... 422
Table 21-25: USBHS Endpoint OUT FIFO Byte Count Register ................................ 423
Table 21-26: USBHS Endpoint 0 IN FIFO Byte Count Register .................................. 423
Table 21-27: USBHS FIFO for Endpoint n Register ...................................................... 423
Table 21-28: USBHS Endpoint Count Info Register ...................................................... 424
Table 21-29: USBHS RAM Info Register ....................................................................... 424
Table 21-30: USBHS Soft Reset Control Register ......................................................... 424
Table 21-31: USBHS Early DMA Register ...................................................................... 424
Table 21-32: USBHS Hi-Speed Chirp Timeout Register .................................................. 425
Table 21-33: USBHS Hi-Speed RESUME Delay Register .............................................. 425
Table 23-1. Cryptographic Accelerator DMA Sources .................................................... 431
Table 23-2. Symmetric Block Ciphers .................................................................................................. 433
Table 23-3. Hash Functions .............................................................................................................. 436
Table 23-4. Cryptographic Memory Segments .................................................................................. 440
Table 23-5. MAA Memory Segments and Locations ....................................................................... 441
Table 23-6. MAA Memory Blinding Example (Memory Instance 0, MAWS > 1024) ......................... 441
Table 23-7. Cryptographic Registers, Offsets and Descriptions ..................................................... 443
Table 23-8: Cryptographic Control Register ...................................................................................... 444
Table 23-9: Cipher Control Register ................................................................................................. 446
Table 23-10: Hash Control Register ................................................................................................. 447
Table 23-11: CRC Control Register ................................................................................................. 448
Table 23-12: Cryptographic DMA Source Register ......................................................................... 449
Table 23-13: Cryptographic DMA Destination Register ................................................................. 449
Table 23-14: Cryptographic DMA Count Register .......................................................................... 449
Table 23-15: MAA Control Register ............................................................................................... 449
Table 23-16: Cryptographic Data Input Register ............................................................................. 452
Table 23-17: Cryptographic Data Output Register .......................................................................... 452
Table 23-18: CRC Polynomial Register ......................................................................................... 453
Table 23-19: CRC Value Register .................................................................................................. 453
Table 23-20: CRC PRNG Register .................................................................................................. 453
Table 23-21: Hamming Error Correction Code Register ................................................................. 453
Table 23-22: Cipher Initial Vector Register [3:0] ............................................................................ 453
Table 23-23: Cipher Key Register [7:0] .......................................................................................... 454
Table 23-24: HASH Message Digest Register [15:0] ..................................................................... 454
Table 23-25: Hash Message Size Registers .................................................................................... 454
Table 23-26: MAA Word Size Register ......................................................................................... 455
Table 23-27: TRNG Control Register (Base address 0x400B_5000) .............................................. 456
Table 23-28: TRNG Data Register (Base address 0x400B_5000) ................................................... 456
1. Overview

The MAX32665—MAX32668 are Arm® Cortex®-M4 with FPU-based microcontrollers with 1MB flash and up to 560KB SRAM that can be configured as 448KB SRAM with error correction coding (ECC). They are ideal for wearable medical fitness applications. Optionally available is a second Arm Cortex-M4 with FPU for audio signal processing in a wireless headset/earbud application (MAX32665/MAX32666 only). The architecture includes a Bluetooth® 5 Low Energy radio.

The MAX32665/MAX32666 feature a second Arm Cortex-M4 with FPU with supporting ROM and cache. This feature supports extended data processing capabilities such as audio processing for wireless Bluetooth applications. Refer to the Ordering Information in the device datasheet for device feature detail.

The devices feature five powerful and flexible power modes. Built-in dynamic clock gating and firmware-controlled power gating allows the user to optimize power for the specific application. A built-in single inductor multiple output (SIMO) switch mode power supply allows the device to be optionally self-powered by a primary lithium cell.

The flash memory is split into two banks of 512KB to provide flexibility when programming over-the-air. The devices have an ECC with single error correction double error detection (SEC-DED) for flash, and SRAM providing extremely reliable code execution. Dedicated hardware runs the Bluetooth 5 Low Energy stack freeing the CPUs for data processing tasks. Multiple SPI, UART, and I2C serial interfaces, 1-Wire® Master, and USB 2.0 High-Speed Device interface allow for interconnection to a wide variety of external sensors. An audio subsystem supporting PDM, PCM, I²S, and TDM. An 8-input, 10-bit ADC is available to monitor analog input from external sensors and meters.

The MAX32666/MAX32668 incorporate a trust protection unit (TPU) with encryption and advanced security features. These features include a modular arithmetic accelerator (MAA) for fast ECDSA, a hardware AES engine, a hardware TRNG entropy generator, a SHA-2 accelerator and a secure bootloader.

The high-level block diagram for the MAX32665—MAX32668 is shown in Figure 1-1.

---

1-Wire is a registered trademark of Maxim Integrated Products, Inc.

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

Cortex is a registered trademark of Arm Limited.

Bluetooth is a registered trademark of Bluetooth SIG, Inc.
1.1 Block Diagram

Figure 1-1: MAX32665—MAX32668 Block Diagram
2. Resource Protection Unit (RPU)

The resource protection unit (RPU) is a dedicated module that augments the Arm Cortex-M4 core memory protection unit (MPU). The RPU allows the system to provide resource protection for AHB bus masters accessing system memory or AHB/APB bus slaves.

The MPU enforces privilege and access controls that restrict a CPU from accessing user-defined segments of system memory. It does not, however, provide access controls for other AHB bus masters accessing system memory or AHB/APB bus slaves.

The RPU is separate to and maintains software compatibility with Arm’s MPU privilege levels on the CPU side and uses the standard AMBA bus.

The RPU features include the following:

- Software compatibility with Arm Memory Protection Unit (MPU)
- Provides access control for DMA and other AHB masters
- Access controls for each bus slave independently configurable
- Dedicated access-protection register for each bus slave

2.1 Instances

The IC has three buses:

- AHB
- APB Bus 0
- APB Bus 1

The peripherals in Table 2-1 can be mapped to either APB Bus 0 or the fixed frequency APB Bus 1. Each peripheral has a different register set depending on which bus the peripheral is connected to.

Table 2-1: Dual Mapped APB Peripherals

<table>
<thead>
<tr>
<th>Peripheral</th>
<th>APB Bus 0</th>
<th>APB Bus 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2C0</td>
<td>I2C0_BUS0</td>
<td>I2C0_BUS1</td>
</tr>
<tr>
<td>I2C1</td>
<td>I2C1_BUS0</td>
<td>I2C1_BUS1</td>
</tr>
<tr>
<td>I2C2</td>
<td>I2C2_BUS0</td>
<td>I2C2_BUS1</td>
</tr>
</tbody>
</table>

The AHB bus masters are listed in Table 2-2. Each bus slave has a dedicated RPU access control register. Each bit position in a slave’s RPU register allows or denies access by a specific AHB bus master as shown in Table 2-2. Register bits corresponding to unimplemented bus masters should not be changed from their reset default value.

Because of the structure of the APB bus, there is only one access control bit. This means that the read and write access permissions for a particular master must always be the same. Access permissions for read and write can be configured separately for AHB slaves.

Table 2-2. MAX32665—MAX32668 Master Permission Bits

<table>
<thead>
<tr>
<th>AHB Master</th>
<th>Bit Position in SLAVEAPB Register</th>
<th>Bit Position in SLAVEAHB Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Write</td>
<td>Read</td>
<td>Write</td>
</tr>
<tr>
<td>DMAC0</td>
<td>access[0]</td>
<td></td>
<td>access[1]</td>
</tr>
</tbody>
</table>
Table 2-3 lists the AHB slaves addressable by AHB bus masters. Each AHB slave has a dedicated RPU access control register similar to the APB RPU access control registers, but each AHB slave has two control bits.

Table 2-3: MAX32665—MAX32668 AHB Slaves

<table>
<thead>
<tr>
<th>AHB Slave</th>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USBHS</td>
<td>USB FIFO</td>
<td>USB Endpoint Data</td>
</tr>
<tr>
<td>SDIO/SDHC Target</td>
<td>SDIO/SDHC Target Memory</td>
<td>SDIO/SDHC Target Memory</td>
</tr>
<tr>
<td>SPIM</td>
<td>SPI FIFO Memory</td>
<td>SPI Bus Master FIFO</td>
</tr>
<tr>
<td>QSPI/SPI</td>
<td>QSPI FIFO</td>
<td>QSPI Data Buffer</td>
</tr>
<tr>
<td>SYS_RAM (MI0)</td>
<td>Configurable by Arm MPU</td>
<td>System RAM, Memory Instance 0</td>
</tr>
<tr>
<td>SYS_RAM (MI1)</td>
<td>Configurable by Arm MPU</td>
<td>System RAM, Memory Instance 1</td>
</tr>
<tr>
<td>SYS_RAM (MI2)</td>
<td>Configurable by Arm MPU</td>
<td>System RAM, Memory Instance 2</td>
</tr>
<tr>
<td>SYS_RAM (MI3)</td>
<td>Configurable by Arm MPU</td>
<td>System RAM, Memory Instance 3</td>
</tr>
<tr>
<td>SYS_RAM (MI4)</td>
<td>Configurable by Arm MPU</td>
<td>System RAM, Memory Instance 4</td>
</tr>
<tr>
<td>SYS_RAM (MI5)</td>
<td>Configurable by Arm MPU</td>
<td>System RAM, Memory Instance 5</td>
</tr>
<tr>
<td>SYS_RAM (MI6)</td>
<td>Configurable by Arm MPU</td>
<td>System RAM, Memory Instance 6</td>
</tr>
</tbody>
</table>

The AHB bus prohibits some AHB master and slave interactions as shown in Table 2-4. The AHB slave ignores the state of prohibited combinations.

Table 2-4. MAX32665—MAX32668 AHB Master/Slave Interconnect Matrix

<table>
<thead>
<tr>
<th>AHB Slave</th>
<th>DMAC0</th>
<th>DMAC1</th>
<th>USB</th>
<th>SYS0</th>
<th>SYS1</th>
<th>SDMAD</th>
<th>SDMAI</th>
<th>CRYPTO</th>
<th>SDIO/SDHC MASTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS_RAM (MI0)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>SYS_RAM (MI1)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>SYS_RAM (MI2)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>SYS_RAM (MI3)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>SYS_RAM (MI4)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>SYS_RAM (MI5)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>SYS_RAM (MI6)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
2.2 Usage

2.2.1 Reset State

During a power-on-reset event, RPU registers are reset to their reset value. If RPU protection is desired, the registers must be reprogrammed during the boot sequence.

The RPU registers can also be reset by writing 1 to the Global Control register bit \textit{RSTR.RPU}.

2.2.2 MPU Implementation

Accesses to system memory still involve interaction with both the RPU first and then MPU. The RPU grants access to MPU functionality including:

- Protection regions
- Overlapping protection regions, with ascending region priority
- Access permissions
- Exporting memory attributes to the system

The MPU can:

- Enforce privilege rules
- Separate processes
- Enforce access rules

2.2.3 MPU Protection Fault

The MPU can generate three types of faults:

- Background fault
- Permission fault
- Alignment fault

When a fault occurs, the memory access or instruction fetch is synchronously aborted, and a prefetch abort or data abort exception is taken as appropriate. No memory accesses are performed on the AXI bus master interface or peripheral.

2.2.4 RPU Protection Fault

An RPU protection fault occurs when an AHB master attempts to access a slave that does not have the corresponding bits in its RPU register cleared. This will be expressed as an AHB bus fault.

2.2.5 RPU Fault Handler

Sample code demonstrating the implementation of an RPU Fault Handler is provided in the SDK.
2.3 Registers

See Table 3-1: APB Peripheral Base Address Map for the RPU Peripheral Base Address.

Table 2-5: RPU APB Register Offsets, Names, Access, and Descriptions

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>GCR</td>
<td>R/W</td>
<td>GCR RPU Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>SIR</td>
<td>R/W</td>
<td>SIR RPU Register</td>
</tr>
<tr>
<td>[0x0008]</td>
<td>FCR</td>
<td>R/W</td>
<td>FCR RPU Register</td>
</tr>
<tr>
<td>[0x0010]</td>
<td>TPU</td>
<td>R/W</td>
<td>TPU RPU Register</td>
</tr>
<tr>
<td>[0x0020]</td>
<td>RPU</td>
<td>R/W</td>
<td>RPU Register</td>
</tr>
<tr>
<td>[0x0030]</td>
<td>WDT0</td>
<td>R/W</td>
<td>WDT0 RPU Register</td>
</tr>
<tr>
<td>[0x0034]</td>
<td>WDT1</td>
<td>R/W</td>
<td>WDT1 RPU Register</td>
</tr>
<tr>
<td>[0x0038]</td>
<td>WDT2</td>
<td>R/W</td>
<td>WDT2 RPU Register</td>
</tr>
<tr>
<td>[0x0040]</td>
<td>SMON</td>
<td>R/W</td>
<td>SMON RPU Register</td>
</tr>
<tr>
<td>[0x0044]</td>
<td>SIMO</td>
<td>R/W</td>
<td>SIMO RPU Register</td>
</tr>
<tr>
<td>[0x0048]</td>
<td>DVS</td>
<td>R/W</td>
<td>DVS RPU Register</td>
</tr>
<tr>
<td>[0x0050]</td>
<td>AES</td>
<td>R/W</td>
<td>AES RPU Register</td>
</tr>
<tr>
<td>[0x0060]</td>
<td>RTC</td>
<td>R/W</td>
<td>RTC RPU Register</td>
</tr>
<tr>
<td>[0x0064]</td>
<td>WUT</td>
<td>R/W</td>
<td>WUT RPU Register</td>
</tr>
<tr>
<td>[0x0068]</td>
<td>PWRSEQ</td>
<td>R/W</td>
<td>PWRSEQ RPU Register</td>
</tr>
<tr>
<td>[0x006C]</td>
<td>MCR</td>
<td>R/W</td>
<td>MCR RPU Register</td>
</tr>
<tr>
<td>[0x0080]</td>
<td>GPIO0</td>
<td>R/W</td>
<td>GPIO0 RPU Register</td>
</tr>
<tr>
<td>[0x0090]</td>
<td>GPIO1</td>
<td>R/W</td>
<td>GPIO1 RPU Register</td>
</tr>
<tr>
<td>[0x0100]</td>
<td>TMR0</td>
<td>R/W</td>
<td>TMR0 RPU Register</td>
</tr>
<tr>
<td>[0x0110]</td>
<td>TMR1</td>
<td>R/W</td>
<td>TMR1 RPU Register</td>
</tr>
<tr>
<td>[0x0120]</td>
<td>TMR2</td>
<td>R/W</td>
<td>TMR2 RPU Register</td>
</tr>
<tr>
<td>[0x0130]</td>
<td>TMR3</td>
<td>R/W</td>
<td>TMR3 RPU Register</td>
</tr>
<tr>
<td>[0x0140]</td>
<td>TMR4</td>
<td>R/W</td>
<td>TMR4 RPU Register</td>
</tr>
<tr>
<td>[0x0150]</td>
<td>TMR5</td>
<td>R/W</td>
<td>TMR5 RPU Register</td>
</tr>
<tr>
<td>[0x01B0]</td>
<td>HTMR0</td>
<td>R/W</td>
<td>HTMR0 RPU Register</td>
</tr>
<tr>
<td>[0x01C0]</td>
<td>HTMR1</td>
<td>R/W</td>
<td>HTMR1 RPU Register</td>
</tr>
<tr>
<td>[0x01D0]</td>
<td>I2C0_BUS0</td>
<td>R/W</td>
<td>I2C0_BUS0 RPU Register</td>
</tr>
<tr>
<td>[0x01E0]</td>
<td>I2C1_BUS0</td>
<td>R/W</td>
<td>I2C1_BUS0 RPU Register</td>
</tr>
<tr>
<td>[0x01F0]</td>
<td>I2C2_BUS0</td>
<td>R/W</td>
<td>I2C2_BUS0 RPU Register</td>
</tr>
<tr>
<td>[0x0260]</td>
<td>SPIXF</td>
<td>R/W</td>
<td>SPIXF RPU Register</td>
</tr>
<tr>
<td>Offset</td>
<td>Register</td>
<td>Access</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>--------</td>
<td>---------------------</td>
</tr>
<tr>
<td>[0x0270]</td>
<td>SPIXFC</td>
<td>R/W</td>
<td>SPIXFC RPU Register</td>
</tr>
<tr>
<td>[0x0280]</td>
<td>DMA0</td>
<td>R/W</td>
<td>DMA0 RPU Register</td>
</tr>
<tr>
<td>[0x0290]</td>
<td>FLC0</td>
<td>R/W</td>
<td>FLC0 RPU Register</td>
</tr>
<tr>
<td>[0x0294]</td>
<td>FLC1</td>
<td>R/W</td>
<td>FLC1 RPU Register</td>
</tr>
<tr>
<td>[0x02A0]</td>
<td>ICC0</td>
<td>R/W</td>
<td>ICC0 RPU Register</td>
</tr>
<tr>
<td>[0x02A4]</td>
<td>ICC1</td>
<td>R/W</td>
<td>ICC1 RPU Register</td>
</tr>
<tr>
<td>[0x02F0]</td>
<td>ICX</td>
<td>R/W</td>
<td>ICX RPU Register</td>
</tr>
<tr>
<td>[0x0330]</td>
<td>EMCC</td>
<td>R/W</td>
<td>EMCC RPU Register</td>
</tr>
<tr>
<td>[0x0340]</td>
<td>ADC</td>
<td>R/W</td>
<td>ADC RPU Register</td>
</tr>
<tr>
<td>[0x0350]</td>
<td>DMA1</td>
<td>R/W</td>
<td>DMA1 RPU Register</td>
</tr>
<tr>
<td>[0x0360]</td>
<td>SDMA</td>
<td>R/W</td>
<td>SDMA RPU Register</td>
</tr>
<tr>
<td>[0x0370]</td>
<td>SDHC</td>
<td>R/W</td>
<td>SD Host Controller (APB)</td>
</tr>
<tr>
<td>[0x03A0]</td>
<td>SPIXR</td>
<td>R/W</td>
<td>SPIXR RPU Register</td>
</tr>
<tr>
<td>[0x03C0]</td>
<td>PTG_BUS0</td>
<td>R/W</td>
<td>PTG_BUS0 RPU Register</td>
</tr>
<tr>
<td>[0x03D0]</td>
<td>OWM</td>
<td>R/W</td>
<td>OWM RPU Register</td>
</tr>
<tr>
<td>[0x03E0]</td>
<td>SEMA</td>
<td>R/W</td>
<td>SEMA RPU Register</td>
</tr>
<tr>
<td>[0x0420]</td>
<td>UART0</td>
<td>R/W</td>
<td>UART0 RPU Register</td>
</tr>
<tr>
<td>[0x0430]</td>
<td>UART1</td>
<td>R/W</td>
<td>UART1 RPU Register</td>
</tr>
<tr>
<td>[0x0440]</td>
<td>UART2</td>
<td>R/W</td>
<td>UART2 RPU Register</td>
</tr>
<tr>
<td>[0x0460]</td>
<td>SPI1</td>
<td>R/W</td>
<td>SPI1 RPU Register</td>
</tr>
<tr>
<td>[0x0470]</td>
<td>SPI2</td>
<td>R/W</td>
<td>SPI2 RPU Register</td>
</tr>
<tr>
<td>[0x04C0]</td>
<td>AUDIO</td>
<td>R/W</td>
<td>AUDIO RPU Register</td>
</tr>
<tr>
<td>[0x04D0]</td>
<td>TRNG</td>
<td>R/W</td>
<td>TRNG RPU Register</td>
</tr>
<tr>
<td>[0x0500]</td>
<td>BTLE</td>
<td>R/W</td>
<td>BTLE RPU Register</td>
</tr>
<tr>
<td>[0x0B10]</td>
<td>USBHS</td>
<td>R/W</td>
<td>USBHS RPU Register</td>
</tr>
<tr>
<td>[0x0B60]</td>
<td>SDIO/SDHC Target</td>
<td>R/W</td>
<td>SDIO/SDHC Target RPU Register</td>
</tr>
<tr>
<td>[0x0BC0]</td>
<td>SPIM</td>
<td>R/W</td>
<td>SPIM RPU Register</td>
</tr>
<tr>
<td>[0x0BE0]</td>
<td>QSPI/SPI</td>
<td>R/W</td>
<td>QSPI/SPI RPU Register</td>
</tr>
</tbody>
</table>

Table 2-6: RPU AHB Slave Register Addresses, Names, Access, and Descriptions
### 2.4 Register Details

**Table 2-7: RPU APB Slave Permission Registers**

<table>
<thead>
<tr>
<th>APB Address</th>
<th>Register</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0F00]</td>
<td>SYS_RAM (M0)</td>
<td>R/W</td>
<td>SYS_RAM (M0) RPU Register</td>
</tr>
<tr>
<td>[0x0F10]</td>
<td>SYS_RAM (M1)</td>
<td>R/W</td>
<td>SYS_RAM (M1) RPU Register</td>
</tr>
<tr>
<td>[0x0F20]</td>
<td>SYS_RAM (M2)</td>
<td>R/W</td>
<td>SYS_RAM (M2) RPU Register</td>
</tr>
<tr>
<td>[0x0F30]</td>
<td>SYS_RAM (M3)</td>
<td>R/W</td>
<td>SYS_RAM (M3) RPU Register</td>
</tr>
<tr>
<td>[0x0F40]</td>
<td>SYS_RAM (M4)</td>
<td>R/W</td>
<td>SYS_RAM (M4) RPU Register</td>
</tr>
<tr>
<td>[0x0F50]</td>
<td>SYS_RAM (M5)</td>
<td>R/W</td>
<td>SYS_RAM (M5) RPU Register</td>
</tr>
<tr>
<td>[0x0F60]</td>
<td>SYS_RAM (M6)</td>
<td>R/W</td>
<td>SYS_RAM (M6) RPU Register</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Register Mnemonic</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Control RPU Register</td>
<td>GCR</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>System Interface RPU Register</td>
<td>SIR</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Function Control RPU Register</td>
<td>FCR</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Trust Protection Unit RPU Register</td>
<td>TPU</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Resource Protection Unit RPU Register</td>
<td>RPU</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Watchdog Timer 0 RPU Register</td>
<td>WDT0</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Watchdog Timer 1 RPU Register</td>
<td>WDT1</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Watchdog Timer 2 RPU Register</td>
<td>WDT2</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Security Monitor RPU Register</td>
<td>SMON</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>SIMO Controller RPU Register</td>
<td>SIMO</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>DVS Controller RPU Register</td>
<td>DVS</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>AES Keys RPU Register</td>
<td>AES</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Real-Time Clock RPU Register</td>
<td>RTC</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Wake-Up Timer RPU Register</td>
<td>WUT</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Power Sequencer RPU Register</td>
<td>PWRSEQ</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Miscellaneous Control Register RPU Register</td>
<td>MCR</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>GPIO Port 0 RPU Register</td>
<td>GPIO0</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>GPIO Port 1 RPU Register</td>
<td>GPIO1</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Timer 0 RPU Register</td>
<td>TMR0</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Timer 1 RPU Register</td>
<td>TMR1</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Timer 2 RPU Register</td>
<td>TMR2</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Timer 3 RPU Register</td>
<td>TMR3</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Register Name</td>
<td>Register Mnemonic</td>
<td>Reference</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Timer 4 RPU Register</td>
<td>TMR4</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Timer 5 RPU Register</td>
<td>TMR5</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>HTimer 0 RPU Register</td>
<td>HTMR0</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>HTimer 1 RPU Register</td>
<td>HTMR1</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>I2C Bus 0 (Bus 0) RPU Register</td>
<td>I2C0_BUS0</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>I2C Bus 1 (Bus 0) RPU Register</td>
<td>I2C1_BUS0</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>I2C Bus 2 (Bus 0) RPU Register</td>
<td>I2C2_BUS0</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>SPIXF Master RPU Register</td>
<td>SPIXF</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>SPIXF Master Controller RPU Register</td>
<td>SPIXFC</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Standard DMA 0 RPU Register</td>
<td>DMA0</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Flash Controller 0 RPU Register</td>
<td>FLC0</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Flash Controller 1 RPU Register</td>
<td>FLC1</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Instruction Cache Controller 0 RPU Register</td>
<td>ICC0</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Instruction Cache Controller 1 RPU Register</td>
<td>ICC1</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Instruction Cache Controller XIP RPU Register</td>
<td>ICX</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>External Memory Cache Controller RPU Register</td>
<td>EMCC</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Analog-to-Digital Converter RPU Register</td>
<td>ADC</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Standard DMA 1 RPU Register</td>
<td>DMA1</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Smart DMA RPU Register</td>
<td>SDMA</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>SD Host Controller (APB) RPU Register</td>
<td>SDHC</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>SPIXR Master Controller RPU Register</td>
<td>SPIXR</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Pulse Train Engine (Bus 0) RPU Register</td>
<td>PTG_BUS0</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>1-Wire Module RPU Register</td>
<td>OWM</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Semaphores RPU Register</td>
<td>SEMA</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>UART 0 RPU Register</td>
<td>UART0</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>UART 1 RPU Register</td>
<td>UART1</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>UART 2 RPU Register</td>
<td>UART2</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>SPI 1 RPU Register</td>
<td>SPI1</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>SPI 2 RPU Register</td>
<td>SPI2</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>Audio Subsystem RPU Register</td>
<td>AUDIO</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>True Random Number Generator RPU Register</td>
<td>TRNG</td>
<td>See Table 2-5</td>
</tr>
<tr>
<td>BTLE Registers and IQ RAMs RPU Register</td>
<td>BTLE</td>
<td>See Table 2-5</td>
</tr>
</tbody>
</table>
### Register Name
<table>
<thead>
<tr>
<th>Register Mnemonic</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2C 0 (Bus 1) RPU Register</td>
<td>I2C0_BUS1</td>
</tr>
<tr>
<td>I2C 1 (Bus 1) RPU Register</td>
<td>I2C1_BUS1</td>
</tr>
<tr>
<td>I2C 2 (Bus 1) RPU Register</td>
<td>I2C2_BUS1</td>
</tr>
<tr>
<td>Pulse Train Engine (Bus 1) RPU Register</td>
<td>PTG_BUS1</td>
</tr>
</tbody>
</table>

### Table 2-8: RPU AHB Slave Permission Register
<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>access</td>
<td>R/W</td>
<td>0</td>
<td>APB Slave Peripheral Access Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: AHB master read/write access allowed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: AHB master read/write access denied</td>
</tr>
</tbody>
</table>

This field allows or denies access to the peripheral by one or more AHB masters as shown in Table 2-2. Unused bits should not be changed from their reset default values.

### Table 2-8: RPU AHB Slave Permission Register
<table>
<thead>
<tr>
<th>Register Name</th>
<th>Register Mnemonic</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>USB Endpoint Data RPU Register</td>
<td>USBHS</td>
<td>See Table 2-6</td>
</tr>
<tr>
<td>SDIO/SDHC Target Memory RPU Register</td>
<td>SDIO/SDHC Target</td>
<td>See Table 2-6</td>
</tr>
<tr>
<td>SPI Bus Master FIFO RPU Register</td>
<td>SPIM</td>
<td>See Table 2-6</td>
</tr>
<tr>
<td>QSPI Data Buffer RPU Register</td>
<td>SPI/QSPI</td>
<td>See Table 2-6</td>
</tr>
<tr>
<td>System RAM, Memory Instance 0</td>
<td>SYS_RAM (MI0)</td>
<td>See Table 2-6</td>
</tr>
<tr>
<td>System RAM, Memory Instance 1</td>
<td>SYS_RAM (MI1)</td>
<td>See Table 2-6</td>
</tr>
<tr>
<td>System RAM, Memory Instance 2</td>
<td>SYS_RAM (MI2)</td>
<td>See Table 2-6</td>
</tr>
<tr>
<td>System RAM, Memory Instance 3</td>
<td>SYS_RAM (MI3)</td>
<td>See Table 2-6</td>
</tr>
<tr>
<td>System RAM, Memory Instance 4</td>
<td>SYS_RAM (MI4)</td>
<td>See Table 2-6</td>
</tr>
<tr>
<td>System RAM, Memory Instance 5</td>
<td>SYS_RAM (MI5)</td>
<td>See Table 2-6</td>
</tr>
<tr>
<td>System RAM, Memory Instance 6</td>
<td>SYS_RAM (MI6)</td>
<td>See Table 2-6</td>
</tr>
</tbody>
</table>

### Table 2-8: RPU AHB Slave Permission Register
<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>access</td>
<td>R/W</td>
<td>0</td>
<td>AHB Slave Peripheral Access Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00: AHB master write/read access allowed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01: AHB master write access allowed / read access denied</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b10: AHB master write access denied / read access allowed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b11: AHB master write/read access denied</td>
</tr>
</tbody>
</table>

Each bit pair of this field allows or denies access to the peripheral by one or more AHB masters as shown in Table 2-2. Unused bits should not be changed from their reset default values.
3. Memory, Register Mapping, and Access

3.1 Memory, Register Mapping, and Access Overview

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

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 3-1: Code Memory Mapping

Legend:
- Arm Cortex-M Defined Buses
- Memory Spaces
- Memory Spaces (Cached)
- External Memory Device (Optional)
- Internal Memory Instances
- Undefined/Reserved

System AHB Bus Master (for code fetches)

External SPI SRAM (QSPI SPIXF, 128MB Maximum)

SRAM 560KB (no ECC) 448KB (with ECC)

External SPI Flash Memory (QSPI SPIXR, 512MB Maximum)

Internal Program/Data Flash Memory 512KB (Block 0) + ECC

Internal Program/Data Flash Memory 512KB (Block 1) + ECC

External SPI Flash Memory (QSPI SPIXF, 128MB Maximum)
### Figure 3-2: Data Memory Mapping

<table>
<thead>
<tr>
<th>Address Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000_0000</td>
<td>Undefined/Reserved</td>
</tr>
<tr>
<td>0x0000_FFFF</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0800_0000</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0800_FFFF</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x2000_0000</td>
<td>SRAM 560KB (no ECC) 448KB (with ECC)</td>
</tr>
<tr>
<td>0x2000_BFFFF</td>
<td>SRAM 560KB (no ECC) 448KB (with ECC)</td>
</tr>
<tr>
<td>0x2000_0000</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x2007_0000</td>
<td>Information Block 1</td>
</tr>
<tr>
<td>0x2006_FFFF</td>
<td>Information Block 0</td>
</tr>
<tr>
<td>0x1008_0000</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x1008_3FFF</td>
<td>Information Block 1</td>
</tr>
<tr>
<td>0x1008_0000</td>
<td>Information Block 0</td>
</tr>
<tr>
<td>0x1000_0000</td>
<td>Information Block 1</td>
</tr>
<tr>
<td>0x1008_FFFFF</td>
<td>Information Block 0</td>
</tr>
<tr>
<td>0x0800_0000</td>
<td>Internal Memory Instances</td>
</tr>
<tr>
<td>0x0000_0000</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x0000_FFFF</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x0800_0000</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x0800_FFFF</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x2000_0000</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x2000_BFFFF</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x2000_0000</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x2007_0000</td>
<td>Information Block 1</td>
</tr>
<tr>
<td>0x2006_FFFF</td>
<td>Information Block 0</td>
</tr>
<tr>
<td>0x1008_0000</td>
<td>Information Block 1</td>
</tr>
<tr>
<td>0x1008_3FFF</td>
<td>Information Block 0</td>
</tr>
<tr>
<td>0x1008_0000</td>
<td>Information Block 1</td>
</tr>
<tr>
<td>0x1000_0000</td>
<td>Information Block 1</td>
</tr>
<tr>
<td>0x1000_FFFFF</td>
<td>Information Block 0</td>
</tr>
<tr>
<td>0x0800_0000</td>
<td>Internal Memory Instances</td>
</tr>
<tr>
<td>0x0000_0000</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x0000_FFFF</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x0800_0000</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x0800_FFFF</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x2000_0000</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x2000_BFFFF</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x2000_0000</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x2007_0000</td>
<td>Information Block 1</td>
</tr>
<tr>
<td>0x2006_FFFF</td>
<td>Information Block 0</td>
</tr>
<tr>
<td>0x1008_0000</td>
<td>Information Block 1</td>
</tr>
<tr>
<td>0x1008_3FFF</td>
<td>Information Block 0</td>
</tr>
<tr>
<td>0x1008_0000</td>
<td>Information Block 1</td>
</tr>
<tr>
<td>0x1000_0000</td>
<td>Information Block 1</td>
</tr>
<tr>
<td>0x1000_FFFFF</td>
<td>Information Block 0</td>
</tr>
<tr>
<td>0x0800_0000</td>
<td>Internal Memory Instances</td>
</tr>
<tr>
<td>0x0000_0000</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x0000_FFFF</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x0800_0000</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x0800_FFFF</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x2000_0000</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x2000_BFFFF</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x2000_0000</td>
<td>Code Space Access to Code Space</td>
</tr>
<tr>
<td>0x2007_0000</td>
<td>Information Block 1</td>
</tr>
<tr>
<td>0x2006_FFFF</td>
<td>Information Block 0</td>
</tr>
<tr>
<td>0x1008_0000</td>
<td>Information Block 1</td>
</tr>
<tr>
<td>0x1008_3FFF</td>
<td>Information Block 0</td>
</tr>
<tr>
<td>0x1008_0000</td>
<td>Information Block 1</td>
</tr>
<tr>
<td>0x1000_0000</td>
<td>Information Block 1</td>
</tr>
<tr>
<td>0x1000_FFFFF</td>
<td>Information Block 0</td>
</tr>
<tr>
<td>0x0800_0000</td>
<td>Internal Memory Instances</td>
</tr>
</tbody>
</table>
3.2 Standard Memory Regions

Several standard memory regions are defined for the Arm Cortex-M4 architecture; the use of many of these is optional for the system integrator. At a minimum, the MAX32665—MAX32668, Cortex-M4-based devices, must contain some code and data memory for application code and variable/stack use, as well as certain components which are part of the instantiated core.

3.2.1 Code Space

The code space area of memory is designed to contain the primary memory used for code execution by the device. This memory area is defined from byte address range 0x0000 0000 to 0x1FFF FFFF (0.5GB maximum). 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.

The MAX32665—MAX32668 code memory mapping is illustrated in Figure 3-1: Code Memory Mapping. The code space memory area contains the main internal flash memory, which holds most of the instruction code that will be executed on the device. The internal flash memory is mapped into both code and data space from 0x1000 0000 to 0x100F FFFF. It is partitioned as two 512KB blocks of usable flash plus extra flash storage for Error Correction Coding (ECC) check bits, if ECC is enabled. This additional storage is not user accessible, even when ECC is disabled.

This program memory area must also contain the default system vector table and the initial settings for all system exception handlers and interrupt handlers. The reset vector for the device is 0x0000 0000 where a vector to re-direct to 0x1000 0000 is located.

The code space memory on the MAX32665—MAX32668 also contains the mapping for the flash information block, from 0x1080 0000 to 0x1080 7FFF. However, this mapping is generally only present during Maxim Integrated production test; it is disabled once the information block has been loaded with valid data and the info block lockout option has been set. This memory is accessible for data reads only and cannot be used for code execution.

Optionally, the SPIXF (SPI Execute In Place Flash) and the SPIXR (SPI Execute In Place RAM) modules can be used to expand the available code and data memory space. This expansion consists of mapping the contents of an external SPI flash memory device (up to 128MB) or an external SPI RAM memory device (up to 512MB) into a read-only area of the code memory map. If enabled, the external SPIXF memory is mapped starting at byte address 0x0800 0000 up to a maximum of 0x0FFF FFFF (for a 128MB device). Also, if enabled, the external SPIXR memory is mapped starting at byte 0x8000 0000 up to a maximum of 0x9FFF FFFF (for a 512MB device). This external memory can be used for code execution as well as static data storage.

3.2.2 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 Arm Cortex-M4 stack.

The MAX32665—MAX32668 data memory mapping is illustrated in Figure 3-2: Data Memory Mapping. This memory area contains the main system SRAM. The size of the internal SRAM is 560KB when not using ECC. Its address range is 0x2000 0000 to 0x2008 BFFF. If ECC is enabled, the SRAM size decreases to 448KB. The address range with ECC enabled is 0x2000 0000 to 0x2006 FFFF.

The entirety of the SRAM memory space on the MAX32665—MAX32668 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 1MB 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.

The SRAM area on the MAX32665—MAX32668 can be used to contain executable code. Code stored in the SRAM is accessed directly for execution (using the system bus) and is not cached. The SRAM is also where the Arm Cortex-M4 stack must be located, as it is the only general-purpose SRAM memory on the device. A valid stack location inside the SRAM must be set by the system exception table (which is, by default, stored at the beginning of the internal flash memory).

The MAX32665—MAX32668 specific AH Bus Masters can access the SRAM to use as general storage or working space. Specifically, in the case of the USB interface, SRAM memory area can be used to store the descriptor table for the endpoint buffers as well as the endpoint buffers themselves.

### 3.2.3 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 MAX32665—MAX32668, 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 MAX32665—MAX32668, 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.

A secondary region within the peripheral memory space (0x0400B 0000 to 0x400F FFFF) allows peripherals that require more rapid data transfer to handle this data transfer using their own local AHB slave instances (instead of going indirectly through the AHB-to-APB bridge). This allows peripherals which have FIFOs or other functions requiring large amounts of data to be transferred quickly (such as the SD/SDIO/SDHC/MMC or communications peripherals like SPI) to benefit from the more rapid data transfer rate of the AHB bus.

### 3.2.4 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 MAX32665—MAX32668 implements support for external SPI SRAM. The external SPI SRAM SPIXR interface is mapped to byte address 0x8000 0000 to 0x9FFF FFFF (up to 512MB).
3.2.5  **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 MAX32665—MAX32668 does not implement this memory area.

3.2.6  **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 SD/SDIO/SDHC/MMC 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.

3.2.7  **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 MAX32665—MAX32668 does not implement this memory region.

3.3  **Device Memory Instances**

This section details physical memory instances on the MAX32665—MAX32668 (including internal flash memory and SRAM instances) that are accessible as standalone memory regions using either the AHB or APB bus matrix. Memory areas which are only accessible via FIFO interfaces, or memory areas consisting of only a few registers for a specific peripheral, are not covered here.

3.3.1  **Main Program Flash Memory**

The main program flash memory is 1MB in size (two banks of 512KB) and consists of 128 logical pages of 8,192 Bytes per page.

3.3.2  **Cache Memories**

3.3.2.1  **Instruction Cache Controller 0 (ICC0)**

The internal flash memory instruction cache is 16,384 Bytes in size and is used to cache instructions fetched using the I-Code bus, including instructions fetched from the internal flash memory. This instruction cache controller is referred to as ICC0 throughout this document.

3.3.2.2  **Instruction Cache Controller 1 (ICC1)**

The SPIXF instruction cache, managed by ICC1, is also 16,384 Bytes and is used to cache instructions fetched from an external SPI memory device. ICC1 is only available if the SPIXF controller is enabled.

*Note: The instruction caches, ICC0 and ICC1, are used for instruction fetches only. Data fetches (including code literal values) from the internal flash memory or external SPIXF memory do not use the instruction cache.*
3.3.3  **External Memory Cache Controller (EMCC)**

The SPIXR RAM interface is supported by a dedicated 16,384 Byte 2-way set-associative Least Recently Used (LRU) write-through cache. This cache is managed through the EMCC interface.

3.3.4  **Information Block Flash Memory**

The information block is a separate flash instance of 16KB. It is used to store trim settings (option configuration and analog trim) as well as other nonvolatile device-specific information intended for use by firmware.

3.3.5  **System SRAM**

The system SRAM is 560KB in size and can be used for general purpose data storage, the Arm system stack, USB data transfers (endpoints), SD Host Controller (SDHC) interface, TPU and code execution if desired.

3.3.6  **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.

3.3.7  **MAA Key and Working Space Memory**

The MAA contains a dedicated memory for key storage, input and output parameters for operations, and working space. It is mapped into the AHB memory space for ease of loading and unloading.

3.3.8  **TPU Memory**

The MAX32665—MAX32668 contains a specialized 128-bit memory that is designed to preserve critical data (such as a 128-bit AES key) even when the device is in the lowest power-saving state. As long as the RTC power supply is still available, the contents of this memory will be retained, even if the AES block and the main SRAM are shut down completely.

The Secure Key Storage Area consists of four VRTC supply backed 32-bit registers: TPU_TSR_SKS0, TPU_TSR_SKS1, TPU_TSR_SKS2, and TPU_TSR_SKS3.

3.4  **AHB Interfaces**

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

3.4.1  **Core AHB Interfaces**

3.4.1.1  **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 and the external SPIF flash memory (if SPIXF is enabled). 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 or the external SPIXF flash memory when a cache miss occurs.

3.4.1.2  **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, the external SPIXF flash memory (if SPIXF is enabled), and the information block.

3.4.1.3  **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.
### 3.4.2 AHB Masters

#### 3.4.2.1 USB Endpoint Buffer Manager

The USB AHB bus master is used to manage endpoint buffers in the SRAM. It has access to the SRAM (read/write, for storage and retrieval of endpoint buffer data), as well as the internal and/or external flash data contents (which can be used to contain static data for transmission by the USB).

#### 3.4.2.2 Standard DMA

The Standard DMA bus master has access to all off-core memory areas accessible by the System bus. It does not have access to the Arm Private Peripheral Bus area.

#### 3.4.2.3 SDHC

The SDHC bus master has access to all off-core memory areas accessible by the System bus. It does not have access to the Arm Private Peripheral Bus area.

#### 3.4.2.4 Trust Protection Unit (TPU)

The TPU bus master has access to all off-core memory areas accessible by the System bus. It does not have access to the Arm Private Peripheral Bus area.

### 3.5 Peripheral Register Map

#### 3.5.1 APB Peripheral Base Address Map

*Table 3-1, below,* contains the base address for each of the APB mapped peripherals. The base address for a given peripheral is the start of the register map for the peripheral. For a given peripheral, the address for a register within the peripheral is defined as the APB peripheral base address plus the registers offset.

<table>
<thead>
<tr>
<th>Peripheral Register Name</th>
<th>Register Prefix</th>
<th>APB Base Address</th>
<th>APB End Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Control</td>
<td>GCR_</td>
<td>0x4000 0000</td>
<td>0x4000 03FF</td>
</tr>
<tr>
<td>System Interface</td>
<td>SIR_</td>
<td>0x4000 0400</td>
<td>0x4000 07FF</td>
</tr>
<tr>
<td>Function Control</td>
<td>FCR_</td>
<td>0x4000 0800</td>
<td>0x4000 0BFF</td>
</tr>
<tr>
<td>Trust Protection Unit</td>
<td>TPU_</td>
<td>0x4000 1000</td>
<td>0x4000 1FFF</td>
</tr>
<tr>
<td>Resource Protection Unit</td>
<td>RPU_</td>
<td>0x4000 2000</td>
<td>0x4000 2FFF</td>
</tr>
<tr>
<td>Watchdog Timer 0</td>
<td>WDT0_</td>
<td>0x4000 3000</td>
<td>0x4000 33FF</td>
</tr>
<tr>
<td>Watchdog Timer 1</td>
<td>WDT1_</td>
<td>0x4000 3400</td>
<td>0x4000 37FF</td>
</tr>
<tr>
<td>Watchdog Timer 2</td>
<td>WDT2_</td>
<td>0x4000 3800</td>
<td>0x4000 3BFF</td>
</tr>
<tr>
<td>Security Monitor</td>
<td>SMON_</td>
<td>0x4000 4000</td>
<td>0x4000 43FF</td>
</tr>
<tr>
<td>SIMO Controller</td>
<td>SIMO_</td>
<td>0x4000 4400</td>
<td>0x4000 47FF</td>
</tr>
<tr>
<td>DVS Controller</td>
<td>DVS_</td>
<td>0x4000 4800</td>
<td>0x4000 4BFF</td>
</tr>
<tr>
<td>AES Keys</td>
<td>AES_</td>
<td>0x4000 5000</td>
<td>0x4000 53FF</td>
</tr>
<tr>
<td>Real-Time Clock</td>
<td>RTC_</td>
<td>0x4000 6000</td>
<td>0x4000 63FF</td>
</tr>
<tr>
<td>Wake Up Timer</td>
<td>WUT_</td>
<td>0x4000 6400</td>
<td>0x4000 67FF</td>
</tr>
<tr>
<td>Power Sequencer</td>
<td>PWRSEQ_</td>
<td>0x4000 6800</td>
<td>0x4000 6BFF</td>
</tr>
<tr>
<td>Misc. Control Registers</td>
<td>MCR_</td>
<td>0x4000 6C00</td>
<td>0x4000 6FFF</td>
</tr>
<tr>
<td>GPIO Port 0</td>
<td>GPIO0_</td>
<td>0x4000 8000</td>
<td>0x4000 8FFF</td>
</tr>
<tr>
<td>GPIO Port 1</td>
<td>GPIO1_</td>
<td>0x4000 9000</td>
<td>0x4000 9FFF</td>
</tr>
<tr>
<td>Timer 0</td>
<td>TMR0_</td>
<td>0x4001 0000</td>
<td>0x4001 0FFF</td>
</tr>
<tr>
<td>Peripheral Register Name</td>
<td>Register Prefix</td>
<td>APB Base Address</td>
<td>APB End Address</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Timer 1</td>
<td>TMR1_</td>
<td>0x4001 1000</td>
<td>0x4001 1FFF</td>
</tr>
<tr>
<td>Timer 2</td>
<td>TMR2_</td>
<td>0x4001 2000</td>
<td>0x4001 2FFF</td>
</tr>
<tr>
<td>Timer 3</td>
<td>TMR3_</td>
<td>0x4001 3000</td>
<td>0x4001 3FFF</td>
</tr>
<tr>
<td>Timer 4</td>
<td>TMR4_</td>
<td>0x4001 4000</td>
<td>0x4001 4FFF</td>
</tr>
<tr>
<td>Timer 5</td>
<td>TMR5_</td>
<td>0x4001 5000</td>
<td>0x4001 5FFF</td>
</tr>
<tr>
<td>HTimer 0</td>
<td>HTMRO_</td>
<td>0x4001 B000</td>
<td>0x4001 BFFF</td>
</tr>
<tr>
<td>HTimer 1</td>
<td>HTMR1_</td>
<td>0x4001 C000</td>
<td>0x4001 CFFF</td>
</tr>
<tr>
<td>I2C 0 (bus 0)</td>
<td>I2C0_BUS0_</td>
<td>0x4001 D000</td>
<td>0x4001 DFFF</td>
</tr>
<tr>
<td>I2C 1 (bus 0)</td>
<td>I2C1_BUS0_</td>
<td>0x4001 E000</td>
<td>0x4001 EFFF</td>
</tr>
<tr>
<td>I2C 2 (bus 0)</td>
<td>I2C2_BUS0_</td>
<td>0x4001 F000</td>
<td>0x4001 FFFF</td>
</tr>
<tr>
<td>SPIXF Master</td>
<td>SPIXF_</td>
<td>0x4002 6000</td>
<td>0x4002 6FFF</td>
</tr>
<tr>
<td>SPIXF Master Controller</td>
<td>SPIXFC_</td>
<td>0x4002 7000</td>
<td>0x4002 7FFF</td>
</tr>
<tr>
<td>Standard DMA 0</td>
<td>DMA0_</td>
<td>0x4002 8000</td>
<td>0x4002 8FFF</td>
</tr>
<tr>
<td>Flash Controller 0</td>
<td>FLC0_</td>
<td>0x4002 9000</td>
<td>0x4002 93FF</td>
</tr>
<tr>
<td>Flash Controller 1</td>
<td>FLC1_</td>
<td>0x4002 9400</td>
<td>0x4002 97FF</td>
</tr>
<tr>
<td>Instruction-Cache Controller 0</td>
<td>ICC0_</td>
<td>0x4002 A000</td>
<td>0x4002 A3FF</td>
</tr>
<tr>
<td>Instruction Cache Controller 1</td>
<td>ICC1_</td>
<td>0x4002 A800</td>
<td>0x4002 A8FF</td>
</tr>
<tr>
<td>Instruction Cache Controller XIP</td>
<td>SFCC_</td>
<td>0x4002 F000</td>
<td>0x4002 FFFF</td>
</tr>
<tr>
<td>External Memory Cache Controller</td>
<td>SRCC_</td>
<td>0x4003 3000</td>
<td>0x4003 3FFF</td>
</tr>
<tr>
<td>Analog to Digital Converter</td>
<td>ADC_</td>
<td>0x4003 4000</td>
<td>0x4003 4FFF</td>
</tr>
<tr>
<td>Standard DMA 1</td>
<td>DMA1_</td>
<td>0x4003 5000</td>
<td>0x4003 5FFF</td>
</tr>
<tr>
<td>Reserved</td>
<td>-</td>
<td>0x4003 6000</td>
<td>0x4003 6FFF</td>
</tr>
<tr>
<td>Reserved</td>
<td>-</td>
<td>0x4003 7000</td>
<td>0x4003 7FFF</td>
</tr>
<tr>
<td>SPIXR Master Controller</td>
<td>SPIXR_</td>
<td>0x4003 A000</td>
<td>0x4003 AFFF</td>
</tr>
<tr>
<td>Pulse Train Engine (bus 0)</td>
<td>PTG_BUS0_</td>
<td>0x4003 C000</td>
<td>0x4003 CFFF</td>
</tr>
<tr>
<td>1-Wire</td>
<td>OWM_</td>
<td>0x4003 D000</td>
<td>0x4003 DFFF</td>
</tr>
<tr>
<td>Semaphores</td>
<td>SEMA_</td>
<td>0x4003 E000</td>
<td>0x4003 EFFF</td>
</tr>
<tr>
<td>UART 0</td>
<td>UART0_</td>
<td>0x4004 2000</td>
<td>0x4004 2FFF</td>
</tr>
<tr>
<td>UART 1</td>
<td>UART1_</td>
<td>0x4004 3000</td>
<td>0x4004 3FFF</td>
</tr>
<tr>
<td>UART 2</td>
<td>UART2_</td>
<td>0x4004 4000</td>
<td>0x4004 4FFF</td>
</tr>
<tr>
<td>SPI1</td>
<td>SPI1_</td>
<td>0x4004 6000</td>
<td>0x4004 6FFF</td>
</tr>
<tr>
<td>SPI2</td>
<td>SPI2_</td>
<td>0x4004 7000</td>
<td>0x4004 7FFF</td>
</tr>
<tr>
<td>Audio Subsystem</td>
<td>AUDIO_</td>
<td>0x4004 C000</td>
<td>0x4004 CFFF</td>
</tr>
<tr>
<td>TRNG</td>
<td>TRNG_</td>
<td>0x4004 D000</td>
<td>0x4004 DFFF</td>
</tr>
<tr>
<td>BTLE Registers and IQ RAMs</td>
<td>BTLE_</td>
<td>0x4005 0000</td>
<td>0x4005 FFFF</td>
</tr>
<tr>
<td>I2C 0 (bus 1)</td>
<td>I2C0_BUS1_</td>
<td>0x4011 D000</td>
<td>0x4011 DFFF</td>
</tr>
<tr>
<td>I2C 1 (bus 1)</td>
<td>I2C1_BUS1_</td>
<td>0x4011 E000</td>
<td>0x4011 EFFF</td>
</tr>
<tr>
<td>I2C 2 (bus 1)</td>
<td>I2C2_BUS1_</td>
<td>0x4011 F000</td>
<td>0x4011 FFFF</td>
</tr>
<tr>
<td>Pulse Train Engine (bus 1)</td>
<td>PTG_BUS1_</td>
<td>0x4013 C000</td>
<td>0x4013 CFFF</td>
</tr>
</tbody>
</table>

### 3.5.2 AHB Peripheral Base Address Map

Table 3-2 contains the base address for each of the AHB 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 AHB peripheral base address plus the registers offset.
Table 3-2: AHB Peripheral Base Address Map

<table>
<thead>
<tr>
<th>AHB Peripheral Register Name</th>
<th>Register Prefix</th>
<th>AHB Base Address</th>
<th>AHB End Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>USB Hi-Speed Host</td>
<td>USBS_</td>
<td>0x400B 1000</td>
<td>0x400B 1FFF</td>
</tr>
<tr>
<td>SDIO/SDHC Controller (AHB)</td>
<td>SDHC_</td>
<td>0x400B 6000</td>
<td>0x400B 6FFF</td>
</tr>
<tr>
<td>SPIXF Master Controller FIFO</td>
<td>SPIXF_FIFO_</td>
<td>0x400B C000</td>
<td>0x400B CFFF</td>
</tr>
<tr>
<td>SPI0</td>
<td>SPI0_</td>
<td>0x400B E000</td>
<td>0x400B E3FF</td>
</tr>
</tbody>
</table>

3.6 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 memory arrays. 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 memory. These check bits are then stored along with the data. During a read, both the data and check bits are used to determine if one or more bits have become corrupt. If a single bit has been corrupted this can be corrected. If two bits have been corrupted, it will be detected, but not corrected.

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

3.6.1 SRAM

To integrate the ECC SEC-DED module into a RAM, there must be a secondary RAM instance to store the check bits. In the case of a 32-bit wide RAM, 7 check bits are needed. The secondary check bit RAM can hold the 7 check bits in each byte, therefore needs ¼ the number of words as the RAM itself. Also, the address sent to the check bit RAM is divided by 4 to map the 32-bit data words to 8-bit check bit addresses.

For example, a 32-bit by 8192 word RAM would need a 32-bit by 2048 word sized secondary RAM instance. When ECC is enabled, each system RAM module requires an appropriately sized secondary RAM.

3.6.2 FLASH

The flash lines will need to be larger in width to so that the check bits can be placed on the same line as the data. 128 data bits would require 9 check bits for a total of 137 bits. This will require changes to the flash controller to write these check bits each time any of the 128 data bits on a line are changed. Also, any write that is less than the full width of the data will require a read first to fetch the other unchanging bytes to properly calculate the check bits. This will cause an extra delay.

3.6.3 Cache

Any type of ECC error (single or double) is treated as a cache miss. There are separate ECC check bits for both the data RAM and tag RAM inside the cache.

3.6.4 Limitations

Any read from non-initialized memory could trigger an ECC error since the random check bits will most likely not match the random data bits. Writing the memory to all zeroes at bootup can prevent this at the expense of the time required.
### Table 3-3: Error Correction Coding (ECC) Enable Register

<table>
<thead>
<tr>
<th>Error Correction Coding Enable</th>
<th>GCR_ECC_EN</th>
<th>[0x6C00]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>--------</td>
</tr>
</tbody>
</table>
| 31:13 | - | RO | 0 | Reserved for Future Use  
Do not modify this field. |
| 12 | fl1eccen | R/W | 0 | Flash1 ECC Enable  
0: Disable  
1: Enable |
| 11 | fl0eccen | R/W | 0 | Flash0 ECC Enable  
0: Disable  
1: Enable |
| 10 | icspixfeccen | R/W | 0 | ICacheXIP ECC Enable  
0: Disable  
1: Enable |
| 9 | ic1eccen | R/W | 0 | ICache1 ECC Enable  
0: Disable  
1: Enable |
| 8 | ic0eccen | R/W | 0 | ICache0 ECC Enable  
0: Disable  
1: Enable |
| 7:6 | - | RO | 0 | Reserved for Future Use  
Do not modify this field. |
| 5 | sysram5en | R/W | 0 | Sysram5 ECC Enable  
0: Disable  
1: Enable |
| 4 | sysram4en | R/W | 0 | Sysram4 ECC Enable  
0: Disable  
1: Enable |
| 3 | sysram3en | R/W | 0 | Sysram3 ECC Enable  
0: Disable  
1: Enable |
| 2 | sysram2en | R/W | 0 | Sysram2 ECC Enable  
0: Disable  
1: Enable |
| 1 | sysram1en | R/W | 0 | Sysram1 ECC Enable  
0: Disable  
1: Enable |
| 0 | sysram0en | R/W | 0 | Sysram0 ECC Enable  
0: Disable  
1: Enable |

### Table 3-4: Error Correction Coding (ECC) Error Register

<table>
<thead>
<tr>
<th>Error Correction Coding Error</th>
<th>GCR_ECC_ER</th>
<th>[0x0064]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>--------</td>
</tr>
</tbody>
</table>
| 31:13 | - | RO | 0 | Reserved for Future Use  
Do not modify this field. |
<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>fl1eccerr</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>ECC Flash1 Error</strong>&lt;br&gt;Indicates an ECC error in the Flash1 bank. Write to 1 to clear.&lt;br&gt;0: No Error&lt;br&gt;1: Error</td>
</tr>
<tr>
<td>11</td>
<td>fl0eccerr</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>ECC Flash0 Error</strong>&lt;br&gt;Indicates an ECC error in the Flash0 bank. Write to 1 to clear.&lt;br&gt;0: No Error&lt;br&gt;1: Error</td>
</tr>
<tr>
<td>10</td>
<td>icspixfeccerr</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>ECC SPIXF Instruction Cache Error</strong>&lt;br&gt;Indicates an ECC error in SPIXF Instruction Cache. Write to 1 to clear.&lt;br&gt;0: No Error&lt;br&gt;1: Error</td>
</tr>
<tr>
<td>9</td>
<td>ic1eccerr</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>ECC Instruction Cache 1 Error</strong>&lt;br&gt;Indicates an ECC error in Instruction Cache 1. Write to 1 to clear.&lt;br&gt;0: No Error&lt;br&gt;1: Error</td>
</tr>
<tr>
<td>8</td>
<td>ic0eccerr</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>ECC Instruction Cache 0 Error</strong>&lt;br&gt;Indicates an ECC error in Instruction Cache 0. Write to 1 to clear.&lt;br&gt;0: No Error&lt;br&gt;1: Error</td>
</tr>
<tr>
<td>7:6</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>5</td>
<td>sysram5eccerr</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>ECC Sysram5 Error</strong>&lt;br&gt;Indicates an ECC error in the Sysram5 block. Write to 1 to clear.&lt;br&gt;0: No Error&lt;br&gt;1: Error</td>
</tr>
<tr>
<td>4</td>
<td>sysram4eccerr</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>ECC Sysram4 Error</strong>&lt;br&gt;Indicates an ECC error in the Sysram4 block. Write to 1 to clear.&lt;br&gt;0: No Error&lt;br&gt;1: Error</td>
</tr>
<tr>
<td>3</td>
<td>sysram3eccerr</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>ECC Sysram3 Error</strong>&lt;br&gt;Indicates an ECC error in the Sysram3 block. Write to 1 to clear.&lt;br&gt;0: No Error&lt;br&gt;1: Error</td>
</tr>
<tr>
<td>2</td>
<td>sysram2eccerr</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>ECC Sysram2 Error</strong>&lt;br&gt;Indicates an ECC error in the Sysram2 block. Write to 1 to clear.&lt;br&gt;0: No Error&lt;br&gt;1: Error</td>
</tr>
<tr>
<td>1</td>
<td>sysram1eccerr</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>ECC Sysram1 Error</strong>&lt;br&gt;Indicates an ECC error in the Sysram1 block. Write to 1 to clear.&lt;br&gt;0: No Error&lt;br&gt;1: Error</td>
</tr>
<tr>
<td>0</td>
<td>sysram0eccerr</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>ECC Sysram00 Error</strong>&lt;br&gt;Indicates an ECC error in the Sysram0 block. Write to 1 to clear.&lt;br&gt;0: No Error&lt;br&gt;1: Error</td>
</tr>
<tr>
<td>Correctable Error Detected</td>
<td>GCR_ECC_CED</td>
<td>[0x0068]</td>
<td>Bits</td>
<td>Field</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------</td>
<td>---------</td>
<td>------</td>
<td>-------</td>
</tr>
</tbody>
</table>
| 31:13                      | -          | RO      | 0    |       |        |       | Reserved for Future Use  
Do not modify this field. |
| 12                         | fl1eccnded | R/WIC   | 0    |       |        |       | ECC Flash1 Correctable Error Detected  
Indicates a single bit correctable error in the Flash1 bank. Write to 1 to clear.  
0: No single bit error detected  
1: Correctable |
| 11                         | fl0eccnded | R/WIC   | 0    |       |        |       | ECC Flash0 Correctable Error Detected  
Indicates a single bit correctable error in the Flash0 bank. Write to 1 to clear.  
0: No single bit error detected  
1: Correctable |
| 10                         | icspixfeccnded | R/WIC | 0 |       |        |       | ECC SPIXF Correctable Error Detected  
Indicates a single bit correctable error in SPIXF Instruction Cache. Write to 1 to clear.  
0: No single bit error detected  
1: Correctable |
| 9                          | ic1eccnded | R/WIC   | 0    |       |        |       | ECC Instruction Cache 1 Correctable Error Detected  
Indicates a single bit correctable error in Instruction Cache 1. Write to 1 to clear.  
0: No single bit error detected  
1: Correctable |
| 8                          | ic0eccnded | R/WIC   | 0    |       |        |       | ECC Instruction Cache 0 Correctable Error Detected  
Indicates a single bit correctable error in Instruction Cache 0. Write to 1 to clear.  
0: No single bit error detected  
1: Correctable |
| 7                          | -          | RO      | 0    |       |        |       | Reserved for Future Use  
Do not modify this field. |
| 6                          | -          | RO      | 0    |       |        |       | Reserved for Future Use  
Do not modify this field. |
| 5                          | sysram5ecc_ced | R/WIC | 0 |       |        |       | ECC Sysram5 Correctable Error Detected  
Indicates a single bit correctable error in the Sysram5 block. Write to 1 to clear.  
0: No single bit error detected  
1: Correctable |
| 4                          | sysram4ecc_ced | R/WIC | 0 |       |        |       | ECC Sysram4 Correctable Error Detected  
Indicates a single bit correctable error in the Sysram 4 block. Write to 1 to clear.  
0: No single bit error detected  
1: Correctable |
| 3                          | sysram3ecc_ced | R/WIC | 0 |       |        |       | ECC Sysram3 Correctable Error Detected  
Indicates a single bit correctable error in the Sysram 3 block. Write to 1 to clear.  
0: No single bit error detected  
1: Correctable |
| 2                          | sysram2ecc_ced | R/WIC | 0 |       |        |       | ECC Sysram2 Correctable Error Detected  
Indicates a single bit correctable error in the Sysram 2 block. Write to 1 to clear.  
0: No single bit error detected  
1: Correctable |
### Correctable Error Detected

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>sysram1ecc_ced</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>ECC Sysram1 Correctable Error Detected</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indicates a single bit correctable error in the Sysram 1 block. Write to 1 to clear.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No single bit error detected</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Correctable</td>
</tr>
<tr>
<td>0</td>
<td>sysram0ecc_ced</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>ECC Sysram0 Correctable Error Detected</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indicates a single bit correctable error in the Sysram 0 block. Write to 1 to clear.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No single bit error detected</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Correctable</td>
</tr>
</tbody>
</table>

### Table 3-6: Error Correction Coding (ECC) Interrupt Enable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:13</td>
<td></td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>12</td>
<td>fl1eccirqen</td>
<td>R/W</td>
<td>0</td>
<td><strong>ECC Flash1 Interrupt Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When set, indicates that ECC is enabled for the block and interrupts occur upon a detected error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>11</td>
<td>fl0eccirqen</td>
<td>R/W</td>
<td>0</td>
<td><strong>ECC Flash0 Interrupt Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When set, indicates that ECC is enabled for the block and interrupts occur upon a detected error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>10</td>
<td>icspixfeccirqen</td>
<td>R/W</td>
<td>0</td>
<td><strong>ECC SPIXF Instruction Cache Interrupt Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When set, indicates that ECC is enabled for the block and interrupts occur upon a detected error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>9</td>
<td>ic1eccirqen</td>
<td>R/W</td>
<td>0</td>
<td><strong>ECC Instruction Cache 1 Interrupt Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When set, indicates that ECC is enabled for the block and interrupts occur upon a detected error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>8</td>
<td>ic0eccirqen</td>
<td>R/W</td>
<td>0</td>
<td><strong>ECC Instruction Cache 0 Interrupt Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When set, indicates that ECC is enabled for the block and interrupts occur upon a detected error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>7:6</td>
<td></td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>5</td>
<td>sysram5irqen</td>
<td>R/W</td>
<td>0</td>
<td><strong>ECC Sysram5 Interrupt Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When set, indicates that the interrupt is enabled for occurrence upon a detected error in the Sysram5 block if GCR_ECC_EN.sysram5en is set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
</tbody>
</table>
### Error Correction Coding Interrupt Enable

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>sysram4irqen</td>
<td>R/W</td>
<td>0</td>
<td><strong>ECC Sysram4 Interrupt Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When set, indicates that the interrupt is enabled for occurrence upon a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>detected error in the Sysram4 block if <code>GCR_ECC_EN.sysram4en</code> is set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>0</td>
<td>sysram3irqen</td>
<td>R/W</td>
<td>0</td>
<td><strong>ECC Sysram3 Interrupt Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When set, indicates that the interrupt is enabled for occurrence upon a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>detected error in the Sysram3 block and <code>GCR_ECC_EN.sysram3en</code> is set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>0</td>
<td>sysram2irqen</td>
<td>R/W</td>
<td>0</td>
<td><strong>ECC Sysram2 Interrupt Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When set, indicates that the interrupt is enabled for occurrence upon a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>detected error in the Sysram2 block and <code>GCR_ECC_EN.sysram2en</code> is set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>0</td>
<td>sysram1irqen</td>
<td>R/W</td>
<td>0</td>
<td><strong>ECC Sysram1 Interrupt Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When set, indicates that the interrupt is enabled for occurrence upon a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>detected error in the Sysram1 block and <code>GCR_ECC_EN.sysram1en</code> is set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>0</td>
<td>sysram0irqen</td>
<td>R/W</td>
<td>0</td>
<td><strong>ECC Sysram0 Interrupt Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When set, indicates that the interrupt is enabled for occurrence upon a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>detected error in the Sysram0 block and <code>GCR_ECC_EN.sysram0en</code> is set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
</tbody>
</table>

### Error Correction Coding Address (ECC) Address Register

<table>
<thead>
<tr>
<th>Error Correction Coding Address</th>
<th>GCR_ECC_ERRAD</th>
<th>[0x0070]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>tagramerr</td>
<td>R</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Data depends on which block has reported the error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If sysram, fl0, or fl1, then this bit(s) represents</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bit 31 of the AMBA address of read which produced the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>error. If the error is from one of the caches, then</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>this bit is set as shown below:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: No Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Tag_Error. The error is in the TAG RAM</td>
</tr>
<tr>
<td>30</td>
<td>tagrambank</td>
<td>R</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Data depends on which block has reported the error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If sysram, fl0, or fl1, then this bit(s) represents</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bit 30 of the AMBA address of read which produced the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>error. If the error is from one of the caches, then</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>this bit is set as shown below:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: Error is in TAG RAM Bank 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Error is in TAG RAM Bank 1</td>
</tr>
<tr>
<td>29:16</td>
<td>tagramaddr</td>
<td>R</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Data depends on which block has reported the error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If sysram, fl0, or fl1, then this bit(s) represents</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>bits 29:16 of the AMBA address of read which produced</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the error. If the error is from one of the caches,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>then this bit is set as shown below:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[TAG ADDRESS]: Represents the TAG RAM Address</td>
</tr>
<tr>
<td>Error Correction Coding Address</td>
<td>GCR_ECC_ERRAD</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------</td>
<td>-------------</td>
<td></td>
</tr>
</tbody>
</table>
| **15** dataramerr            | R 0          | ECC Error Address/DATA RAM Error Address  
Data depends on which block has reported the error. If sysram, fl0, or fl1, then this bit(s) represents bit 15 of the AMBA address of read which produced the error. If the error is from one of the caches, then this bit is set as shown below:  
0: No Error  
1: DATA RAM Error. The error is in the Data RAM |
| **14** datarambank           | R 0          | ECC Error Address/DATA RAM Error Bank  
Data depends on which block has reported the error. If sysram, fl0, or fl1, then this bit(s) represents bit 14 of the AMBA address of read which produced the error. If the error is from one of the caches, then this bit is set as shown below:  
0: Error is in DATA RAM Bank 0  
1: Error is in DATA RAM Bank 1 |
| **13:0** dataramaddr         | R 0          | ECC Error Address/TAG RAM Error Address  
Data depends on which block has reported the error. If sysram, fl0, or fl1, then this bit(s) represents bits 13:0 of the AMBA address of read which produced the error. If the error is from one of the caches, then this bit is set as shown below:  
[DATA ADDRESS]: Represents the DATA RAM Error Address |
4. 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.

The selected System Oscillator (SYS OSC) is the clock source for most internal blocks. Select SYS OSC from the following clock sources:

- 96MHz Internal High-Frequency Oscillator
- 60MHz Low-Power Internal Oscillator
- 7.3728MHz Internal Oscillator
  - Selectable for UART baud rate generation
- 8kHz Internal Ultra-Low Power Nano-Ring Oscillator
- 32.768kHz External Crystal Oscillator
  - Clock source for the Real-Time Clock (RTC)
- 32MHz External Crystal Oscillator
  - Clock source for the Bluetooth 5 radio

The selected SYS OSC is the input to the system oscillator prescaler to generate the System Clock (SYS CLK). The system oscillator prescaler divides SYS OSC by a prescaler using the GCR_CLK_CTRL.sysclk_prescale field as shown in Equation 4-1.

Equation 4-1: System Clock Scaling

\[
SYS\_CLK = \frac{SYS\_OSC}{2^{sysclk\_prescale}}
\]

GCR_CLK_CTRL.sysclk_prescale 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 cores and is used to generate the following internal clocks as shown below:

- Advanced High-Performance Bus (AHB) Clock
  - HCLK = SYS CLK
- Advanced Peripheral Bus (APB) Clock,
  - PCLK = SYS CLK / 2
- Always On Domain (AOD) Clock,
  - AODCLK = PCLK / 2^{GCR_PCLK_DIV.aondiv}
  - GCR_PCLK_DIV.aondiv is selectable from 0 to 3 for divisors of 1, 2, 4 or 8

There are additional internal clocks that are generated. These clocks are independent of SYS OSC and SYS CLK as follows:

- The USB PHY uses the 96MHz oscillator
- The SDHC/SDIO controller uses the high speed 96MHz oscillator divided by 2
- The Audio interface uses the high speed 96MHz oscillator
- The RTC uses the 32.768kHz oscillator
- (MAX32666/MAX32668 only) The Trust Protection Unit (TPU) uses the 60MHz Low-Power Internal Oscillator

All oscillators are reset to default at Power-On Reset (POR) and System Reset. Oscillator status is not reset by a Soft Reset or Peripheral Reset.

4.1 Oscillator Sources and Clock Switching

On Power-On Reset (POR) and System Reset, all oscillator states are reset to the default: The 60MHz and 8kHz oscillators are enabled, while the 96MHz, 32MHz 32.768kHz and 7.3728MHz oscillators are disabled. Oscillators are not reset on Soft Reset or Peripheral Reset.
Figure 4-1: Clock Block Diagram
### 4.1.1 Oscillator Implementation

Before using any oscillator, the desired oscillator must first be enabled by setting the oscillator’s enable bit in the `GCR_CLK_CTRL` register. Once an oscillator’s enable bit is set, the oscillator’s ready bit must read 1 prior to attempting to use the oscillator as a system oscillator source. The oscillator ready status flags are contained in the `GCR_CLK_CTRL` register.

Once the corresponding oscillator ready bit is set, the oscillator can then be selected as SYS_OSC by configuring the Clock Source Select field (`GCR_CLK_CTRL.sysosc_sel`).

Any time firmware changes SYS_OSC by changing `GCR_CLK_CTRL.sysosc_sel`, the Clock Ready bit `GCR_CLK_CTRL.sysosc_rdy` is automatically cleared to indicate that a SYS_OSC switchover is in progress. When switchover is complete, `GCR_CLK_CTRL.sysosc_rdy` is set to 1 by hardware indicating the oscillator selected is ready for use.

#### 4.1.2 96MHz Internal Main High-Speed Oscillator

The devices are available with a 96MHz internal high-speed oscillator. This is the fastest oscillator and draws the most power.

This oscillator is also used by the USB PHY, Audio subsystem, and the SDHC. If the USB or SDHC or Audio subsystem is enabled, the 96MHz oscillator must be enabled, independent of the selection of SYS_OSC.

Optionally, this oscillator can be powered down automatically when in DEEPSLEEP mode by setting register bit `GCR_PMR.hircmpd`.

#### 4.1.3 60MHz Low Power Internal Oscillator

This is a low-power internal oscillator that can be selected as SYS_OSC. This oscillator is automatically selected as SYS_OSC after a System Reset or POR. This oscillator is also the dedicated clock for the TPU. If the TPU is enabled, the 60MHz internal oscillator must be enabled, independent of the selection of SYS_OSC. When used as the TPU clock it can be divided by 2.

#### 4.1.4 32MHz Bluetooth Radio Oscillator

This is the oscillator that directly drives the Bluetooth radio. It can also be selected as SYS_OSC. It is important to use the correct capacitor values on the PCB when connecting the crystal. Figure 4-2 depicts the method to determine the capacitor values $C_{LIN}$ and $C_{LOUT}$. In order to enable this oscillator, the Bluetooth LDOs must be enabled by setting the `GCR_BTLE_LDOCR.ldorxen`, `GCR_BTLE_LDOCR.ildotxen`, `GCR_BTLE_LDOCR.ldorxvsel`, `GCR_BTLE_LDOCR.ildotxvsel` register fields.
**Figure 4-2: Example 32MHz Crystal Capacitor Determination**

The crystal load, $C_L$, as specified in the MAX32665-MAX32668 datasheet Electrical Characteristics Table is required to be 12pF. Therefore, the total capacitance seen by the crystal must equal $C_L$.

$$C_L = \frac{(C_{HFXIN} \times C_{HFXOUT})}{(C_{HFXIN} + C_{HFXOUT})}$$

Assume that $C_{LIN} = C_{LOUT}$.

Assume the device pin capacitance of the HFXOUT and HFXIN pins respectively is 4pF each. This specification is outlined in the Electrical Characteristics Table of the MAX32665-MAX32668 datasheet as $C_{HFX_PIN}$.

Assume the circuit board stray capacitance represented in the diagram by $C_{STIN}$ and $C_{STOUT} = 0.5$pf each.

Solve for $C_{LOUT}$,

$$C_{LOUT} = 19.5\text{pF} = C_{LIN}$$

Choose 20pF for $C_{LOUT} = C_{LIN}$

### 4.1.5 7.3728MHz Internal Oscillator

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 allow the targeted UART baud rate.

Firmware selection of the voltage that controls this oscillator is controlled by the register bit $GCR_CLK_CTRL.hirc7m_vs$. The internal CPU core supply voltage ($V_{CORE}$) is the default option. The external pin $V_{DDA}$ can also be selected. The $V_{DDA}$ pin goes to an internal 1V regulator that also provides the analog supply voltage for this device.

This oscillator can optionally be automatically powered down when in DEEPSLEEP mode by setting register bit $GCR_PMR.hirc7mpd$.

This oscillator is disabled by default at power-up.

### 4.1.6 32.768kHz External Crystal Oscillator

This is a very low power internal oscillator that can be selected as SYS_OSC. This oscillator can optionally use a 32.768kHz input clock instead of an external crystal. The internal 32.768kHz clock is available as an output on GPIO as an alternate function (SQWOUT).

This oscillator is the dedicated clock for the Real-Time Clock (RTC). If the RTC is enabled, the 32.768kHz external oscillator must be enabled, independent of the selection of SYS_OSC. This oscillator is disabled at power-up.

### 4.1.7 8kHz Ultra-Low Power Nano-Ring Internal Oscillator

This is an ultra-low power internal oscillator that can be selected as SYS_OSC.

This oscillator is enabled at power-up and cannot be disabled by firmware.
4.2 Operating Modes

The MAX32665—MAX32668 provides four operating modes:

- ACTIVE
- SLEEP
- DEEPSLEEP
- BACKUP

ACTIVE is the highest performance operating mode. Any low power state can wake up to ACTIVE by a wakeup event shown in Table 4-1.

Table 4-1: Wakeup Sources

<table>
<thead>
<tr>
<th>OPERATING MODE</th>
<th>WAKEUP SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLEEP</td>
<td>Interrupts (RTC, GPIO, USB, Comparators), RSTN assertion, Wakeup Timer.</td>
</tr>
<tr>
<td>DEEPSLEEP</td>
<td>Interrupts (RTC, GPIO, USB, Comparators), RSTN assertion, Wakeup Timer.</td>
</tr>
<tr>
<td>BACKUP</td>
<td>Interrupts (RTC, GPIO, USB, Comparators), RSTN assertion, Wakeup Timer.</td>
</tr>
</tbody>
</table>

The Arm Cortex-M family of CPUs have two built-in low power modes, designated SLEEP and DEEPSLEEP. Implementation of these low-power modes are specific to the microcontroller’s design. These modes are enabled using the System Control Register (SCR), an Arm Cortex System Control Block register. Write register bit SCR.sleepdeep to select the low power mode as shown in the pseudocode below.

```
SCR.sleepdeep = 0; // SLEEP mode enabled
SCR.sleepdeep = 1; // DEEPSLEEP mode enabled
```

Once enabled, the device enters the enabled low power mode when either a WFI (Wait For Interrupt) or WFE (Wait For Event) instruction is executed.

Refer to the Arm Cortex-M4 core reference for more information on SCR.

4.2.1 ACTIVE Mode

This is the highest performance mode. All internal clocks, registers, memory, and peripherals are enabled. The CPU is running and executing application code. All oscillators are available.

Dynamic clocking allows firmware to selectively enable or disable clocks and power to individual peripherals, providing the optimal mix of high-performance and power conservation. Internal RAM that can be enabled, disabled, or placed in low-power RAM Retention Mode include data SRAM memory blocks, on-chip caches, and on-chip FIFOs.

4.2.2 SLEEP Low Power Mode

This is a low power mode that suspends the CPU with a fast wakeup time to ACTIVE mode. It is like ACTIVE mode except the CPU clock is disabled, which temporarily prevents the CPU from executing code. All oscillators remain active if enabled and the Always On Domain (AOD) and RAM retention is enabled.

The device returns to ACTIVE mode from any internal or external interrupt.

The following pseudocode places the device in SLEEP mode:

```
SCR.sleepdeep = 0; // SLEEP mode enabled
WFI (or WFE);      // Enter the low power mode enabled by SCR.sleepdeep
```

Figure 4-3, below, shows the clocks available and blocks disabled during SLEEP mode.
Figure 4-3: SLEEP Mode Clock Control
4.2.3 DEEPSLEEP Low Power Mode

All internal clocks, except the 8kHz, are gated off. SYS_OSC is gated off, so the two main bus clocks PCLK and HCLK are inactive. The CPU state is retained. The 32kHz oscillator can be enabled via firmware.

Because the main bus clocks are disabled, all peripherals are inactive except for the RTC which has its own independent oscillator. Only the RTC, USB wakeup or external interrupt can return the device to ACTIVE. The Watchdog Timers are inactive in this mode.

All internal register contents and all RAM contents are preserved. The GPIO pins retain their state in this mode.

To enter DEEPSLEEP mode,

```
SCR.sleepdeep = 1; // DEEPSLEEP mode enabled
WFI (or WFE);      // Enter DEEPSLEEP mode
```
Figure 4-4: DEEPSLEEP Clock Control
4.2.4 BACKUP Low Power Mode

This is the lowest power operating mode. All oscillators are disabled except for the 8kHz and the 32kHz oscillator. The 32kHz oscillator is firmware controlled. SYS_OSC is gated off, so PCLK and HCLK are inactive. The CPU state is not maintained.

Only the RTC can operate in BACKUP mode. The AoD and RAM retention can optionally be set to automatically disable (and clear) themselves when entering this mode. RAM may be optionally retained. The amount of RAM retained is controlled by appropriately setting the PWRSEQ_LPCN.ramret register bits.

BACKUP mode supports the same wakeup sources as DEEPSLEEP mode.

To immediately enter BACKUP mode, write GCR_PMR.mode = 0b100.

Figure 4-5, shows the clock control during BACKUP mode.
Figure 4-5: BACKUP Mode Clock Control
4.3 Device Resets

Four device resets are available – Peripheral Reset, Soft Reset, System Reset, and 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 VDDA and VCOREA.

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 4-2: Reset and Low Power Mode Effects shows the effects of the four reset types and the four power modes.
### Table 4-2: Reset and Low Power Mode Effects

<table>
<thead>
<tr>
<th></th>
<th>Peripheral Reset</th>
<th>Soft Reset</th>
<th>System Reset</th>
<th>POR</th>
<th>ACTIVE Mode</th>
<th>SLEEP Mode</th>
<th>DEEPSLEEP Mode</th>
<th>BACKUP Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCR</td>
<td>-</td>
<td>-</td>
<td>Reset</td>
<td>Reset</td>
<td>R</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8kHz Osc</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>32kHz Osc</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Off</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7.3728 MHz Osc</td>
<td>-</td>
<td>-</td>
<td>Off</td>
<td>Off</td>
<td>R</td>
<td>-</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>60MHz Osc</td>
<td>-</td>
<td>-</td>
<td>On²</td>
<td>On²</td>
<td>R</td>
<td>-</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>32MHz Osc</td>
<td>-</td>
<td>-</td>
<td>Off</td>
<td>Off</td>
<td>R</td>
<td>-</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>96MHz Osc</td>
<td>-</td>
<td>-</td>
<td>Off</td>
<td>Off</td>
<td>R</td>
<td>-</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>SYS_CLK</td>
<td>On</td>
<td>On</td>
<td>On²</td>
<td>On²</td>
<td>On</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>CPU0, CPU1 Clock</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>VREGO_A</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>VREGO_B</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>FW</td>
</tr>
<tr>
<td>VREGO_C</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>VREGO_D</td>
<td>-</td>
<td>-</td>
<td>On</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Bluetooth LDOs</td>
<td>-</td>
<td>-</td>
<td>Reset</td>
<td>Reset</td>
<td>R</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>CPU</td>
<td>-</td>
<td>-</td>
<td>Reset</td>
<td>Reset</td>
<td>R</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>RPU</td>
<td>-</td>
<td>-</td>
<td>Reset</td>
<td>Reset</td>
<td>R</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>WDTO/1/2</td>
<td>-</td>
<td>-</td>
<td>Reset</td>
<td>Reset</td>
<td>R</td>
<td>-</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>GPIO</td>
<td>-</td>
<td>Reset</td>
<td>Reset</td>
<td>Reset</td>
<td>R</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Other Peripherals</td>
<td>Reset</td>
<td>Reset</td>
<td>Reset</td>
<td>Reset</td>
<td>R</td>
<td>-</td>
<td>Off</td>
<td>-</td>
</tr>
<tr>
<td>Always-On Domain¹</td>
<td>-</td>
<td>-</td>
<td>Reset</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RAM Retention</td>
<td>-</td>
<td>-</td>
<td>Reset</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>On</td>
<td>FW</td>
</tr>
</tbody>
</table>

Table key:
- FW = Controlled by firmware
- On = Enabled by hardware (Cannot be disabled)
- Off = Disabled by hardware (Cannot be enabled)
- - = No Effect
- R = Restored to previous ACTIVE mode setting when exiting DEEPSLEEP, restored to System Reset state when exiting BACKUP

1: The Always On Domain (AOD) is only reset on power-cycling VDDA and VCOREA.
2: On a System Reset or POR, the 60MHz Osc will automatically be selected as SYS_OSC.
3: A System Reset occurs when returning from BACKUP low-power mode.
4: Peripheral, Soft, and System Resets are initiated by firmware though the GCR_RST0 register. System Reset can also be triggered by the RSTN device pin or Watchdog reset.

### 4.3.1 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_rst = 1`. The reset will be completed immediately upon setting `GCR_RST0.periph_rst = 1`.

### 4.3.2 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 $GCR\_RST0\_.soft\_rst = 1$. The reset will be completed immediately upon setting $GCR\_RST0\_.soft\_rst = 1$.

### 4.3.3 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 $GCR\_RST0\_.sys\_rst = 1$.

### 4.3.4 Power-On Reset

A POR resets everything in the device to its default state.

### 4.4 Cache

Each of the four cache controllers are independently managed. *Figure 4-6 MAX32665—MAX32668 Cache Controllers* Diagram shows the four cache controllers and their memory interfaces. Instruction Cache Controller 0 (ICC0 dedicated to CPU0) and Instruction Cache Controller 1 (ICC1 dedicated to CPU1) and the SPIXF Cache Controller (SFCC) are used for instruction caching only. ICC0 and ICC1 interfaces to the internal 1MB Flash and SFCC interfaces to an external SPI Flash device for external code execution. The SPIXR Data Cache Controller (SRCC) is used for data and instruction caching for external SPI SRAM memories. The SRCC is implemented as a write-through cache.

All four caches are managed separately using their specific cache controller, ICC0, ICC1, SFCC, SRCC. Each controller can be enabled, disabled, and invalidated. Each cache clock can be disabled by placing it in LIGHTSLEEP.
4.5 Instruction Cache Controller

ICC0, ICC1 and SFCC are independent cache controllers and each is controlled directly using their respective register set.

4.5.1 Enabling ICC0/ICC1/SFCC

Perform the following steps to enable ICC0 or ICC1.

1. Set `PWRSEQ_LPMEMSD.icachensd` to 0 to ensure the cache power is on.
2. Set `ICCn_CACHE_CTRL.enable` to 1.
3. Read `ICCn_CACHE_CTRL.ready` until it returns 1.
Perform the following steps to enable SFCC.

1. Set `PWRSEQ_LPMEMSD.icachexipsd` to 0 to ensure the cache power is on.
2. Set `SFCC_CACHE_CTRL.enable` to 1.
3. Read `SFCC_CACHE_CTRL.ready` until it returns 1.

### 4.5.2 Flushing the ICC0/ICC1/SFCC Cache

The System Configuration Register (GCR_SCON) includes a field for flushing these caches simultaneously. Setting `GCR_SCON.ccache_flush` to 1 performs a flush of all three caches. Flush only one of the caches by invalidating the cache contents. Setting the ICCn_INvalidate register to 1 invalidates the respective cache and forces a cache flush. Read the `ICCn_CACHE_CTRL.ready` field until it returns 1 to determine when the flush is completed.

#### 4.5.3 Flushing SRCC Cache

The System Configuration Register (GCR_SCON) includes a field for flushing this caches. Setting `GCR_SCON.dcache_flush` to 1 performs a flush of the cache.

### 4.6 Instruction Cache Controller Registers

See Table 3-1: APB Peripheral Base Address Map for the ICC0, ICC1, and SFCC, Peripheral Base Addresses.

#### Table 4-3: Instruction Cache Controller Register Summary

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>ICCn_CACHE_ID</td>
<td>Cache ID Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>ICCn_MEM_SIZE</td>
<td>Cache Memory Size Register</td>
</tr>
<tr>
<td>[0x0100]</td>
<td>ICCn_CACHE_CTRL</td>
<td>Instruction Cache Control Register</td>
</tr>
<tr>
<td>[0x0700]</td>
<td>ICCn_INVALIDATE</td>
<td>Instruction Cache Controller Invalidate Register</td>
</tr>
</tbody>
</table>

#### Table 4-4: SPIXF Cache Controller Register Summary

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>SFCC_CACHE_ID</td>
<td>Cache ID Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>SFCC_MEM_SIZE</td>
<td>Cache Memory Size Register</td>
</tr>
<tr>
<td>[0x0100]</td>
<td>SFCC_CACHE_CTRL</td>
<td>SPIXF Cache Control Register</td>
</tr>
<tr>
<td>[0x0700]</td>
<td>SFCC_INVALIDATE</td>
<td>SPIXF Cache Controller Invalidate Register</td>
</tr>
</tbody>
</table>

#### Table 4-5: ICCn Cache ID Register

<table>
<thead>
<tr>
<th>ICCn Cache ID</th>
<th>ICCn_CACHE_ID</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:16</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:10</td>
<td><em>cchid</em></td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9:6</td>
<td><em>partnum</em></td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5:0</td>
<td><em>relnum</em></td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 4-6: ICCn Memory Size Register

<table>
<thead>
<tr>
<th>ICCn Memory Size</th>
<th>ICCn_MEM_SIZE</th>
<th>[0x0004]</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
<td>Reset</td>
<td>Description</td>
</tr>
<tr>
<td>31:16</td>
<td>memsz</td>
<td>RO</td>
<td>-</td>
<td>Addressable Memory Size</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indicates the size of addressable memory by this cache controller instance in 128KB units.</td>
</tr>
<tr>
<td>15:0</td>
<td>cchsz</td>
<td>RO</td>
<td>-</td>
<td>Cache Size</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Returns the size of the cache RAM memory in 1KB units.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16: 16KB Cache RAM</td>
</tr>
</tbody>
</table>

### Table 4-7: ICCn Cache Control Register

<table>
<thead>
<tr>
<th>ICCn Cache Control</th>
<th>ICCn_CACHE_CTRL</th>
<th>[0x0100]</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
<td>Reset</td>
<td>Description</td>
</tr>
<tr>
<td>31:17</td>
<td></td>
<td>R/W</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>16</td>
<td>ready</td>
<td>RO</td>
<td>-</td>
<td>Ready</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This field is cleared by hardware anytime the cache as a whole is invalidated (including a Power On Reset event). Hardware automatically sets this field to 1 when the invalidate operation is complete and the cache is ready.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Cache Invalidate in process.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Cache is ready.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: While this field reads 0, the cache is bypassed and reads come directly from the line fill buffer.</td>
</tr>
<tr>
<td>15:1</td>
<td></td>
<td>R/W</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>0</td>
<td>enable</td>
<td>R/W</td>
<td>0</td>
<td>Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set this field to 1 to enable the cache. Setting this field to 0 automatically invalidates the cache contents. When this cache is disabled, reads are handled by the line fill buffer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable cache</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable cache</td>
</tr>
</tbody>
</table>

### Table 4-8: ICCn Invalidate Register

<table>
<thead>
<tr>
<th>ICCn Invalidate</th>
<th>ICCn_INVALIDATE</th>
<th>[0x0700]</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
<td>Reset</td>
<td>Description</td>
</tr>
<tr>
<td>31:0</td>
<td>invalid</td>
<td>WO</td>
<td>-</td>
<td>Invalidate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Any write to this register of any value invalidates the cache.</td>
</tr>
</tbody>
</table>

### Table 4-9: SFCC Cache ID Register

<table>
<thead>
<tr>
<th>SFCC Cache ID</th>
<th>SFCC_CACHE_ID</th>
<th>[0x0000]</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
<td>Reset</td>
<td>Description</td>
</tr>
<tr>
<td>31:16</td>
<td></td>
<td>RO</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>15:10</td>
<td>cchid</td>
<td>RO</td>
<td>-</td>
<td>Cache ID</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Returns the Cache ID for this Cache instance.</td>
</tr>
<tr>
<td>9:6</td>
<td>partnum</td>
<td>RO</td>
<td>-</td>
<td>Cache Part Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Returns the part number indicator for this Cache instance.</td>
</tr>
</tbody>
</table>
### SFCC Cache ID

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 5:0  | renum | RO     | -     | Cache Release Number  
Returns the release number for this Cache instance. |

#### Table 4-10: SFCC Memory Size Register

<table>
<thead>
<tr>
<th>SFCC Memory Size</th>
<th>SFCC_MEM_SIZE</th>
<th>[0x0004]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 31:16            | memsz  | RO     | -     | Addressable Memory Size  
Indicates the size of addressable memory by this cache controller instance in 128KB units. |
| 15:0             | cchsz  | RO     | -     | Cache Size  
Returns the size of the cache RAM memory in 1KB units.  
16: 16KB Cache RAM |

#### Table 4-11: SFCC Cache Control Register

<table>
<thead>
<tr>
<th>SFCC Cache Control</th>
<th>SFCC_CACHE_CTRL</th>
<th>[0x0100]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 31:17              | -     | R/W    | -     | Reserved  
Do not modify this field. |
| 16                 | ready  | RO     | -     | Ready  
This field is cleared by hardware anytime the cache as a whole is invalidated  
(including a Power On Reset event). Hardware automatically sets this field to 1  
when the invalidate operation is complete and the cache is ready.  
0: Cache Invalidate in process.  
1: Cache is ready.  
Note: While this field reads 0, the cache is bypassed and reads come directly from  
the line fill buffer. |
| 15:1               | -     | R/W    | -     | Reserved  
Do not modify this field. |
| 0                  | enable | R/W    | 0     | Enable  
Set this field to 1 to enable the cache. Setting this field to 0 automatically  
invalidates the cache contents. When this cache is disabled, reads are handled by  
the line fill buffer.  
0: Disable cache  
1: Enable cache |

#### Table 4-12: SFCC Invalidate Register

<table>
<thead>
<tr>
<th>SFCC Invalidate</th>
<th>SFCC_INVALIDATE</th>
<th>[0x0700]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 31:0            | -     | WO     | -     | Invalidate  
Any write to this register of any value invalidates the cache. |

### 4.7 External RAM SPIXR Cache Controller (SRCC)

See Section 8.4 SPIXR Cache Controller (SRCC) for detailed usage information for the SRCC and the SRCC register interface.
4.8 RAM Memory Management

This device has many features for managing the on-chip RAM. The on-chip RAM includes data RAM, instruction and data caches, and peripheral FIFOs.

4.8.1 RAM Zeroization

The GCR Memory Zeroize Register, GCR_MEM_ZERO, allows clearing memory for firmware or security reasons. Zeroization writes all zeros to the specified memory.

The following RAM memories can be zeroized:

- The Internal Data RAMs 0 through 6.
  - Each of the internal data RAM segments can be zeroized independently by setting the GCR_MEM_ZERO.sram0z through GCR_MEM_ZERO.sram6z fields to 1.
- The USB FIFO
  - Write 1 to GCR_MEM_ZERO.usbfifoz
- ICC0 16KB Cache
  - Write 1 to GCR_MEM_ZERO.icache0z
- ICC1 16KB Cache
  - Write 1 to GCR_MEM_ZERO.icache1z
- SRCC Cache Tags
  - Write 1 to GCR_MEM_ZERO.scachetagz
  - This clears the Cache tags, ultimately invalidating the SRCC cache memory as well.
- SRCC 16KB Cache Data
  - Write 1 to GCR_MEM_ZERO.scachedataz
- SFCC 16KB Cache
  - Write 1 to GCR_MEM_ZERO.icachexipz
- Crypto MAA RAM (MAX32666/MAX32668)
  - Write 1 to GCR_MEM_ZERO.cryptoz

4.8.2 RAM Low Power Modes

RAM low power modes and shutdown are controlled on a bank basis. The System RAM banks are shown with corresponding bank sizes and base addresses in below:

<table>
<thead>
<tr>
<th>System RAM Block #</th>
<th>Size (Words)</th>
<th>Base Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysram0</td>
<td>8K</td>
<td>2000-0000</td>
</tr>
<tr>
<td>sysram1</td>
<td>8K</td>
<td>2000-8000</td>
</tr>
<tr>
<td>sysram2</td>
<td>16K</td>
<td>2001-0000</td>
</tr>
<tr>
<td>sysram3</td>
<td>16K</td>
<td>2002-0000</td>
</tr>
<tr>
<td>sysram4</td>
<td>32K</td>
<td>2003-0000</td>
</tr>
<tr>
<td>sysram5</td>
<td>32K</td>
<td>2005-0000</td>
</tr>
<tr>
<td>sysram6</td>
<td>2K</td>
<td>2007-0000</td>
</tr>
<tr>
<td>sysram7</td>
<td>2K</td>
<td>2007-2000</td>
</tr>
<tr>
<td>sysram8</td>
<td>4K</td>
<td>2007-4000</td>
</tr>
<tr>
<td>sysram9</td>
<td>4K</td>
<td>2007-8000</td>
</tr>
<tr>
<td>sysram10</td>
<td>8K</td>
<td>2007-C000</td>
</tr>
<tr>
<td>sysram11</td>
<td>8K</td>
<td>2008-4000</td>
</tr>
</tbody>
</table>
4.8.2.1 RAM LIGHTSLEEP

RAM can be placed in a low power mode, referred to as LIGHTSLEEP, using the Memory Clock Control Register, `GCR_MEM_CLK`. LIGHTSLEEP gates off the clock to the RAM and makes the RAM unavailable for read/write operations, while memory contents are retained, reducing power consumption. LIGHTSLEEP is available for the 6 Data RAM blocks and the ECC RAM block, the USB FIFO, Crypto RAM, ICC0 RAM, ICC1 RAM, SFCC RAM and the SRCC RAM. RAM contents are available when exiting LIGHTSLEEP mode.

4.8.2.2 RAM Shut Down

RAM memories can individually be shut down further reducing the power consumption for the device. Shutting down a memory gates off the clock and removes power to the memory. Shutting down a memory invalidates (destroys) the contents of the memory and results in a POR of the memory when it is enabled. RAM memory shut down is configured using the `PWRSEQ_LPMEMSD` register.

4.9 Miscellaneous Control Registers

This set of control registers provides control for system related aspects such as ECC enable, Comparator enable, Square Wave Out enable, Power Down signaling enable, Power Control, Rest Pullup control, and SIMO Clock enable.

See Table 3-1: APB Peripheral Base Address Map for the Miscellaneous Control Peripheral Base Address.

Table 4-14 Miscellaneous Control Register Summary

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>MCR_ECCEN</td>
<td>Error Correction Coding Enable Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>MCR_HIRC96M</td>
<td>96MHz High Frequency Clock Adjustment Register</td>
</tr>
<tr>
<td>[0x0008]</td>
<td>MCR_OUTEN</td>
<td>SQWOUT and PDOWN Enable Register</td>
</tr>
<tr>
<td>[0x000C]</td>
<td>MCR_AINCOMP</td>
<td>Comparator Enable Register</td>
</tr>
<tr>
<td>[0x0010]</td>
<td>MCR_CTRL</td>
<td>Control Register</td>
</tr>
</tbody>
</table>

4.10 Miscellaneous Control Registers Details

Table 4-15: Error Correction Coding (ECC) Enable Register

<table>
<thead>
<tr>
<th>Error Correction Coding Enable</th>
<th>MCR_ECCEN</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:13</td>
<td>FL1eccen</td>
<td>R/O</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>FL0eccen</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>ICSP1XPR</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>IC0eccen</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>IC1eccen</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>IC0eccen</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Error Correction Coding Enable

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:6</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>5</td>
<td>sysram5eccen</td>
<td>R/W</td>
<td>0</td>
<td>Sysram5 ECC Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable</td>
</tr>
<tr>
<td>4</td>
<td>sysram4eccen</td>
<td>R/W</td>
<td>0</td>
<td>Sysram4 ECC Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable</td>
</tr>
<tr>
<td>3</td>
<td>sysram3eccen</td>
<td>R/W</td>
<td>0</td>
<td>Sysram3 ECC Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable</td>
</tr>
<tr>
<td>2</td>
<td>sysram2eccen</td>
<td>R/W</td>
<td>0</td>
<td>Sysram2 ECC Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable</td>
</tr>
<tr>
<td>1</td>
<td>sysram1eccen</td>
<td>R/W</td>
<td>0</td>
<td>Sysram1 ECC Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable</td>
</tr>
<tr>
<td>0</td>
<td>Sysram0eccen</td>
<td>R/W</td>
<td>0</td>
<td>Sysram0 ECC Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable</td>
</tr>
</tbody>
</table>

Table 4-16: SQWOUT and PDOWN Output Enable Register

<table>
<thead>
<tr>
<th>SQWOUT/PDOWN Output Enable</th>
<th>MCR_OUTEN</th>
<th>[0x0008]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------</td>
</tr>
<tr>
<td>31:4</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>pdownout1en</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>pdownout0en</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>sqwout1en</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>sqwout0en</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4-17: Comparator Enable Register

<table>
<thead>
<tr>
<th>Comparator Enable</th>
<th>MCR_AINCOMP</th>
<th>[0x000C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td>31:6</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5:4</td>
<td>aincomphyst</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Comparator Enable

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>aincomp3pd</td>
<td>R/W</td>
<td>1</td>
<td>Comparator 3 Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Comparator is powered and enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Comparator is powered down and disabled</td>
</tr>
<tr>
<td>2</td>
<td>aincomp2pd</td>
<td>R/W</td>
<td>1</td>
<td>Comparator 2 Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Comparator is powered and enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Comparator is powered down and disabled</td>
</tr>
<tr>
<td>1</td>
<td>aincomp1pd</td>
<td>R/W</td>
<td>1</td>
<td>Comparator 1 Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Comparator is powered and enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Comparator is powered down and disabled</td>
</tr>
<tr>
<td>0</td>
<td>aincomp0pd</td>
<td>R/W</td>
<td>1</td>
<td>Comparator 0 Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Comparator is powered and enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Comparator is powered down and disabled</td>
</tr>
</tbody>
</table>

### Table 4-18: Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:11</td>
<td></td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>10</td>
<td>rstn_voltage_sel</td>
<td>R/W</td>
<td>0</td>
<td>RSTN Voltage Supply Select</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit setting determines which supply voltage drives the RSTN device pin.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: (V_{DDIO})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: (V_{DDIOH})</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: This bit is only affected by a Power On Reset event. RSTN assertion has no affect on this bit setting.</td>
</tr>
<tr>
<td>9</td>
<td>p1m</td>
<td>R/W</td>
<td>0</td>
<td>RSTN Device Pin Internal Pullup</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This pin controls the internal pullup value connected to the RSTN device pin.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: 1MOhm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: 25KOhm</td>
</tr>
<tr>
<td>8</td>
<td>buckclkscalen</td>
<td>R/W</td>
<td>0</td>
<td>SIMO Dynamic Clock Scaling Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Allows the dynamic scaling of the SIMO clock as part of the Dynamic Voltage Scaling operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>7:4</td>
<td></td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>3</td>
<td>usbswen_n</td>
<td>R/W</td>
<td>0</td>
<td>USB PHY Power Gate Control</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit is sampled when entering DEEPSLEEP or BACKUP mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: USB Switch On</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: USB Switch Off</td>
</tr>
</tbody>
</table>
Control MCR_CTRL [0x0010]

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2:1</td>
<td>vddcsw</td>
<td>R/W</td>
<td>01</td>
<td><strong>V(_{\text{COREB}}) Switch</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V(<em>{\text{COREB}}) can be operated at a lower voltage to minimize leakage in any of the low power modes SLEEP, DEEPSLEEP, and BACKUP. Allows the CPU cores to operate from V(</em>{\text{COREA}}) during these low power modes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00: Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01: CPU0, CPU1 operate from V(<em>{\text{COREA}}) in low power mode. When V(</em>{\text{COREB}}) reaches operating voltage during exit from low power mode, firmware sets this and the device switches to operate the cores from V(_{\text{COREB}}).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b10: CPU0, CPU1 operate from V(<em>{\text{COREA}}) in low power mode. When V(</em>{\text{COREB}}) is less than V(_{\text{COREA}}), firmware sets this and the device switches back to a low power mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b11: Automatically set by hardware to switch the CPU0, CPU1 cores to V(_{\text{COREB}}).</td>
</tr>
<tr>
<td>0</td>
<td>vddcswen</td>
<td>R/W</td>
<td>0</td>
<td><strong>V(_{\text{COREB}}) Switch Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Allows exit from low power mode according to the setting of MCR_CTRL.vddcsw</td>
</tr>
</tbody>
</table>

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

The Single Inductor Multiple Output (SIMO) switch mode power supply allows the device to operate autonomously from a single lithium cell. The SIMO provides four buck switching regulators (V\(_{\text{REGO}_A}\) thru V\(_{\text{REGO}_D}\)). Each of the four regulator voltages can be controlled by the CPU individually. For the SIMO top operate properly, the four buck regulator outputs must drive the power supply pins of the device as follows in *Table 4-19*.

#### 4.11.1 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:

- V\(_{\text{COREA}}\) (V\(_{\text{COREA}}\)) Digital Core Supply Voltage A for the Always-On Domain
- V\(_{\text{COREB}}\) (V\(_{\text{COREB}}\)) Digital Core Supply Voltage B
- V\(_{\text{DDIO}}\) (V\(_{\text{DDIO}}\)) GPIO Supply Voltage
- V\(_{\text{DDIOH}}\) (V\(_{\text{DDIOH}}\)) GPIO High Supply Voltage
- V\(_{\text{DDA}}\) (V\(_{\text{DDA}}\)) AOD Analog Supply Voltage
- V\(_{\text{REGI}}\) (V\(_{\text{REGI}}\)) Input Supply Voltage, Battery
- V\(_{\text{DDB}}\) (V\(_{\text{DDB}}\)) USB Supply Voltage
- V\(_{\text{TXOUT}}\) (V\(_{\text{TXOUT}}\)) Bluetooth Transmitter Supply Voltage Output
- V\(_{\text{RXOUT}}\) (V\(_{\text{RXOUT}}\)) Bluetooth Receiver Supply Voltage Output

Each of the power supply monitors’ settings are found in the Low Power Control register **PWRSEQ_LPCN**. When the corresponding power monitor is enabled, the input voltage pin is constantly monitored. 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 4-19: SIMO Power Supply Device Pin Connectivity*

<table>
<thead>
<tr>
<th>SIMO Supply Output Pin</th>
<th>Connection</th>
<th>Device Power Supply Input Pin</th>
<th>Supply Monitor Reset Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>V(_{\text{REGO}_A})</td>
<td>→</td>
<td>V(_{\text{DDA}})</td>
<td>Domain reset</td>
</tr>
<tr>
<td>V(_{\text{REGO}_B})</td>
<td>→</td>
<td>V(_{\text{COREB}})</td>
<td>Domain Reset</td>
</tr>
<tr>
<td>V(_{\text{REGO}_C})</td>
<td>→</td>
<td>V(_{\text{COREA}})</td>
<td>Domain Reset</td>
</tr>
<tr>
<td>V(_{\text{REGO}_D})</td>
<td>→</td>
<td>V(<em>{\text{RXIN}}, \text{V}</em>{\text{TXIN}})</td>
<td>-</td>
</tr>
</tbody>
</table>
### 4.12 Single Inductor Multiple Output (SIMO) Registers

See Table 3-1: APB Peripheral Base Address Map for the SIMO Controller Peripheral Base Address.

#### Table 4-20: SIMO Controller Register Summary

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0004]</td>
<td>VREGO_A</td>
<td>Buck Voltage Regulator A Control Register</td>
</tr>
<tr>
<td>[0x0008]</td>
<td>VREGO_B</td>
<td>Buck Voltage Regulator B Control Register</td>
</tr>
<tr>
<td>[0x000C]</td>
<td>VREGO_C</td>
<td>Buck Voltage Regulator C Control Register</td>
</tr>
<tr>
<td>[0x0010]</td>
<td>VREGO_D</td>
<td>Buck Voltage Regulator D Control Register</td>
</tr>
<tr>
<td>[0x0014]</td>
<td>IPKA</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>[0x0018]</td>
<td>IPKB</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>[0x001C]</td>
<td>MAXTON</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>[0x0020]</td>
<td>ILOAD_A</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>[0x0024]</td>
<td>ILOAD_B</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>[0x0028]</td>
<td>ILOAD_C</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>[0x002C]</td>
<td>ILOAD_D</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>[0x0030]</td>
<td>BUCK_ALERT_THR_A</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>[0x0034]</td>
<td>BUCK_ALERT_THR_B</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>[0x0038]</td>
<td>BUCK_ALERT_THR_C</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>[0x003C]</td>
<td>BUCK_ALERT_THR_D</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>[0x0040]</td>
<td>BUCK_OUTREADY</td>
<td>Buck Regulator Output Ready Register</td>
</tr>
<tr>
<td>[0x0044]</td>
<td>ZERO_CROSS_CAL_A</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>[0x0048]</td>
<td>ZERO_CROSS_CAL_B</td>
<td>Reserved. Do not modify this field.</td>
</tr>
</tbody>
</table>
4.13 Single Inductor Multiple Output (SIMO) Registers Details

Table 4-21: Buck Voltage Regulator A Control Register

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x004C]</td>
<td>ZERO_CROSS_CAL_C</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>[0x0050]</td>
<td>ZERO_CROSS_CAL_D</td>
<td>Reserved. Do not modify this field.</td>
</tr>
</tbody>
</table>

**Table 4-21: Buck Voltage Regulator A Control Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td></td>
<td>R/W</td>
<td>-</td>
<td><strong>Reserved</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>7</td>
<td>rangea</td>
<td>R/W</td>
<td>1</td>
<td><strong>Regulator Output Range</strong>&lt;br&gt;0: 0.5V to 1.77&lt;br&gt;1: 0.6V to 1.87V</td>
</tr>
<tr>
<td>6:0</td>
<td>vseta</td>
<td>R/W</td>
<td>0x78h</td>
<td><strong>Regulator Output Voltage</strong>&lt;br&gt;Each increment in the register represents 10mV.&lt;br&gt;rangea = 1: Output Voltage = 0.6V + (10mV × vseta)&lt;br&gt;rangea = 0: Output Voltage = 0.5V + (10mV × vseta)&lt;br&gt;Default: 0x78h = 1.7V when rangea = 0; 1.8V when rangea = 1&lt;br&gt;Warning: When this regulator is connected as shown in <em>SIMO Power Supply Device Pin Connectivity</em>:&lt;br&gt;A: The maximum setting for this regulator must be followed for VDDA as indicated in the device datasheet.&lt;br&gt;B: Setting the regulator to a voltage below the Power-Fail Reset Voltage for VDDA will initiate the Power Monitor Reset Action.</td>
</tr>
</tbody>
</table>

**Table 4-22: Buck Voltage Regulator B Control Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td></td>
<td>R/W</td>
<td>-</td>
<td><strong>Reserved</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>7</td>
<td>rangeb</td>
<td>R/W</td>
<td>1</td>
<td><strong>Regulator Output Range</strong>&lt;br&gt;0: 0.5V to 1.77&lt;br&gt;1: 0.6V to 1.87V</td>
</tr>
<tr>
<td>6:0</td>
<td>vsetb</td>
<td>R/W</td>
<td>0x32h</td>
<td><strong>Regulator Output Voltage</strong>&lt;br&gt;Each increment in the register represents 10mV.&lt;br&gt;rangeb = 1: Output Voltage = 0.6V + (10mV × vsetb)&lt;br&gt;rangeb = 0: Output Voltage = 0.5V + (10mV × vsetb)&lt;br&gt;0x32h: 1.77V when rangeb = 0; 1.87V when rangeb = 1&lt;br&gt;Default: 0x32h = 1.0V when rangeb = 0; 1.1V when rangeb = 1&lt;br&gt;Warning: When this regulator is connected as shown in <em>SIMO Power Supply Device Pin Connectivity</em>:&lt;br&gt;A: The maximum setting for this regulator must be followed for VCOREB as indicated in the device datasheet.&lt;br&gt;B: Setting the regulator to a voltage below the Power-Fail Reset Voltage for VCOREB will initiate the Power Monitor Reset Action.</td>
</tr>
</tbody>
</table>
Table 4-23: Buck Voltage Regulator C Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>-</td>
<td>R/W</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>7</td>
<td>ranged</td>
<td>R/W</td>
<td>1</td>
<td>Regulator Output Range</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The voltage regulator output range setting.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: 0.5V to 1.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: 0.6V to 1.87V</td>
</tr>
<tr>
<td>6:0</td>
<td>vsetc</td>
<td>R/W</td>
<td>0x32h</td>
<td>Regulator Output Voltage</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Each increment in the register represents 10mV.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ranged = 1; Output Voltage = 0.6V + (10mV × vsetc)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ranged = 0; Output Voltage = 0.5V + (10mV × vsetc)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x7Fh: 1.77V when ranged = 0; 1.87V when ranged = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Default: 0x32h = 1.0V when ranged = 0; 1.1V when ranged = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Warning: When this regulator is connected as shown in SIMO Power Supply Device Pin Connectivity:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A: The maximum setting for this regulator must be followed for VCOREA as indicated in the device datasheet.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B: Setting the regulator to a voltage below the Power-Fail Reset Voltage for VCOREA will initiate the Power Monitor Reset Action.</td>
</tr>
</tbody>
</table>

Table 4-24: Buck Voltage Regulator D Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>-</td>
<td>R/W</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>7</td>
<td>ranged</td>
<td>R/W</td>
<td>1</td>
<td>Regulator Output Range</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: 0.5V to 1.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: 0.6V to 1.87V</td>
</tr>
<tr>
<td>6:0</td>
<td>vsetd</td>
<td>R/W</td>
<td>0x32h</td>
<td>Regulator Output Voltage</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Each increment in the register represents 10mV.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ranged = 1; Output Voltage = 0.6V + (10mV × vsetd)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ranged = 0; Output Voltage = 0.5V + (10mV × vsetd)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x7Fh: 1.77V when ranged = 0; 1.87V when ranged = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Default: 0x32h = 1.0V when ranged = 0; 1.1V when ranged = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Warning: When this regulator is connected as shown in SIMO Power Supply Device Pin Connectivity:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A: The maximum setting for this regulator must be followed for VRXIN and VTXIN as indicated in the device datasheet.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B: Setting the regulator to a voltage below the Power-Fail Reset Voltage for VRXIN and/or VTXIN will initiate the Power Monitor Reset Action.</td>
</tr>
</tbody>
</table>

Table 4-25: High Side FET Peak Current VREGO_A VREGO_B Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>-</td>
<td>R/W</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved. Do not modify this field.</td>
</tr>
</tbody>
</table>
### Table 4-26: High Side FET Peak Current VREGO_A VREGO_B Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:4</td>
<td>ipksetb</td>
<td>R/W</td>
<td>0x8h</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>3:0</td>
<td>ipkseta</td>
<td>R/W</td>
<td>0x8h</td>
<td>Reserved. Do not modify this field.</td>
</tr>
</tbody>
</table>

### Table 4-27: Maximum High Side FET Time On Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>-</td>
<td>R/W</td>
<td>-</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>7:4</td>
<td>ipksetd</td>
<td>R/W</td>
<td>0x8h</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>3:0</td>
<td>ipksetc</td>
<td>R/W</td>
<td>0x8h</td>
<td>Reserved. Do not modify this field.</td>
</tr>
</tbody>
</table>

### Table 4-28: Buck Cycle Count VREGO_A Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>7:0</td>
<td>iloada</td>
<td>RO</td>
<td>0</td>
<td>Reserved. Do not modify this field.</td>
</tr>
</tbody>
</table>

### Table 4-29: Buck Cycle Count VREGO_B Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>7:0</td>
<td>iloadb</td>
<td>RO</td>
<td>0</td>
<td>Reserved. Do not modify this field.</td>
</tr>
</tbody>
</table>
### Table 4-30: Buck Cycle Count VREGO_C Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>7:0</td>
<td>iloadc</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved. Do not modify this field.</td>
</tr>
</tbody>
</table>

### Table 4-31: Buck Cycle Count VREGO_D Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>7:0</td>
<td>iloadd</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved. Do not modify this field.</td>
</tr>
</tbody>
</table>

### Table 4-32: Buck Cycle Count Alert VREGO_A Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>7:0</td>
<td>buckthra</td>
<td>R/W</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved. Do not modify this field.</td>
</tr>
</tbody>
</table>

### Table 4-33: Buck Cycle Count Alert VREGO_B Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>7:0</td>
<td>buckthrb</td>
<td>R/W</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved. Do not modify this field.</td>
</tr>
</tbody>
</table>

### Table 4-34: Buck Cycle Count Alert VREGO_C Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>7:0</td>
<td>buckthrc</td>
<td>R/W</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved. Do not modify this field.</td>
</tr>
</tbody>
</table>
### Table 4-35: Buck Cycle Count Alert VREGO_D Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>7:0</td>
<td>buckthrd</td>
<td>R/W</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved. Do not modify this field.</td>
</tr>
</tbody>
</table>

### Table 4-36: Buck Regulator Output Ready Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:4</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>3</td>
<td>buckoutrdya</td>
<td>RO</td>
<td>0</td>
<td>VREGO_A Output Ready</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When VREGO_A.vseta changes, this bit will be set when the output voltage has reached its regulated value. It will not be cleared if the output voltage drops below its set value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Not ready</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Ready</td>
</tr>
<tr>
<td>2</td>
<td>buckoutrdyb</td>
<td>RO</td>
<td>0</td>
<td>VREGO_B Output Ready</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When VREGO_B.vsetb changes, this bit will be set when the output voltage has reached its regulated value. It will not be cleared if the output voltage drops below its set value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Not ready</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Ready</td>
</tr>
<tr>
<td>1</td>
<td>buckoutrdyc</td>
<td>RO</td>
<td>0</td>
<td>VREGO_C Output Ready</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When VREGO_C.vsetc changes, this bit will be set when the output voltage has reached its regulated value. It will not be cleared if the output voltage drops below its set value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Not ready</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Ready</td>
</tr>
<tr>
<td>0</td>
<td>buckoutrdyd</td>
<td>RO</td>
<td>0</td>
<td>VREGO_D Output Ready</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When VREGO_D.vsetd 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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Not ready</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Ready</td>
</tr>
</tbody>
</table>

### Table 4-37: Zero Cross Calibration VREGO_A Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:5</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>4:0</td>
<td>zxcala</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved. Do not modify this field.</td>
</tr>
</tbody>
</table>

---

*MAX32665-MAX32668 User Guide*

Maxim Integrated

Page 77 of 457
### Table 4-38: Zero Cross Calibration VREGO_B Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:5</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>4:0</td>
<td>zxcalb</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved. Do not modify this field.</td>
</tr>
</tbody>
</table>

### Table 4-39: Zero Cross Calibration VREGO_C Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:5</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>4:0</td>
<td>zxcalc</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved. Do not modify this field.</td>
</tr>
</tbody>
</table>

### Table 4-40: Zero Cross Calibration VREGO_D Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:5</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>4:0</td>
<td>zxcald</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved. Do not modify this field.</td>
</tr>
</tbody>
</table>

### 4.14 Power Sequencer and Always-On Domain Registers

See Table 3-1: APB Peripheral Base Address Map for the Power Sequencer Peripheral Base Address.

### Table 4-41: Power Sequencer and Always-On Domain Register Summary

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>PWRSEQ_LPCN</td>
<td>Low Power Control Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>PWRSEQ_LPWKST0</td>
<td>GPIO0 Low Power Wakeup Status Flags</td>
</tr>
<tr>
<td>[0x0008]</td>
<td>PWRSEQ_LPWKEN0</td>
<td>GPIO0 Low Power Wakeup Enable Register</td>
</tr>
<tr>
<td>[0x000C]</td>
<td>PWRSEQ_LPWKST1</td>
<td>GPIO1 Low Power Wakeup Status Flags</td>
</tr>
<tr>
<td>[0x0010]</td>
<td>PWRSEQ_LPWKEN1</td>
<td>GPIO1 Low Power Wakeup Enable Register</td>
</tr>
<tr>
<td>[0x0030]</td>
<td>PWRSEQ_LPPWST</td>
<td>Peripheral Low Power Wakeup Status Flags</td>
</tr>
<tr>
<td>[0x0034]</td>
<td>PWRSEQ_LPPWEN</td>
<td>Peripheral Low Power Wakeup Enable Register</td>
</tr>
<tr>
<td>[0x0040]</td>
<td>PWRSEQ_LPMEMSD</td>
<td>RAM Shutdown Control Register</td>
</tr>
<tr>
<td>[0x0044]</td>
<td>PWRSEQ_LPVDPPD</td>
<td>VDD Low Power Domain Control Register</td>
</tr>
<tr>
<td>[0x0048]</td>
<td>PWRSEQ_BURET</td>
<td>BACKUP Return Vector Register</td>
</tr>
<tr>
<td>[0x004C]</td>
<td>PWRSEQ_BUAOD</td>
<td>BACKUP AoD Register</td>
</tr>
</tbody>
</table>
### 4.15 Power Sequencer and Always-On Domain Register Details

**Table 4-42: Low Power Control Register**

<table>
<thead>
<tr>
<th>Low Power Control</th>
<th>PWRSEQ_LPCN</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31</td>
<td>-</td>
<td>RO</td>
</tr>
</tbody>
</table>
| 30                | pdowndslen  | R/W      | 0     | PDOWN DEEPSLEEP Output Enable  
  0: Disabled  
  1: PDOWN signal is asserted when device enters DEEPSLEEP            |
| 29                | vtxoutmd    | R/W      | 0     | VTXOUT \(V_{TXout}\) Bluetooth Transmitter Supply Power Monitor Disable  
  Reserved. Do not modify this field.                                   |
| 28                | vrxoutmd    | R/W      | 0     | VRXOUT \(V_{RXout}\) Bluetooth Receiver Supply Power Monitor Disable  
  Reserved. Do not modify this field.                                   |
| 27                | vddbmd      | R/W      | 0     | VDDB \(V_{DDB}\) USB Supply Power Monitor Disable  
  Reserved. Do not modify this field.                                   |
| 26                | porvddiohm  | R/W      | 0     | VDDIOH \(V_{DDIOH}\) GPIO Supply Power On Reset Monitor Disable Reserved. Do not modify this field. |
| 25                | porvddiomd  | R/W      | 0     | VDDIO \(V_{DDIO}\) GPIO Supply Power On Reset Monitor Disable Reserved. Do not modify this field. |
| 24                | vddiohmd    | R/W      | 0     | VDDIOH \(V_{DDIOH}\) GPIO Supply Power Fail Monitor Disable Reserved. Do not modify this field. |
| 23                | vddiomd     | R/W      | 0     | VDDIO \(V_{DDIO}\) GPIO Supply Power Fail Monitor Disable Reserved. Do not modify this field. |
| 22                | vddamd      | R/W      | 0     | VDDA \(V_{DDA}\) Analog Supply Power Monitor Disable Reserved. Do not modify this field. |
| 21                | vregimd     | R/W      | 0     | VREGI \(V_{REGI}\) Power Monitor Disable Reserved. Do not modify this field. |
| 20                | vcoremd     | R/W      | 0     | VCOREA and VCOREB Supply Power Monitor Disable Do not modify this field.      |
| 19:12             | -           | RO       | 0     | Reserved Do not modify this field.                                          |
| 11                | bgoff       | R/W      | 1     | DEEPSLEEP and BACKUP Modes Bandgap Off  
  0: System Bandgap is always on  
  1: System Bandgap is off in DEEPSLEEP and BACKUP modes                |
| 10                | fwkm        | RO       | 0     | DEEPSLEEP Mode Fast Wakeup Enable                                           |
| 9                 | bkgrnd      | R/W      | 0     | BACKGROUND Mode Enable Reserved. Do not modify this field.                  |
| 8:2               | -           | RO       | 0     | Reserved Do not modify this field.                                          |
| 1:0               | ramret      | R/W      | 0b00  | BACKUP Mode Data RAM Retention  
  0b00: RAM retention in BACKUP mode disabled  
  0b01: Sysram0  
  0b10: Sysram0 and Sysram1  
  0b11: All System RAM                                                   |
### Table 4-43: GPIO0 Low Power Wakeup Status Flags

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>wakest0</td>
<td>R/W1C</td>
<td>0</td>
<td>GPIO0 Pin Wakeup Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Whenever a GPIO0 pin, in any power mode, transitions from low-to-high or high-to-low, the corresponding bit in this register is set. The device will transition from a low-power to ACTIVE mode if the corresponding interrupt enable bit is set in PWRSEQ_LPWKST0. This register should be cleared before entering any low power mode.</td>
</tr>
</tbody>
</table>

### Table 4-44: GPIO0 Low Power Wakeup Enable Registers

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>wakeen0</td>
<td>R/W</td>
<td>0</td>
<td>GPIO0 Pin Wakeup Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting a bit in this register will cause an interrupt be generated and will wakeup the device from any low power mode to ACTIVE mode if the corresponding bit in the PWRSEQ_LPWKST0 register is set. Bits corresponding to unimplemented GPIO are ignored. Note: To enable the device to wakeup from a low power mode on a GPIO pin transition, first set the “GPIO Wakeup enable” register bit GCR_PMR.gpiowken = 1.</td>
</tr>
</tbody>
</table>

### Table 4-45: GPIO1 Low Power Wakeup Status Flags

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:18</td>
<td></td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>17:0</td>
<td>wakest1</td>
<td>R/W1C</td>
<td>0</td>
<td>GPIO1 Pin Wakeup Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Whenever a GPIO0 pin, in any power mode, transitions from low-to-high or high-to-low, the corresponding bit in this register is set. Bits corresponding to unimplemented GPIO are ignored. The device will transition from a low-power to ACTIVE mode if the corresponding interrupt enable bit is set in PWRSEQ_LPWKEN1. This register should be cleared before entering any low power mode.</td>
</tr>
</tbody>
</table>

### Table 4-46: GPIO1 Low Power Wakeup Enable Registers

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:18</td>
<td></td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
</tbody>
</table>
17:0 Wakeen1 R/W 0 GPIO Pin Wakeup Interrupt Enable
Write 1 to any bit to enable the corresponding pin on the 32-bit GPIO port to generate an interrupt to wakeup the device from any low power mode to ACTIVE mode.
A wakeup occurs on any low-to-high or high-to-low transition on the corresponding pin.
Note: To enable the device to wakeup from a low power mode on a GPIO pin transition, first set the “GPIO Wakeup enable” register bit GCR_PMR.gpiowken = 1.

Table 4-47: Peripheral Low Power Wakeup Status Flags

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:18</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>17</td>
<td>rstwkst</td>
<td>R/W1C</td>
<td>0</td>
<td>Reset Detect Wakeup Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set when an external reset has caused a wakeup.</td>
</tr>
<tr>
<td>16</td>
<td>bbmodest</td>
<td>R/W1C</td>
<td>0</td>
<td>BACKUP Mode Wakeup Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set when the device wakes from BACKUP mode</td>
</tr>
<tr>
<td>15:8</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>11</td>
<td>aincomp3st</td>
<td>RO</td>
<td>0</td>
<td>Analog Input Comparator 3 Output Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The state of this bit reflects output of Analog Input Comparator 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Comparator output is low.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Comparator output is high.</td>
</tr>
<tr>
<td>10</td>
<td>aincomp2st</td>
<td>RO</td>
<td>0</td>
<td>Analog Input Comparator 2 Output Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The state of this bit reflects output of Analog Input Comparator 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Comparator output is low.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Comparator output is high.</td>
</tr>
<tr>
<td>9</td>
<td>aincomp1st</td>
<td>RO</td>
<td>0</td>
<td>Analog Input Comparator 1 Output Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The state of this bit reflects output of Analog Input Comparator 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Comparator output is low.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Comparator output is high.</td>
</tr>
<tr>
<td>8</td>
<td>aincomp0st</td>
<td>RO</td>
<td>0</td>
<td>Analog Input Comparator 0 Output Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The state of this bit reflects output of Analog Input Comparator 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Comparator output is low.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Comparator output is high.</td>
</tr>
<tr>
<td>7</td>
<td>aincomp3wkst</td>
<td>R/W1C</td>
<td>0</td>
<td>Analog Input Comparator 3 Wakeup Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit is set when the comparator inputs detect an event. Write 1 to clear.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: If the corresponding bit PWRSEQ_LPPWEN in register is set, the event generates an interrupt to wakeup the device from a low power mode.</td>
</tr>
<tr>
<td>6</td>
<td>aincomp2wkst</td>
<td>R/W1C</td>
<td>0</td>
<td>Analog Input Comparator 2 Wakeup Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit is set when the comparator inputs detect an event. Write 1 to clear.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: If the corresponding bit in PWRSEQ_LPPWEN register is set, the event generates an interrupt to wakeup the device from a low power mode.</td>
</tr>
</tbody>
</table>
### Peripheral Low Power Wakeup Status Flags

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>aincomp1wkst</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>Analog Input Comparator 1 Wakeup Status Flag</strong>&lt;br&gt;Write 1 to clear.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> If the corresponding bit in PWRSEQ_LPPWEN register is set, the event generates an interrupt to wakeup the device from a low power mode.</td>
</tr>
<tr>
<td>4</td>
<td>aincomp0wkst</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>Analog Input Comparator 0 Wakeup Status Flag</strong>&lt;br&gt;Write 1 to clear.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> If the corresponding bit in PWRSEQ_LPPWEN register is set, the event generates an interrupt to wakeup the device from a low power mode.</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved.</strong>&lt;br&gt;Do not modify this bit from its reset value.</td>
</tr>
<tr>
<td>2</td>
<td>usbvbuswkst</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>USB VBUS State Change Detect Flag</strong>&lt;br&gt;0: Normal operation&lt;br&gt;1: The USB has been powered on or off by plugging or unplugging an external USB Host.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> If the corresponding bit in PWRSEQ_LPPWEN register is set, the event generates an interrupt to wakeup the device from a low power mode.</td>
</tr>
<tr>
<td>1:0</td>
<td>usblswkst</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>USB Line State Change Detect Status Flag</strong>&lt;br&gt;If one or both USB differential pair D+/D- pins change state, one or both of this field’s bits are correspondingly set.&lt;br&gt;usblswkst[0] corresponds to D+&lt;br&gt;usblswkst[1] corresponds to D-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> If the corresponding bit in PWRSEQ_LPPWEN register is set, the event generates an interrupt to wakeup the device from a low power mode.</td>
</tr>
</tbody>
</table>

### Table 4-48: Peripheral Low Power Wakeup Enable Register

<table>
<thead>
<tr>
<th>Peripheral Low Power Wakeup Enable</th>
<th>PWRSEQ_LPPWEN</th>
<th>[0x0034]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>--------</td>
</tr>
<tr>
<td>31:8</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td>7</td>
<td>aincomp3wken</td>
<td>R/W</td>
</tr>
<tr>
<td>6</td>
<td>aincomp2wken</td>
<td>R/W</td>
</tr>
<tr>
<td>5</td>
<td>aincomp1wken</td>
<td>RO</td>
</tr>
<tr>
<td>4</td>
<td>aincomp0wken</td>
<td>R/W</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>R/W</td>
</tr>
</tbody>
</table>
### Peripheral Low Power Wakeup Enable

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>usbvbuswken</td>
<td>R/W</td>
<td>0</td>
<td><strong>USB VBUS State Change Wakeup Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to enable an interrupt and wakeup the device from any low power</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mode when ( PWRSEQ_LPPWST.\text{usbvbuswken} = 1 ).</td>
</tr>
<tr>
<td>1:0</td>
<td>usblswkst</td>
<td>R/W</td>
<td>0</td>
<td><strong>USB Line State Change Wakeup Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 0b11 to enable an interrupt and wakeup the device from any low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>power mode when ( PWRSEQ_LPPWST.\text{usblswkst} \neq 0 ).</td>
</tr>
</tbody>
</table>

#### Table 4-49: RAM Shutdown Control Register

<table>
<thead>
<tr>
<th>RAM Shutdown Control</th>
<th>PWRSEQ_LPMEMSD</th>
<th>[0x0040]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:15</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td>14</td>
<td>icache1sd</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Rom1sd</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>rom0sd</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>usbfifosd</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>cryptosd</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>scachesd</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>icachexipsd</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>icache0sd</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### RAM Shutdown Control

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td><strong>Reserved</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>5</td>
<td>sram5sd</td>
<td>R/W</td>
<td>0</td>
<td>Sysram5 and Sysram11 Shut Down&lt;br&gt;0: Power enabled.&lt;br&gt;1: Power shut down. Affected memory is destroyed.&lt;br&gt;See Table 4-13 RAM Block Size and Base Address for base address and size information. Note: See GCR_MEM_CLK register for retention mode power settings.</td>
</tr>
<tr>
<td>4</td>
<td>sram4sd</td>
<td>R/W</td>
<td>0</td>
<td>Sysram4 and Sysram10 Shut Down&lt;br&gt;0: Power enabled.&lt;br&gt;1: Power shut down. Affected memory is destroyed.&lt;br&gt;See Table 4-13 RAM Block Size and Base Address for base address and size information. Note: See GCR_MEM_CLK register for retention mode power settings.</td>
</tr>
<tr>
<td>3</td>
<td>sram3sd</td>
<td>R/W</td>
<td>0</td>
<td>Sysram3 and Sysram9 Shut Down&lt;br&gt;0: Power enabled.&lt;br&gt;1: Power shut down. Affected memory is destroyed.&lt;br&gt;See Table 4-13 RAM Block Size and Base Address for base address and size information. Note: See GCR_MEM_CLK register for retention mode power settings.</td>
</tr>
<tr>
<td>2</td>
<td>sram2sd</td>
<td>R/W</td>
<td>0</td>
<td>Sysram2 and Sysram8 Shut Down&lt;br&gt;0: Power enabled.&lt;br&gt;1: Power shut down. Affected memory is destroyed.&lt;br&gt;See Table 4-13 RAM Block Size and Base Address for base address and size information. Note: See GCR_MEM_CLK register for retention mode power settings.</td>
</tr>
<tr>
<td>1</td>
<td>sram1sd</td>
<td>R/W</td>
<td>0</td>
<td>Sysram1 and Sysram7 Shut Down&lt;br&gt;0: Power enabled.&lt;br&gt;1: Power shut down. Affected memory is destroyed.&lt;br&gt;See Table 4-13 RAM Block Size and Base Address for base address and size information. Note: See GCR_MEM_CLK register for retention mode power settings.</td>
</tr>
<tr>
<td>0</td>
<td>sram0sd</td>
<td>R/W</td>
<td>0</td>
<td>Sysram0 and Sysram6 Shut Down&lt;br&gt;0: Power enabled.&lt;br&gt;1: Power shut down. Affected memory is destroyed.&lt;br&gt;See Table 4-13 RAM Block Size and Base Address for base address and size information. Note: See GCR_MEM_CLK register for retention mode power settings.</td>
</tr>
</tbody>
</table>

Table 4-50: Low Power VDD Power Down Register

<table>
<thead>
<tr>
<th>Low Power VDD Power Down</th>
<th>PWRSEQ_LPVDDPD</th>
<th>[0x0044]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:12</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td>11</td>
<td>vdd5pd</td>
<td>R/W</td>
</tr>
</tbody>
</table>
### Low Power VDD Power Down

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>vdd4pd</td>
<td>R/W</td>
<td>0</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>9</td>
<td>vdd3pd</td>
<td>R/W</td>
<td>0</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>8</td>
<td>vdd2pd</td>
<td>R/W</td>
<td>0</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>7:2</td>
<td></td>
<td>R/W</td>
<td>-</td>
<td>Reserved. Do not modify this field.</td>
</tr>
<tr>
<td>1</td>
<td>vregodpd</td>
<td>R/W</td>
<td>0</td>
<td>VREGO_D Regulator ACTIVE Mode Power Down</td>
</tr>
<tr>
<td>0</td>
<td>vregobpd</td>
<td>R/W</td>
<td>0</td>
<td>VREGO_B Regulator BACKUP Mode Power Down</td>
</tr>
</tbody>
</table>

#### Table 4-51: BACKUP Return Vector Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>gpr0</td>
<td>R/W</td>
<td>0</td>
<td>BACKUP Return Vector</td>
</tr>
</tbody>
</table>

#### Table 4-52: BACKUP AoD Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>gpr1</td>
<td>R/W</td>
<td>0</td>
<td>General Purpose Register 0</td>
</tr>
</tbody>
</table>

### 4.16 Global Control Registers (GCR)

See Table 3-1: APB Peripheral Base Address Map for the General Control Register’s Peripheral Address.

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 4-53: Global Control Register Summary

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>GCR_SCON</td>
<td>System Control Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>GCR_RST0</td>
<td>Reset Register 0</td>
</tr>
<tr>
<td>[0x0008]</td>
<td>GCR_CLK_CTRL</td>
<td>Clock Control Register</td>
</tr>
<tr>
<td>[0x000C]</td>
<td>GCR_PMR</td>
<td>Power Management Register</td>
</tr>
<tr>
<td>[0x0018]</td>
<td>GCR_PCLK_DIV</td>
<td>Peripheral Clocks Divisor</td>
</tr>
<tr>
<td>[0x0024]</td>
<td>GCR_PCLK_DIS0</td>
<td>Peripheral Clocks Disable 0</td>
</tr>
<tr>
<td>[0x0028]</td>
<td>GCR_MEM_CLK</td>
<td>Memory Clock Control</td>
</tr>
<tr>
<td>[0x002C]</td>
<td>GCR_MEM_ZERO</td>
<td>Memory Zeroize Register</td>
</tr>
</tbody>
</table>
### 4.17 Global Control Register Details (GCR)

#### Table 4-54: System Control Register

<table>
<thead>
<tr>
<th>System Control</th>
<th>GCR_SCON</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bits</strong></td>
<td><strong>Field</strong></td>
<td><strong>Access</strong></td>
</tr>
<tr>
<td>31:18</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17:16</td>
<td>ovr</td>
<td>R/W</td>
</tr>
</tbody>
</table>
|                |          |          |          | To allow on-chip volatile memory to operate at the optimal timing range, set this to be the same as V\textsubscript{COREB}.
|                |          |          |          |  
|                |          |          |          | 0b00: 0.9V ±10%                           |
|                |          |          |          | 0b01: 1.0V ±10%                           |
|                |          |          |          | 0b10: 1.1V ±10%                           |
|                |          |          |          | 0b11: Reserved.                           |
| 15             | chkres   | RO       | 0        | ROM Checksum Calculation Pass/Fail        |
|                |          |          |          | This is the result after setting bit GCR\textsubscript{S}CON.chk.|
|                |          |          |          | This bit is only valid after the ROM checksum is complete and chk is cleared. |
|                |          |          |          | 0: Pass                                   |
|                |          |          |          | 1: Fail                                   |
| 14             | -        | RO       | 0        | Reserved                                   |
|                |          |          |          | Do not modify this field.                  |
| 13             | cchk     | R/W      | 0        | Calculate ROM Checksum                    |
|                |          |          |          | This bit is self-clearing when the ROM checksum calculation is complete, and the result is available at bit GCR\textsubscript{S}CON.chkres. Writing a 0 has no effect. |
|                |          |          |          | 0: No operation                           |
|                |          |          |          | 1: Start ROM checksum calculation.        |
| 12:10          | -        | RO       | 0        | Reserved                                   |
|                |          |          |          | Do not modify this field.                  |
### System Control

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>dcache_dis</td>
<td>R/W</td>
<td>0</td>
<td><strong>SPIXR Cache Controller (SRCC) Disable</strong>&lt;br&gt;This disables the SRCC used for SPIXR code and data cache. Setting this field disables the cache and bypasses the cache line buffer.&lt;br&gt;0: Cache enabled&lt;br&gt;1: Cache disabled and line buffer bypassed</td>
</tr>
<tr>
<td>8</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>7</td>
<td>dcache_flush</td>
<td>R/W</td>
<td>0</td>
<td><strong>SPIXR Cache (SRCC) Flush</strong>&lt;br&gt;Write 1 to flush the SPIXR 16KB cache. This bit is automatically cleared to 0 when the flush is complete. Writing 0 has no effect.&lt;br&gt;0: Memory flush not in progress.&lt;br&gt;1: Memory flush in progress.</td>
</tr>
<tr>
<td>6</td>
<td>ccache_flush</td>
<td>R/W</td>
<td>0</td>
<td><strong>ICCC/ICC1/SFCC Code Cache Flush</strong>&lt;br&gt;Write 1 to flush all three caches. This bit is automatically cleared to 0 when the flush is complete. Writing 0 has no effect.&lt;br&gt;0: Memory flush not in progress.&lt;br&gt;1: Memory flush in progress.</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>4</td>
<td>flash_page_flip</td>
<td>R/*</td>
<td>0</td>
<td><strong>Flash Page Flip Flag</strong>&lt;br&gt;Flips the bottom and top halves of Flash memory. This bit is controlled by hardware. Firmware should not change the state of this bit during normal operation. Any change to this bit also flushes both code and data caches.&lt;br&gt;0: Physical layout matches logical layout&lt;br&gt;1: Top and Bottom halves flipped</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>2:1</td>
<td>sbusarb</td>
<td>R/W</td>
<td>1</td>
<td><strong>System Bus Arbitration Scheme</strong>&lt;br&gt;00: Fixed Burst&lt;br&gt;01: Round-Robin&lt;br&gt;10: Reserved&lt;br&gt;11: Reserved</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td><strong>Boundary Scan Tap Enable</strong>&lt;br&gt;Reserved&lt;br&gt;Do not modify this field.</td>
</tr>
</tbody>
</table>

### Table 4-55: Reset Register 0

<table>
<thead>
<tr>
<th>Reset 0</th>
<th>GCR_RST0</th>
<th>[0x0004]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31</td>
<td>sys_rst</td>
<td>R/W</td>
</tr>
<tr>
<td>Reset 0</td>
<td>GCR_RST0</td>
<td>[0x0004]</td>
</tr>
<tr>
<td>--------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 30     | periph_rst | R/W      | 0     | Peripheral Reset
|        |          |          |       | Write 1 to reset.                                                            |
|        |          |          |       | 0: Not in reset                                                            |
|        |          |          |       | 1: Reset in progress.                                                      |
|        |          |          |       | Note: Watchdog Timers, GPIO Ports, the AoD, RAM Retention and the General Control Registers (GCR) are unaffected. |
| 29     | soft_rst | R/W      | 0     | Soft Reset
|        |          |          |       | Write 1 to reset.                                                           |
|        |          |          |       | 0: Not in reset                                                            |
|        |          |          |       | 1: Reset in progress.                                                      |
| 28     | uart2    | R/W      | 0     | UART2 Reset
|        |          |          |       | Write 1 to reset.                                                           |
|        |          |          |       | 0: Not in reset                                                            |
|        |          |          |       | 1: Reset in progress.                                                      |
| 27     | dma1     | R/W      | 0     | DMA1 Reset
|        |          |          |       | Write 1 to reset.                                                           |
|        |          |          |       | 0: Not in reset                                                            |
|        |          |          |       | 1: Reset in progress.                                                      |
| 26     | adc      | R/W      | 0     | ADC Reset
|        |          |          |       | Write 1 to reset.                                                           |
|        |          |          |       | 0: Not in reset                                                            |
|        |          |          |       | 1: Reset in progress.                                                      |
| 25:24  | -        | R/W      | 0     | Reserved
|        |          |          |       | Do not modify this field.                                                  |
| 23     | usb      | R/W      | 0     | USB Reset
|        |          |          |       | Write 1 to reset.                                                           |
|        |          |          |       | 0: Not in reset                                                            |
|        |          |          |       | 1: Reset in progress.                                                      |
| 22     | smphr    | R/W      | 0     | Semaphore Block Reset
|        |          |          |       | Write 1 to reset.                                                           |
|        |          |          |       | 0: Not in reset                                                            |
|        |          |          |       | 1: Reset in progress.                                                      |
| 21:19  | -        | R/W      | -     | Reserved
|        |          |          |       | Do not modify this field.                                                  |
| 18     | crypto   | R/W      | 0     | Cryptographic Reset
|        |          |          |       | Write 1 to reset. This resets the AES block, SHA block, and DES block.    |
|        |          |          |       | 0: Not in reset                                                            |
|        |          |          |       | 1: Reset in progress.                                                      |
| 17     | rtc      | R/W      | 0     | RTC Reset
|        |          |          |       | Write 1 to reset.                                                           |
|        |          |          |       | 0: Not in reset                                                            |
|        |          |          |       | 1: Reset in progress.                                                      |
| 16     | i2c0     | R/W      | 0     | I2C0 Reset
<p>|        |          |          |       | Write 1 to reset.                                                           |
|        |          |          |       | 0: Not in reset                                                            |
|        |          |          |       | 1: Reset in progress.                                                      |</p>
<table>
<thead>
<tr>
<th>Reset 0</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>spi2</td>
<td>R/W</td>
<td>0</td>
<td>SPI2 Reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to reset.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Not in reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Reset in progress.</td>
</tr>
<tr>
<td>14</td>
<td>spi1</td>
<td>R/W</td>
<td>0</td>
<td>SPI1 Reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to reset.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Not in reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Reset in progress.</td>
</tr>
<tr>
<td>13</td>
<td>spi0</td>
<td>R/W</td>
<td>0</td>
<td>SPI0 Reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to reset.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Not in reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Reset in progress.</td>
</tr>
<tr>
<td>12</td>
<td>uart1</td>
<td>R/W</td>
<td>0</td>
<td>UART1 Reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to reset.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Not in reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Reset in progress.</td>
</tr>
<tr>
<td>11</td>
<td>uart0</td>
<td>R/W</td>
<td>0</td>
<td>UART0 Reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to reset.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Not in reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Reset in progress.</td>
</tr>
<tr>
<td>10</td>
<td>timer5</td>
<td>R/W</td>
<td>0</td>
<td>TMR5 Reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to reset.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Not in reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Reset in progress.</td>
</tr>
<tr>
<td>9</td>
<td>timer4</td>
<td>R/W</td>
<td>0</td>
<td>TMR4 Reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to reset the peripheral.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Reset in progress.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Not in reset</td>
</tr>
<tr>
<td>8</td>
<td>timer3</td>
<td>R/W</td>
<td>0</td>
<td>TMR3 Reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to reset.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Not in reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Reset in progress.</td>
</tr>
<tr>
<td>7</td>
<td>timer2</td>
<td>R/W</td>
<td>0</td>
<td>TMR2 Reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to reset.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Not in reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Reset in progress.</td>
</tr>
<tr>
<td>6</td>
<td>timer1</td>
<td>R/W</td>
<td>0</td>
<td>TMR1 Reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to reset.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Not in reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Reset in progress.</td>
</tr>
<tr>
<td>5</td>
<td>timer0</td>
<td>R/W</td>
<td>0</td>
<td>TMR0 Reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to reset.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Not in reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Reset in progress.</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
</tbody>
</table>
### Table 4-56: System Clock Control Register

<table>
<thead>
<tr>
<th>System Clock Control</th>
<th>GCR_CLK_CTRL</th>
<th>[0x0008]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bits</strong></td>
<td><strong>Field</strong></td>
<td><strong>Access</strong></td>
</tr>
<tr>
<td>31:29</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td>28</td>
<td>hirc7m_rdy</td>
<td>RO</td>
</tr>
<tr>
<td>27</td>
<td>hircmm_rdy</td>
<td>RO</td>
</tr>
<tr>
<td>26</td>
<td>hirc60m_rdy</td>
<td>RO</td>
</tr>
<tr>
<td>25</td>
<td>x32k_rdy</td>
<td>RO</td>
</tr>
<tr>
<td>24</td>
<td>x32M_rdy</td>
<td>RO</td>
</tr>
<tr>
<td>23:22</td>
<td>-</td>
<td>RO</td>
</tr>
</tbody>
</table>
## System Clock Control

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 21   | hirc7m_vs   | R/W    | 0     | **7.3728MHz Internal Oscillator Voltage Source Select**<br>7.3728MHz oscillator voltage is sourced by pin VDDA. When exiting DEEPSLEEP the voltage is automatically switched back to this bit setting.  
0: VDDA  
1: Internal 1V regulator sourced from pin VDDA |
| 20   | hirc7m_en   | R/W    | 0     | **7.3728MHz Internal Oscillator Enable**  
0: Disabled  
1: Enabled and ready when GCR_CLK_CTRL.hirc7m_rdy = 1. |
| 19   | hrcmm_en    | R/W    | 0     | **96MHz Internal Oscillator Enable**  
0: Disabled  
1: Enabled and ready when GCR_CLK_CTRL.hircmm_rdy = 1. |
| 18   | hirc60m_en  | R/W    | 1     | **60MHz Internal Oscillator Enable**  
0: Disabled  
1: Enabled and ready when GCR_CLK_CTRL.hirc60m_rdy = 1. |
| 17   | x32k_en     | R/W    | 0     | **32.768kHz External Oscillator Enable**  
0: Disabled  
1: Enabled and ready when GCR_CLK_CTRL.x32k_rdy = 1. |
| 16   | x32M_en     | R/W    | 0     | **32MHz Bluetooth Oscillator Enable**  
0: Disabled  
1: Enabled and ready when GCR_CLK_CTRL.x32M_rdy = 1. |
| 15   | ccd         | R/W    | 0     | **Crypto Accelerator Clock Divider Status**  
0: Crypto clock divide by 1  
1: Crypto clock is divide by 2 |
| 14   | -           | RO     | 0     | **Reserved**  
Do not modify this field. |
| 13   | sysosc_rdy  | RO     | 0     | **SYS_OSC Select Ready**  
When SYS_OSC is changed by modifying sysosc_sel, there is a delay until the switchover is complete. This bit is cleared until the switchover is complete.  
0: Switch to new clock source not yet complete  
1: SYS_OSC is clock source selected in sysosc_sel |
| 12   | -           | RO     | -     | **Reserved**  
Do not modify this field. |
| 11:9 | sysosc_sel  | R/W    | 0     | **System Oscillator Source Select**<br>Selects the system oscillator (SYS_OSC) source used to generate the system clock (SYS_CLK). Modifying this field immediately clears GCR_CLK_CTRL.sysosc_rdy.<br>  
0: 60MHz LP Internal Oscillator  
1: Reserved  
2: 32MHz Bluetooth Oscillator  
3: 8kHz Internal Oscillator  
4: 96MHz Internal Oscillator  
5: 7.3728MHz Internal Oscillator  
6: 32.768kHz External Oscillator  
7: Reserved |
| 8:6  | sysclk_prescale | R/W | 0 | **System Oscillator Prescaler**<br>Sets the divider for generating SYS_CLK from the selected SYS_OSC as shown in the following equation:<br>\[
SYS_{CLK} = \frac{SYS_{OSC}}{2^{sysclk\_prescale}}
\]
## System Clock Control

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5:0</td>
<td>-</td>
<td>R/W</td>
<td>0b001000</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
</tbody>
</table>

### Table 4-57: Power Management Register

<table>
<thead>
<tr>
<th>Power Management</th>
<th>GCR_PMR</th>
<th>0x000C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:21</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>xtalpb</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19:18</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17:15</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14:10</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>compwken</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>wutwken</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>usbwken</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>rtcwken</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Maxim Integrated
### Power Management

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>gpiowken</td>
<td>R/W</td>
<td>0</td>
<td>GPIO Wakeup Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Activity on any GPIO pin configured for wakeup causes an exit from SLEEP,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DEEPSLEEP, and BACKUP modes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>2:0</td>
<td>mode</td>
<td>R/W</td>
<td>0</td>
<td>Operating Mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b000: ACTIVE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b010: DEEPSLEEP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b011: Reserved.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b100: BACKUP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: All other values are Reserved.</td>
</tr>
</tbody>
</table>

### Table 4-58: Peripheral Clock Divisor Register

<table>
<thead>
<tr>
<th>Peripheral Clocks Divisor</th>
<th>GCR_PCLK_DIV</th>
<th>[0x0018]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:16</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:14</td>
<td>aondiv</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13:10</td>
<td>adcfreq</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9:8</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>sdhcfrq</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6:0</td>
<td>-</td>
<td>RO</td>
</tr>
</tbody>
</table>

Maxim Integrated
### Table 4-59: Peripheral Clock Disable Register 0

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>spixpm</td>
<td>R/W</td>
<td>1</td>
<td>SPIXF Master Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
<tr>
<td>30</td>
<td>spixipf</td>
<td>R/W</td>
<td>1</td>
<td>SPIXF Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
<tr>
<td>29</td>
<td>pt</td>
<td>R/W</td>
<td>1</td>
<td>Pulse Train Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
<tr>
<td>28</td>
<td>i2c1</td>
<td>R/W</td>
<td>1</td>
<td>I2C1 Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
<tr>
<td>27:24</td>
<td>-</td>
<td>R/W</td>
<td>0b1111</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>23</td>
<td>adc</td>
<td>R/W</td>
<td>1</td>
<td>ADC Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
<tr>
<td>22:21</td>
<td>-</td>
<td>R/W</td>
<td>0b11</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>20</td>
<td>timer5</td>
<td>R/W</td>
<td>1</td>
<td>TMR5 Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
<tr>
<td>19</td>
<td>timer4</td>
<td>R/W</td>
<td>1</td>
<td>TMR4 Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
<tr>
<td>18</td>
<td>timer3</td>
<td>R/W</td>
<td>1</td>
<td>TMR3 Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
<td>Reset</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>--------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>17</td>
<td>timer2</td>
<td>R/W</td>
<td>1</td>
<td><strong>TMR2 Clock Disable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reads and writes to peripheral registers are disabled. Peripheral register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
<tr>
<td>16</td>
<td>timer1</td>
<td>R/W</td>
<td>1</td>
<td><strong>TMR1 Clock Disable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reads and writes to peripheral registers are disabled. Peripheral register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
<tr>
<td>15</td>
<td>timer0</td>
<td>R/W</td>
<td>1</td>
<td><strong>TMR0 Clock Disable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reads and writes to peripheral registers are disabled. Peripheral register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
<tr>
<td>14</td>
<td>crypto</td>
<td>R/W</td>
<td>1</td>
<td><strong>Crypto Clock Disable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reads and writes to peripheral registers are disabled. Peripheral register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
<tr>
<td>13</td>
<td>i2c0</td>
<td>R/W</td>
<td>1</td>
<td><strong>I2C0 Clock Disable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reads and writes to peripheral registers are disabled. Peripheral register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
<tr>
<td>12:11</td>
<td></td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved</strong></td>
</tr>
<tr>
<td>10</td>
<td>uart1</td>
<td>R/W</td>
<td>1</td>
<td><strong>UART1 Clock Disable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reads and writes to peripheral registers are disabled. Peripheral register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
<tr>
<td>9</td>
<td>uart0</td>
<td>R/W</td>
<td>1</td>
<td><strong>UART0 Clock Disable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to disable the clock to the corresponding peripheral.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock peripheral disables functionality while also saving</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>power. Reads and writes to peripheral registers are disabled. Peripheral</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>register states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled to peripheral.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled to peripheral.</td>
</tr>
<tr>
<td>8</td>
<td>spi2</td>
<td>R/W</td>
<td>1</td>
<td><strong>SPI2 Clock Disable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reads and writes to peripheral registers are disabled. Peripheral register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
<tr>
<td>7</td>
<td>spi1</td>
<td>R/W</td>
<td>1</td>
<td><strong>SPI1 Clock Disable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reads and writes to peripheral registers are disabled. Peripheral register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>spi0</td>
<td>R/W</td>
<td>1</td>
<td>SPI0 Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
<tr>
<td>5</td>
<td>dma0</td>
<td>R/W</td>
<td>1</td>
<td>DMA0 Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>3</td>
<td>usb</td>
<td>R/W</td>
<td>1</td>
<td>USB Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>1</td>
<td>gpio1</td>
<td>R/W</td>
<td>1</td>
<td>GPIO1 Port and Pad Logic Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
<tr>
<td>0</td>
<td>gpio0</td>
<td>R/W</td>
<td>1</td>
<td>GPIO0 Port and Pad Logic Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling a clock disables functionality while also saving power. Reads and writes to peripheral registers are disabled. Peripheral register states are retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Clock enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Clock disabled.</td>
</tr>
</tbody>
</table>

### Memory Clock Control

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>icache1ls</td>
<td>R/W</td>
<td>0</td>
<td>Internal Flash ICC1 LIGHTSLEEP Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Data is unavailable for read/write operations in LIGHTSLEEP mode but is retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: ACTIVE mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: LIGHTSLEEP mode enabled.</td>
</tr>
<tr>
<td>30</td>
<td>rom1ls</td>
<td>R/W</td>
<td>0</td>
<td>ROM1 LIGHTSLEEP Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Data is unavailable for read/write operations in LIGHTSLEEP mode but is retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: ACTIVE mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: LIGHTSLEEP mode enabled.</td>
</tr>
<tr>
<td>29</td>
<td>rom0ls</td>
<td>R/W</td>
<td>0</td>
<td>ROM0 LIGHTSLEEP Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Data is unavailable for read/write operations in LIGHTSLEEP mode but is retained.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: ACTIVE mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: LIGHTSLEEP mode enabled.</td>
</tr>
</tbody>
</table>
## Memory Clock Control

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>usbls</td>
<td>R/W</td>
<td>0</td>
<td><strong>USB FIFO LIGHTSLEEP Enable</strong>&lt;br&gt;Data is unavailable for read/write operations in LIGHTSLEEP mode but is retained.&lt;br&gt;0: ACTIVE mode.&lt;br&gt;1: LIGHTSLEEP mode enabled.</td>
</tr>
<tr>
<td>27</td>
<td>cryptols</td>
<td>R/W</td>
<td>0</td>
<td><strong>Crypto RAM LIGHTSLEEP Enable</strong>&lt;br&gt;Data is unavailable for read/write operations in LIGHTSLEEP mode but is retained.&lt;br&gt;0: ACTIVE mode.&lt;br&gt;1: LIGHTSLEEP mode enabled.</td>
</tr>
<tr>
<td>26</td>
<td>scachels</td>
<td>R/W</td>
<td>0</td>
<td><strong>SRCC Cache LIGHTSLEEP Enable</strong>&lt;br&gt;Data is unavailable for read/write operations in LIGHTSLEEP mode but is retained.&lt;br&gt;0: ACTIVE mode.&lt;br&gt;1: LIGHTSLEEP mode enabled.</td>
</tr>
<tr>
<td>25</td>
<td>icachexipls</td>
<td>R/W</td>
<td>0</td>
<td><strong>SFCC Cache RAM LIGHTSLEEP Enable</strong>&lt;br&gt;Data is unavailable for read/write operations in LIGHTSLEEP mode but is retained.&lt;br&gt;0: ACTIVE mode.&lt;br&gt;1: LIGHTSLEEP mode enabled.</td>
</tr>
<tr>
<td>24</td>
<td>icache0ls</td>
<td>R/W</td>
<td>0</td>
<td><strong>Internal Flash ICC0 LIGHTSLEEP Enable</strong>&lt;br&gt;Data is unavailable for read/write operations in LIGHTSLEEP mode but is retained.&lt;br&gt;0: ACTIVE mode.&lt;br&gt;1: LIGHTSLEEP mode enabled.</td>
</tr>
<tr>
<td>23</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td><strong>Reserved</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>22</td>
<td>sysram6ls</td>
<td>R/W</td>
<td>0</td>
<td><strong>Sysram6 to Sysram11 LIGHTSLEEP Enable</strong>&lt;br&gt;Data is unavailable for read/write operations in LIGHTSLEEP mode but is retained.&lt;br&gt;See Table 4-13 RAM for base address and size information.&lt;br&gt;0: ACTIVE mode.&lt;br&gt;1: LIGHTSLEEP mode enabled.&lt;br&gt;Note: To put RAM in a shutdown mode that removes all power from the RAM and reset the RAM contents, use the PWRSEQ_LPMEMSD register.</td>
</tr>
<tr>
<td>21</td>
<td>sysram5ls</td>
<td>R/W</td>
<td>0</td>
<td><strong>Sysram5 LIGHTSLEEP Enable</strong>&lt;br&gt;Data is unavailable for read/write operations in LIGHTSLEEP mode but is retained.&lt;br&gt;See Table 4-13 RAM for base address and size information.&lt;br&gt;0: ACTIVE mode.&lt;br&gt;1: LIGHTSLEEP mode enabled.&lt;br&gt;Note: To put RAM in a shutdown mode that removes all power from the RAM and reset the RAM contents, use the PWRSEQ_LPMEMSD register.</td>
</tr>
<tr>
<td>20</td>
<td>sysram4ls</td>
<td>R/W</td>
<td>0</td>
<td><strong>Sysram4 LIGHTSLEEP Enable</strong>&lt;br&gt;Data is unavailable for read/write operations in LIGHTSLEEP mode but is retained.&lt;br&gt;See Table 4-13 RAM for base address and size information.&lt;br&gt;0: ACTIVE mode.&lt;br&gt;1: LIGHTSLEEP mode enabled.&lt;br&gt;Note: To put RAM in a shutdown mode that removes all power from the RAM and reset the RAM contents, use the PWRSEQ_LPMEMSD register.</td>
</tr>
</tbody>
</table>
### Memory Clock Control

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 19   | sysram3ls | R/W    | 0     | **Sysram3 LIGHTSLEEP Enable** Data is unavailable for read/write operations in LIGHTSLEEP mode but is retained. See Table 4-13 RAM for base address and size information.  
0: ACTIVE mode.  
1: LIGHTSLEEP mode enabled.  
*Note: To put RAM in a shutdown mode that removes all power from the RAM and reset the RAM contents, use the PWRSEQ_LPMEMSD register.* |
| 18   | sysram2ls | R/W    | 0     | **Sysram2 LIGHTSLEEP Enable** Data is unavailable for read/write operations in LIGHTSLEEP mode but is retained. See Table 4-13 RAM for base address and size information.  
0: ACTIVE mode.  
1: LIGHTSLEEP mode enabled.  
*Note: To put RAM in a shutdown mode that removes all power from the RAM and reset the RAM contents, use the PWRSEQ_LPMEMSD register.* |
| 17   | sysram1ls | R/W    | 0     | **Sysram1 LIGHTSLEEP Enable** Data is unavailable for read/write operations in LIGHTSLEEP mode but is retained. See Table 4-13 RAM for base address and size information.  
0: ACTIVE mode.  
1: LIGHTSLEEP mode enabled.  
*Note: To put RAM in a shutdown mode that removes all power from the RAM and reset the RAM contents, use the PWRSEQ_LPMEMSD register.* |
| 16   | sysram0ls | R/W    | 0     | **Sysram0 LIGHTSLEEP Enable** Data is unavailable for read/write operations in LIGHTSLEEP mode but is retained. See Table 4-13 RAM for base address and size information.  
0: ACTIVE mode.  
1: LIGHTSLEEP mode enabled.  
*Note: To put RAM in a shutdown mode that removes all power from the RAM and reset the RAM contents, use the PWRSEQ_LPMEMSD register.* |
| 15:3 | -       | RO     | -     | **Reserved** Do not modify this field. |
| 2:0  | fws     | R/W    | 0b101 | **Program Flash Wait States** Number of wait-state cycles per Flash code read access.  
0: Invalid  
1 – 7: Number of Flash code access wait states  
*Note: For the 60MHz clock and slower, minimum wait state is 1.  
Note: For the 96MHz clock, the minimum wait states should be 2.* |

### Table 4-61: Memory Zeroization Control Register

<table>
<thead>
<tr>
<th>Memory Zeroization Control</th>
<th>GCR_MEM_ZERO</th>
<th>[0x002C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td>31:15</td>
<td>-</td>
<td>RO</td>
</tr>
</tbody>
</table>
| 14   | icache1z | R/W    | 0     | **CPU1 ICC1 Cache Data and Tag Zeroization** Write 1 to initiate the operation. Only valid on devices with optional CPU1.  
0: Operation complete.  
1: Operation in progress. |
<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>usbfifoz</td>
<td>R/W</td>
<td>0</td>
<td>USB FIFO Zeroization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to initiate the operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Operation complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Operation in progress.</td>
</tr>
<tr>
<td>12</td>
<td>cryptoz</td>
<td>R/W</td>
<td>0</td>
<td>Crypto MAA Memory Zeroization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to initiate the operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Operation complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Operation in progress.</td>
</tr>
<tr>
<td>11</td>
<td>scachetagz</td>
<td>R/W</td>
<td>0</td>
<td>SRCC Cache Tag Zeroization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to clear the SRCC tag RAM.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Operation complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Operation in progress.</td>
</tr>
<tr>
<td>10</td>
<td>scachedataz</td>
<td>R/W</td>
<td>0</td>
<td>SRCC Cache Data Zeroization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to initiate the operation to clear the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SFCC Data RAM to 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Operation complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Operation in progress.</td>
</tr>
<tr>
<td>9</td>
<td>icachexipz</td>
<td>R/W</td>
<td>0</td>
<td>SFCC Cache Data and Tag Zeroization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to clear the ICC1 16KB cache RAM to 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The bit is set to 0 when the operation is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Operation complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Operation in progress.</td>
</tr>
<tr>
<td>8</td>
<td>icache0z</td>
<td>R/W</td>
<td>0</td>
<td>CPU0 ICC0 Cache Data and Tag Zeroization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to initiate the operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Operation complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Operation in progress.</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>6</td>
<td>sram6z</td>
<td>R/W</td>
<td>0</td>
<td>Sysram6 to Sysram11 Zeroization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to initiate the operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Operation complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Operation in progress.</td>
</tr>
<tr>
<td>5</td>
<td>sram5z</td>
<td>R/W</td>
<td>0</td>
<td>Sysram5 Zeroization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to initiate the operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Operation complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Operation in progress.</td>
</tr>
<tr>
<td>4</td>
<td>sram4z</td>
<td>R/W</td>
<td>0</td>
<td>Sysram4 Zeroization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to initiate the operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Operation complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Operation in progress.</td>
</tr>
<tr>
<td>3</td>
<td>sram3z</td>
<td>R/W</td>
<td>0</td>
<td>Sysram3 Zeroization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to initiate the operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Operation complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Operation in progress.</td>
</tr>
<tr>
<td>2</td>
<td>sram2</td>
<td>R/W1</td>
<td>0</td>
<td>Sysram2 Zeroization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to initiate the operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Operation complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Operation in progress.</td>
</tr>
<tr>
<td>Memory Zeroization Control</td>
<td>GCR_MEM_ZERO</td>
<td>[0x002C]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------</td>
<td>----------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
<td>Reset</td>
<td>Description</td>
</tr>
<tr>
<td>1</td>
<td>sram1z</td>
<td>R/W1</td>
<td>0</td>
<td>Sysram1 Zeroization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to initiate the operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Operation complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Operation in progress.</td>
</tr>
<tr>
<td>0</td>
<td>sram0z</td>
<td>R/W1</td>
<td>0</td>
<td>Sysram0 Zeroization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to initiate the operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Operation complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Operation in progress.</td>
</tr>
</tbody>
</table>

Table 4-62: System Status Flag Register

<table>
<thead>
<tr>
<th>System Status Flag</th>
<th>GCR_SYS_STAT</th>
<th>[0x0040]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:6</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>scmemf</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4:2</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>codeinterr</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>icelock</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4-63: Reset Register 1

<table>
<thead>
<tr>
<th>Reset 1</th>
<th>GCR_RST1</th>
<th>[0x0044]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:26</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>simo</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>dvs</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td>23</td>
<td>htimer1</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>htimer0</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>rpu</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>i2c2</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>audio</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>btle</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>wdt2</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>sema</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>xipr</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14:10</td>
<td></td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>spi3</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>wdt1</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 4-64: Peripheral Clock Disable Register 1

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>cpu1</td>
<td>R/W</td>
<td>1</td>
<td>CPU1 Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Disabled.</td>
</tr>
<tr>
<td>30</td>
<td>-</td>
<td>RO</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>29</td>
<td>wdt2</td>
<td>R/W</td>
<td>1</td>
<td>WDT2 Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Disabled.</td>
</tr>
<tr>
<td>28</td>
<td>wdt1</td>
<td>R/W</td>
<td>1</td>
<td>Watchdog Timer 1 Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Disabled.</td>
</tr>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
<td>Reset</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>--------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| 27   | wdt0    | R/W    | 1     | **Watchdog Timer 0 Disable** Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked.  
0: Enabled.  
1: Disabled. |
| 26   | htimer1 | R/W    | 1     | **HTIMER1 Clock Disable** Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked.  
0: Enabled.  
1: Disabled. |
| 25   | htimer0 | R/W    | 1     | **HTIMER0 Clock Disable** Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked.  
0: Enabled.  
1: Disabled. |
| 24   | i2c2    | R/W    | 1     | **I2C2 Clock Disable** Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked.  
0: Enabled.  
1: Disabled. |
| 23   | audio   | R/W    | 1     | **Audio Interface Clock Disable** Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked.  
0: Enabled.  
1: Disabled. |
| 22   | -       | RO     | 1     | **Reserved** Do not modify this field.                                    |
| 21   | dma1    | R/W    | 1     | **DMA1 Clock Disable** Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked.  
0: Enabled.  
1: Disabled. |
| 20   | spixipr | R/W    | 1     | **SPIXR Clock Disable** Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked.  
0: Enabled.  
1: Disabled. |
| 19:15| -       | RO     | 1     | **Reserved** Do not modify this field.                                    |
| 14   | spi3    | R/W    | 1     | **QSPI3 Clock Disable** Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked.  
0: Enabled.  
1: Disabled. |
<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>ow</td>
<td>R/W</td>
<td>1</td>
<td>One-Wire Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Disabled.</td>
</tr>
<tr>
<td>12</td>
<td>icachexipf</td>
<td>R/W</td>
<td>1</td>
<td>SFCC Flash Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Disabled.</td>
</tr>
<tr>
<td>11</td>
<td>-</td>
<td>R/W</td>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>10</td>
<td>sdhc</td>
<td>R/W</td>
<td>1</td>
<td>SDHC Controller Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Disabled.</td>
</tr>
<tr>
<td>9</td>
<td>smphr</td>
<td>R/W</td>
<td>1</td>
<td>Semaphore Block Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Disabled.</td>
</tr>
<tr>
<td>8</td>
<td>sdma</td>
<td>R/W</td>
<td>1</td>
<td>Bluetooth Hardware Accelerator Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Disabled.</td>
</tr>
<tr>
<td>7</td>
<td>scache</td>
<td>R/W</td>
<td>1</td>
<td>SRCC Cache Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Disabled.</td>
</tr>
<tr>
<td>6:3</td>
<td>-</td>
<td>R/W</td>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>2</td>
<td>trng</td>
<td>R/W</td>
<td>1</td>
<td>TRNG Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Disabled.</td>
</tr>
<tr>
<td>1</td>
<td>uart2</td>
<td>R/W</td>
<td>1</td>
<td>UART2 Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling the clock disables functionality while also saving power. Associated register states are retained but read and write access is blocked.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Disabled.</td>
</tr>
</tbody>
</table>
### Peripheral Clock Disable 1

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>btle</td>
<td>R/W</td>
<td>1</td>
<td>Bluetooth Digital Baseband Clock Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Disabling the clock disables functionality while also saving power.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Associated register states are retained but read and write access is blocked.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Disabled.</td>
</tr>
</tbody>
</table>

**Table 4-65: Event Enable Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:6</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>5</td>
<td>cpu1txevent</td>
<td>R/W</td>
<td>0</td>
<td><strong>CPU1 TXEV On SEV Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When set, an SEV (Send Event) instruction causes a TXEV event from either CPU0 or CPU1 to cause an RXEV event to wake CPU1 from a low power mode entered with a WFE instruction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>4</td>
<td>cpu1dma1event</td>
<td>R/W</td>
<td>0</td>
<td><strong>CPU1 DMA1 CTZ Wake-Up Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Allows a DMA1 CTZ event to generate an RXEV to wake-up CPU1 from a low power mode entered with a WFE instruction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>3</td>
<td>cpu1dma0event</td>
<td>R/W</td>
<td>0</td>
<td><strong>CPU1 DMA0 CTZ Wake-Up Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Allows a DMA0 CTZ event to generate an RXEV to wake-up CPU1 from a low power mode entered with a WFE instruction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>2</td>
<td>cpu0txevent</td>
<td>R/W</td>
<td>0</td>
<td><strong>CPU0 TXEV On SEV Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When set, an SEV (Send Event) instruction causes a TXEV event from either CPU0 or CPU1 to cause an RXEV event to wake CPU1 from a low power mode entered with a WFE instruction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>1</td>
<td>cpu0dma1event</td>
<td>R/W</td>
<td>0</td>
<td><strong>CPU0 DMA1 CTZ Wake-Up Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Allows a DMA1 CTZ event to generate an RXEV to wake-up CPU0 from a low power mode entered with a WFE instruction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>0</td>
<td>cpu0dma0event</td>
<td>R/W</td>
<td>0</td>
<td><strong>CPU0 DMA0 CTZ Wake-Up Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Allows a DMA0 CTZ event to generate an RXEV to wake-up CPU0 from a low power mode entered with a WFE instruction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
</tbody>
</table>
Table 4-66: Revision Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>15:0</td>
<td>revision</td>
<td>RO</td>
<td>*</td>
<td><strong>Device Revision</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Returns the chip revision ID as a packed BCD. For example, 0xA1 would indicate the device is revision A1.</td>
</tr>
</tbody>
</table>

Table 4-67: System Status Interrupt Enable Register

<table>
<thead>
<tr>
<th>System Status Interrupt Enable</th>
<th>GCR_SYS_STAT_IE</th>
<th>[0x0054]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:6</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>scmfie</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4:2</td>
<td></td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>cieie</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>iceulie</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4-68: Error Correction Coding Error Register

<table>
<thead>
<tr>
<th>Error Correction Coding Error</th>
<th>GCR_ECC_ER</th>
<th>[0x0064]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:13</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td>12</td>
<td>f11eccerr</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>f00eccerr</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>icspixfecerr</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 4-69: Error Correction Not Double Error Detected Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:13</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>12</td>
<td>fl1eccnded</td>
<td>R/W1C</td>
<td>1</td>
<td>Flash1 Not Double ECC Error Detected</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When cleared, indicates that there is a single correctable error in the Flash1 bank. Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Error</td>
</tr>
</tbody>
</table>

### Error Correction Coding Error

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>ic1eccerr</td>
<td>R/W1C</td>
<td>0</td>
<td>Instruction Cache 1 ECC Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Error</td>
</tr>
<tr>
<td>8</td>
<td>ic0eccerr</td>
<td>R/W1C</td>
<td>0</td>
<td>Instruction Cache 0 ECC Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Error</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>6</td>
<td>sysram6eccerr</td>
<td>RO</td>
<td>0</td>
<td>System RAM6 ECC Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Error</td>
</tr>
<tr>
<td>5</td>
<td>sysram5eccerr</td>
<td>R/W1C</td>
<td>0</td>
<td>System RAM5 ECC Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Error</td>
</tr>
<tr>
<td>4</td>
<td>sysram4eccerr</td>
<td>R/W1C</td>
<td>0</td>
<td>System RAM4 ECC Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Error</td>
</tr>
<tr>
<td>3</td>
<td>sysram3eccerr</td>
<td>R/W1C</td>
<td>0</td>
<td>System RAM3 ECC Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Error</td>
</tr>
<tr>
<td>2</td>
<td>sysram2eccerr</td>
<td>R/W1C</td>
<td>0</td>
<td>System RAM2 ECC Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Error</td>
</tr>
<tr>
<td>1</td>
<td>sysram1eccerr</td>
<td>R/W1C</td>
<td>0</td>
<td>System RAM1 ECC Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Error</td>
</tr>
<tr>
<td>0</td>
<td>sysram0eccerr</td>
<td>R/W1C</td>
<td>0</td>
<td>System RAM0 ECC Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Error</td>
</tr>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
<td>Reset</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
<td>--------</td>
<td>-------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>11</td>
<td>fl0eccnded</td>
<td>R/W1C</td>
<td>1</td>
<td><strong>Flash0 Not Double ECC Error Detected</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When cleared, indicates that there is a single correctable error in the Flash0 bank. Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Error</td>
</tr>
<tr>
<td>10</td>
<td>icspixfeccnded</td>
<td>R/W1C</td>
<td>1</td>
<td><strong>SPIXF Instruction Cache Not Double ECC Error Detected</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When cleared, indicates that there is a single correctable error in SPIXF Instruction Cache. Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Error</td>
</tr>
<tr>
<td>9</td>
<td>ic1eccnded</td>
<td>R/W1C</td>
<td>1</td>
<td><strong>Instruction Cache 1 Not Double ECC Error Detected</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When cleared, indicates that there is a single correctable error in Instruction Cache 1. Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Error</td>
</tr>
<tr>
<td>8</td>
<td>ic0eccnded</td>
<td>R/W1C</td>
<td>1</td>
<td><strong>Instruction Cache 0 Not Double ECC Error Detected</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When cleared, indicates that there is a single correctable error in Instruction Cache 0. Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Error</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved</strong></td>
</tr>
<tr>
<td>6</td>
<td>sysram6eccnded</td>
<td>R/W1C</td>
<td>1</td>
<td><strong>System RAM6 Not Double ECC Error Detected</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When cleared, indicates that there is a single correctable error in the RAM6 block. Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Error</td>
</tr>
<tr>
<td>5</td>
<td>sysram5eccnded</td>
<td>R/W1C</td>
<td>1</td>
<td><strong>System RAM5 Not Double ECC Error Detected</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When cleared, indicates that there is a single correctable error in the RAM5 block. Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Error</td>
</tr>
<tr>
<td>4</td>
<td>sysram4eccnded</td>
<td>R/W1C</td>
<td>1</td>
<td><strong>System RAM4 Not Double ECC Error Detected</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When cleared, indicates that there is a single correctable error in the RAM4 block. Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: No Error</td>
</tr>
<tr>
<td>3</td>
<td>sysram3eccnded</td>
<td>R/W1C</td>
<td>1</td>
<td><strong>System RAM3 Not Double ECC Error Detected</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When cleared, indicates that there is a single correctable error in the RAM3 block. Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: No Error</td>
</tr>
<tr>
<td>2</td>
<td>sysram2eccnded</td>
<td>R/W1C</td>
<td>1</td>
<td><strong>System RAM2 Not Double ECC Error Detected</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When cleared, indicates that there is a single correctable error in the RAM2 block. Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: No Error</td>
</tr>
</tbody>
</table>
### Table 4-70: Error Correction Coding Not Double Error Detected

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>sysram1eccnded</td>
<td>R/W1C</td>
<td>1</td>
<td>System RAM1 Not Double ECC Error Detected</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When cleared, indicates that there is a single correctable error in the RAM1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>block. Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: No Error</td>
</tr>
<tr>
<td>0</td>
<td>sysram0eccnded</td>
<td>R/W1C</td>
<td>1</td>
<td>System RAM0 Not Double ECC Error Detected</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When cleared, indicates that there is a single correctable error in the RAM0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>block. Write to 1 to clear the flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: No Error</td>
</tr>
</tbody>
</table>

### Table 4-70: Error Correction Coding Interrupt Enable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:13</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved.</td>
</tr>
<tr>
<td>12</td>
<td>fl1eccen</td>
<td>R/W</td>
<td>0</td>
<td>Flash1 ECC Error Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>11</td>
<td>fl0eccen</td>
<td>R/W</td>
<td>0</td>
<td>Flash0 ECC Error Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>10</td>
<td>licxpecccen</td>
<td>R/W</td>
<td>0</td>
<td>SPIXF Instruction Cache ECC Error Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>9</td>
<td>ec1eccen</td>
<td>R/W</td>
<td>0</td>
<td>Instruction Cache 1 ECC Error Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>8</td>
<td>ec0eccen</td>
<td>R/W</td>
<td>0</td>
<td>Instruction Cache 0 ECC Error Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>7:6</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved.</td>
</tr>
<tr>
<td>5</td>
<td>eccsysram5en</td>
<td>R/W</td>
<td>0</td>
<td>Sysram5 ECC Error Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>4</td>
<td>eccsysram4en</td>
<td>R/W</td>
<td>0</td>
<td>Sysram4 ECC Error Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>3</td>
<td>eccsysram3en</td>
<td>R/W</td>
<td>0</td>
<td>Sysram3 ECC Error Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>2</td>
<td>eccsysram2en</td>
<td>R/W</td>
<td>0</td>
<td>Sysram2 ECC Error Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>1</td>
<td>eccsysram1en</td>
<td>R/W</td>
<td>0</td>
<td>Sysram1 ECC Error Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>0</td>
<td>eccsysram0en</td>
<td>R/W</td>
<td>0</td>
<td>Sysram0 ECC Error Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
</tbody>
</table>
### Table 4-71: Error Correction Coding Address Register

<table>
<thead>
<tr>
<th>Error Correction Coding Address</th>
<th>GCR_ECC_ERRAD</th>
<th>[0x0070]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
<td>Reset</td>
</tr>
<tr>
<td>31:0</td>
<td>tagramerr</td>
<td>RO</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>tagrambank</td>
<td>RO</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29:16</td>
<td>tagramaddr</td>
<td>RO</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>dataramerr</td>
<td>RO</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>datarambank</td>
<td>RO</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13:0</td>
<td>dataramaddr</td>
<td>RO</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 4-72: Bluetooth LDO Control Register

<table>
<thead>
<tr>
<th>Bluetooth LDO Control</th>
<th>GCR_BTLE_LDOCR</th>
<th>[0x0074]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
<td>Reset</td>
</tr>
<tr>
<td>31:16</td>
<td></td>
<td>RO</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Idotxbypendenly</td>
<td>R</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Idorxbypendenly</td>
<td>R</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Idorxendly</td>
<td>R</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Bluetooth LDO Control

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>ldotxendly</td>
<td>R</td>
<td>0</td>
<td>LDOTX Enable Delay</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Not used.</td>
</tr>
<tr>
<td>11</td>
<td>ldotxdisch</td>
<td>R/W</td>
<td>0</td>
<td>LDOTX Discharge</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit is used to discharge the LDOTX output using a strong pulldown. For use when switching between Bypass Mode and Regulation Mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No discharge</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Discharge</td>
</tr>
<tr>
<td>10</td>
<td>ldotxbyp</td>
<td>R/W</td>
<td>0</td>
<td>LDOTX Bypass Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit enables the LDOTX Bypass Mode and charge the LDOTX output voltage to its input voltage level.</td>
</tr>
<tr>
<td>9</td>
<td>ldorxdisch</td>
<td>R/W</td>
<td>0</td>
<td>LDORX Discharge</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit is used to discharge the LDORX output using a strong pulldown. For use when switching between Bypass Mode and Regulation Mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No discharge</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Discharge</td>
</tr>
<tr>
<td>8</td>
<td>ldorxbyp</td>
<td>R/W</td>
<td>0</td>
<td>LDORX Bypass Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit enables the LDORX Bypass Mode and charges the LDORX output voltage to its input voltage level.</td>
</tr>
<tr>
<td>7:6</td>
<td>ldorxvsel</td>
<td>R/W</td>
<td>b’01</td>
<td>LDORX Output Voltage Setting</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00: 0.7V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01: 0.85V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b10: 0.9V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b11: 1.1V</td>
</tr>
<tr>
<td>5</td>
<td>ldorxpulld</td>
<td>R/W</td>
<td>1</td>
<td>LDORX Pulldown</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit is used to provide a weak pulldown to the LDORX output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Pulldown disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Pulldown enabled</td>
</tr>
<tr>
<td>4</td>
<td>ldorxen</td>
<td>R/W</td>
<td>0</td>
<td>LDORX Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Enable the LDO and charge its output voltage to the setting in ( GCR_BTLE_LDOCR ).ldorxvsel.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>3:2</td>
<td>ldotxvsel</td>
<td>R/W</td>
<td>b’01</td>
<td>LDOTX Output Voltage Setting</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00: 0.7V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01: 0.85V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b10: 0.9V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b11: 1.1V</td>
</tr>
<tr>
<td>1</td>
<td>ldotxpulld</td>
<td>R/W</td>
<td>1</td>
<td>LDOTX Pull Down</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting this bit enables a weak pulldown on the output of the TX LDO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Pulldown disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Pulldown enabled</td>
</tr>
<tr>
<td>0</td>
<td>ldotxen</td>
<td>R/W</td>
<td>0</td>
<td>LDOTX Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Enable the LDO and charge its output voltage to the setting in ( GCR_BTLE_LDOCR ).ldotxvsel.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
</tbody>
</table>
### Table 4-73: Bluetooth LDO Delay Count Register

<table>
<thead>
<tr>
<th>Bluetooth LDO Delay Count</th>
<th>GCR_BTLE_LDODCR</th>
<th>[0x0078]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:29</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td>28:20</td>
<td>ldotxdlycnt</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19:17</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td>16:8</td>
<td>ldorxdlycnt</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>bypdlycnt</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 4-74: General Purpose 0 Register

<table>
<thead>
<tr>
<th>General Purpose 0</th>
<th>GCR_GPR0</th>
<th>[0x0080]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>gpr0</td>
<td>R/W</td>
</tr>
</tbody>
</table>

### Table 4-75: Arm Peripheral Bus Asynchronous Bridge Select Register

<table>
<thead>
<tr>
<th>Arm Peripheral Bus Asynchronous Bridge Select</th>
<th>GCR_APB_ASYNC</th>
<th>[0x0084]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:4</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>apbasyncpt</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>apbasyncl2C2</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>apbasyncl2C1</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4.18  Function Control Registers

See Table 3-1: APB Peripheral Base Address Map for this module's base address. Unless specified otherwise, all fields are reset only on a system reset or POR, but not a soft reset.

Table 4-76: Function Control Register Summary

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>GCR_FCR</td>
<td>Function Control Register</td>
</tr>
</tbody>
</table>

4.19  Function Control Register Details

Table 4-77: Function Control Register 0

<table>
<thead>
<tr>
<th>Function Control 0</th>
<th>GCR_FCR</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>--------</td>
</tr>
</tbody>
</table>
| 31:26 | - | RO | 0 | Reserved  
Do not modify this field. |
| 25 | i2c2_scl_filter_en | R/W | 0 | I2C2 SCL Glitch Filter Enable  
0: Disabled  
1: Enabled |
| 24 | i2c2_sda_filter_en | R/W | 0 | I2C2 SDA Glitch Filter Enable  
0: Disabled  
1: Enabled |
| 23 | i2c1_scl_filter_en | R/W | 0 | I2C1 SCL Glitch Filter Enable  
0: Disabled  
1: Enabled |
| 22 | i2c1_sda_filter_en | R/W | 0 | I2C1 SDA Glitch Filter Enable  
0: Disabled  
1: Enabled |
| 21 | i2c0_scl_filter_en | R/W | 0 | I2C0 SCL Glitch Filter Enable  
0: Disabled  
1: Enabled |
| 20 | i2c0_sda_filter_en | R/W | 0 | I2C0 SDA Glitch Filter Enable  
0: Disabled  
1: Enabled |
| 19:18 | - | RO | 0 | Reserved  
Do not modify this field. |
### Function Control 0

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>qspi0_fnc_sel</td>
<td>R/W</td>
<td>0</td>
<td>QSPI0 Function Select</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: High speed 96MHz oscillator</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: External clock input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: See the GPIO chapter for the external clock input pin</td>
</tr>
<tr>
<td>16</td>
<td>usb_clk_sel</td>
<td>R/W</td>
<td>0</td>
<td>USB Reference Clock Source Select</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: High speed 96MHz oscillator</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: External clock input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>See the GPIO chapter for the external clock input pin</td>
</tr>
<tr>
<td>15:0</td>
<td></td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
</tbody>
</table>

#### 4.20 AES Key Registers

See *Table 3-1: APB Peripheral Base Address Map* for the AES Key Registers’ Peripheral Base Address.

*Table 4-78: AES Key Register Summary*

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>AES_KEY0</td>
<td>128-bit AES Key Register 0</td>
</tr>
<tr>
<td>[0x0080]</td>
<td>AES_KEY1</td>
<td>128-bit AES Key Register 1</td>
</tr>
<tr>
<td>[0x0100]</td>
<td>AES_KEY2</td>
<td>128-bit AES Key Register 2</td>
</tr>
<tr>
<td>[0x0180]</td>
<td>AES_KEY3</td>
<td>128-bit AES Key Register 3</td>
</tr>
</tbody>
</table>

#### 4.21 AES Key Register Details

*Table 4-79: AES Key 0 and 1 Registers*

<table>
<thead>
<tr>
<th>AES Key 0</th>
<th>AES_KEY0</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>127:0</td>
<td>aes_key</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 4-80: AES Key 2 and 3 Registers*

<table>
<thead>
<tr>
<th>AES Key 2</th>
<th>AES_KEY2</th>
<th>[0x0100]</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES Key 3</td>
<td>AES_KEY3</td>
<td>[0x0180]</td>
</tr>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>127:0</td>
<td>aes_key</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5. Interrupts and Exceptions

Interrupts and exceptions are managed by the Arm Cortex-M4 with FPU Nested Vector Interrupt Controller (NVIC). The NVIC handles the interrupts, exceptions, priorities and masking. Table 5-1 details the MAX32665—MAX32668 interrupt vector table and describes each exception and interrupt.

5.1 Features

- 59 maskable interrupts not including the 15 system exceptions of the Arm Cortex-M4 with FPU
- 8 programmable priority levels
- Nested exception and interrupt support
- Interrupt masking

5.2 Interrupt Vector Table

Table 5-1 lists the interrupt and exception table for the MAX32665—MAX32668. There are 95 interrupt entries for the MAX32665—MAX32668, 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 110.

Table 5-1: MAX32665—MAX32668 Interrupt Vector Table

<table>
<thead>
<tr>
<th>Exception (Interrupt) Number</th>
<th>Offset</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[0x0004]</td>
<td>Reset_Handler</td>
<td>Reset</td>
</tr>
<tr>
<td>2</td>
<td>[0x0008]</td>
<td>NMI_Handler</td>
<td>Non-Maskable Interrupt</td>
</tr>
<tr>
<td>3</td>
<td>[0x000C]</td>
<td>HardFault_Handler</td>
<td>Hard Fault</td>
</tr>
<tr>
<td>4</td>
<td>[0x0010]</td>
<td>MemManage_Handler</td>
<td>Memory Management Fault</td>
</tr>
<tr>
<td>5</td>
<td>[0x0014]</td>
<td>BusFault_Handler</td>
<td>Bus Fault</td>
</tr>
<tr>
<td>6</td>
<td>[0x0018]</td>
<td>UsageFault_Handler</td>
<td>Usage Fault</td>
</tr>
<tr>
<td>7:10</td>
<td>[0x001C]-[0x0028]</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>[0x002C]</td>
<td>SVC_Handler</td>
<td>Supervisor Call Exception</td>
</tr>
<tr>
<td>12</td>
<td>[0x0030]</td>
<td>DebugMon_Handler</td>
<td>Debug Monitor Exception</td>
</tr>
<tr>
<td>13</td>
<td>[0x0034]</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>14</td>
<td>[0x0038]</td>
<td>PendSV_Handler</td>
<td>Request Pending for System Service</td>
</tr>
<tr>
<td>15</td>
<td>[0x003C]</td>
<td>SysTick_Handler</td>
<td>System Tick Timer</td>
</tr>
<tr>
<td>16</td>
<td>[0x0040]</td>
<td>SysFault_IRQHandler</td>
<td>System Fault interrupt</td>
</tr>
<tr>
<td>17</td>
<td>[0x0044]</td>
<td>WDTO_IRQHandler</td>
<td>Watchdog Timer 0 Interrupt</td>
</tr>
<tr>
<td>18</td>
<td>[0x0048]</td>
<td>USB_IRQHandler</td>
<td>USB Interrupt</td>
</tr>
<tr>
<td>19</td>
<td>[0x004C]</td>
<td>RTC_IRQHandler</td>
<td>Real-Time Clock Interrupt</td>
</tr>
<tr>
<td>20</td>
<td>[0x0050]</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>Exception (Interrupt) Number</td>
<td>Offset</td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------</td>
<td>---------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>21</td>
<td>[0x0054]</td>
<td>TMR0_IRQHandler</td>
<td>Timer 0 Interrupt</td>
</tr>
<tr>
<td>22</td>
<td>[0x0058]</td>
<td>TMR1_IRQHandler</td>
<td>Timer 1 Interrupt</td>
</tr>
<tr>
<td>23</td>
<td>[0x005C]</td>
<td>TMR2_IRQHandler</td>
<td>Timer 2 Interrupt</td>
</tr>
<tr>
<td>24</td>
<td>[0x0060]</td>
<td>TMR3_IRQHandler</td>
<td>Timer 3 Interrupt</td>
</tr>
<tr>
<td>25</td>
<td>[0x0064]</td>
<td>TMR4_IRQHandler</td>
<td>Timer 4 Interrupt</td>
</tr>
<tr>
<td>26</td>
<td>[0x0068]</td>
<td>TMR5_IRQHandler</td>
<td>Timer 5 Interrupt</td>
</tr>
<tr>
<td>27</td>
<td>[0x006C]</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>28</td>
<td>[0x0070]</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>29</td>
<td>[0x0074]</td>
<td>I2C0_IRQHandler</td>
<td>I2C Port 0 Interrupt</td>
</tr>
<tr>
<td>30</td>
<td>[0x0078]</td>
<td>UART0_IRQHandler</td>
<td>UART Port 0 Interrupt</td>
</tr>
<tr>
<td>31</td>
<td>[0x007C]</td>
<td>UART1_IRQHandler</td>
<td>UART Port 1 Interrupt</td>
</tr>
<tr>
<td>32</td>
<td>[0x0080]</td>
<td>SPI1_IRQHandler</td>
<td>SPI Port 1 Interrupt</td>
</tr>
<tr>
<td>33</td>
<td>[0x0084]</td>
<td>SPI2_IRQHandler</td>
<td>SPI Port 2 Interrupt</td>
</tr>
<tr>
<td>34</td>
<td>[0x0088]</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>35</td>
<td>[0x008C]</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>36</td>
<td>[0x0090]</td>
<td>ADC_IRQHandler</td>
<td>ADC Interrupt</td>
</tr>
<tr>
<td>37:38</td>
<td>[0x0094]:[0x098]</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>39</td>
<td>[0x009C]</td>
<td>FLC_IRQHandler</td>
<td>Flash Controller Interrupt</td>
</tr>
<tr>
<td>40</td>
<td>[0x00A0]</td>
<td>GPIO0_IRQHandler</td>
<td>GPIO Port 0 Interrupt</td>
</tr>
<tr>
<td>41</td>
<td>[0x00A4]</td>
<td>GPIO1_IRQHandler</td>
<td>GPIO Port 1 Interrupt</td>
</tr>
<tr>
<td>42</td>
<td>[0x00A8]</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>43</td>
<td>[0x00AC]</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>44</td>
<td>[0x00B0]</td>
<td>DMA0_IRQHandler</td>
<td>DMA0 Interrupt</td>
</tr>
<tr>
<td>45</td>
<td>[0x00B4]</td>
<td>DMA1_IRQHandler</td>
<td>DMA1 Interrupt</td>
</tr>
<tr>
<td>46</td>
<td>[0x00B8]</td>
<td>DMA2_IRQHandler</td>
<td>DMA2 Interrupt</td>
</tr>
<tr>
<td>47</td>
<td>[0x00BC]</td>
<td>DMA3_IRQHandler</td>
<td>DMA3 Interrupt</td>
</tr>
<tr>
<td>48:49</td>
<td>[0x00C0 : 0x00C4]</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>50</td>
<td>[0x00C8]</td>
<td>UART2_IRQHandler</td>
<td>UART Port 2 Interrupt</td>
</tr>
<tr>
<td>51</td>
<td>[0x00CC]</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>Exception (Interrupt) Number</td>
<td>Offset</td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------</td>
<td>-----------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>52</td>
<td>[0x00D0]</td>
<td>I2C1_IRQHandler</td>
<td>I2C Port 1 Interrupt</td>
</tr>
<tr>
<td>53</td>
<td>[0x00D4]</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>54</td>
<td>[0x00D8]</td>
<td>SPIXC_IRQHandler</td>
<td>SPI XIP Interrupt</td>
</tr>
<tr>
<td>55</td>
<td>[0x00DC]</td>
<td>BTLE_TX_DONE_IRQHandler</td>
<td>Bluetooth Transmitter Done Interrupt</td>
</tr>
<tr>
<td>56</td>
<td>[0x00E0]</td>
<td>BTLE_RX_RCVD_IRQHandler</td>
<td>Bluetooth Receive Data Interrupt</td>
</tr>
<tr>
<td>57</td>
<td>[0x00E4]</td>
<td>BTLE_RX_ENG_DET_IRQHandler</td>
<td>Bluetooth Receive Energy Detected Interrupt</td>
</tr>
<tr>
<td>58</td>
<td>[0x00E8]</td>
<td>BTLE_SFD_DET_IRQHandler</td>
<td>BTLE SFD Detected</td>
</tr>
<tr>
<td>59</td>
<td>[0x00EC]</td>
<td>BTLE_SFD_TO_IRQHandler</td>
<td>BTLE SFD Timeout</td>
</tr>
<tr>
<td>60</td>
<td>[0x00F0]</td>
<td>BTLE_GP_EVENT_IRQHandler</td>
<td>BTLE Timestamp</td>
</tr>
<tr>
<td>61</td>
<td>[0x00F4]</td>
<td>BTLE_CFO_IRQHandler</td>
<td>BTLE CFO Done</td>
</tr>
<tr>
<td>62</td>
<td>[0x00F8]</td>
<td>BTLE_SIG_DET_IRQHandler</td>
<td>BTLE Signal Detected</td>
</tr>
<tr>
<td>63</td>
<td>[0x00FC]</td>
<td>BTLE_AGC_EVENT_IRQHandler</td>
<td>BTLE AGC Event</td>
</tr>
<tr>
<td>64</td>
<td>[0x0100]</td>
<td>BTLE_RFFE_SPIM_IRQHandler</td>
<td>BTLE RFFE SPIM Done</td>
</tr>
<tr>
<td>65</td>
<td>[0x0104]</td>
<td>BTLE_TX_AES_IRQHandler</td>
<td>BTLE TX AES Done</td>
</tr>
<tr>
<td>66</td>
<td>[0x0108]</td>
<td>BTLE_RX_AES_IRQHandler</td>
<td>BTLE RX AES Done</td>
</tr>
<tr>
<td>67</td>
<td>[0x010C]</td>
<td>BTLE_INV_APB_ADDR_IRQHandler</td>
<td>BTLE Invalid APB Address</td>
</tr>
<tr>
<td>68</td>
<td>[0x0110]</td>
<td>BTLE_IQ_DATA_VALID_IRQHandler</td>
<td>BTLE IQ Data Valid</td>
</tr>
<tr>
<td>69</td>
<td>[0x0114]</td>
<td>WUT_IRQHandler</td>
<td>Wakeup Timer Interrupt</td>
</tr>
<tr>
<td>70</td>
<td>[0x0118]</td>
<td>GPIO_WAKE_IRQHandler</td>
<td>GPIO or USB Wakeup Interrupt</td>
</tr>
<tr>
<td>71</td>
<td>[0x011C]</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>72</td>
<td>[0x0120]</td>
<td>SPI0_IRQHandler</td>
<td>SPI Port 0 AHB Interrupt</td>
</tr>
<tr>
<td>73</td>
<td>[0x0124]</td>
<td>WDT1_IRQHandler</td>
<td>Watchdog Timer 1 Interrupt</td>
</tr>
<tr>
<td>74</td>
<td>[0x0128]</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>75</td>
<td>[0x012C]</td>
<td>PT_IRQHandler</td>
<td>Pulse Train Interrupt</td>
</tr>
<tr>
<td>76</td>
<td>[0x0130]</td>
<td>SDMA0_IRQHandler</td>
<td>Smart DMA Interrupt</td>
</tr>
<tr>
<td>77</td>
<td>[0x0134]</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>78</td>
<td>[0x0138]</td>
<td>I2C2_IRQHandler</td>
<td>I2C Port 2 Interrupt</td>
</tr>
<tr>
<td>79:81</td>
<td>[0x0138 : 0x0144]</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>Exception (Interrupt) Number</td>
<td>Offset</td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-----------</td>
<td>-------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>82</td>
<td>[0x0148]</td>
<td>SDHC_IRQHandler</td>
<td>SDHC Interrupt</td>
</tr>
<tr>
<td>83</td>
<td>[0x014C]</td>
<td>OWM_IRQHandler</td>
<td>1-Wire Master Interrupt</td>
</tr>
<tr>
<td>84</td>
<td>[0x0150]</td>
<td>DMA4_IRQHandler</td>
<td>DMA4 Interrupt</td>
</tr>
<tr>
<td>85</td>
<td>[0x0154]</td>
<td>DMA5_IRQHandler</td>
<td>DMA5 Interrupt</td>
</tr>
<tr>
<td>86</td>
<td>[0x0158]</td>
<td>DMA6_IRQHandler</td>
<td>DMA6 Interrupt</td>
</tr>
<tr>
<td>87</td>
<td>[0x015C]</td>
<td>DMA7_IRQHandler</td>
<td>DMA7 Interrupt</td>
</tr>
<tr>
<td>88</td>
<td>[0x0160]</td>
<td>DMA8_IRQHandler</td>
<td>DMA8 Interrupt</td>
</tr>
<tr>
<td>89</td>
<td>[0x0164]</td>
<td>DMA9_IRQHandler</td>
<td>DMA9 Interrupt</td>
</tr>
<tr>
<td>90</td>
<td>[0x0168]</td>
<td>DMA10_IRQHandler</td>
<td>DMA10 Interrupt</td>
</tr>
<tr>
<td>91</td>
<td>[0x016C]</td>
<td>DMA11_IRQHandler</td>
<td>DMA11 Interrupt</td>
</tr>
<tr>
<td>92</td>
<td>[0x0170]</td>
<td>DMA12_IRQHandler</td>
<td>DMA12 Interrupt</td>
</tr>
<tr>
<td>93</td>
<td>[0x0174]</td>
<td>DMA13_IRQHandler</td>
<td>DMA13 Interrupt</td>
</tr>
<tr>
<td>94</td>
<td>[0x0178]</td>
<td>DMA14_IRQHandler</td>
<td>DMA14 Interrupt</td>
</tr>
<tr>
<td>95</td>
<td>[0x017C]</td>
<td>DMA15_IRQHandler</td>
<td>DMA15 Interrupt</td>
</tr>
<tr>
<td>96</td>
<td>[0x0180]</td>
<td>USBDMA_IRQHandler</td>
<td>USB DMA Interrupt</td>
</tr>
<tr>
<td>97</td>
<td>[0x0184]</td>
<td>WDT2_IRQHandler</td>
<td>Watchdog Timer 2 Interrupt</td>
</tr>
<tr>
<td>98</td>
<td>[0x0188]</td>
<td>ECC_IRQHandler</td>
<td>Error Correction Coding Block Interrupt</td>
</tr>
<tr>
<td>99</td>
<td>[0x018C]</td>
<td>DVS_IRQHandler</td>
<td>DVS Controller Interrupt</td>
</tr>
<tr>
<td>100</td>
<td>[0x0190]</td>
<td>SIMO_IRQHandler</td>
<td>SIMO Controller Interrupt</td>
</tr>
<tr>
<td>101</td>
<td>[0x0194]</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>102</td>
<td>[0x0198]</td>
<td>AUDIO_IRQHandler</td>
<td>Audio Interface Interrupt</td>
</tr>
<tr>
<td>103</td>
<td>[0x019C]</td>
<td>FLC1_IRQHandler</td>
<td>FLASH Controller 1 Interrupt</td>
</tr>
<tr>
<td>104:108</td>
<td>[0x01A0 : 0x01B0]</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>109</td>
<td>[0x01B4]</td>
<td>HTMR0_IRQHandler</td>
<td>HTimer 0 Interrupt</td>
</tr>
<tr>
<td>110</td>
<td>[0x01B8]</td>
<td>HTMR1_IRQHandler</td>
<td>HTimer 1 Interrupt</td>
</tr>
</tbody>
</table>
6. General-Purpose I/O and Alternate Function Pins (GPIO)

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

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

GPIO are logically divided into ports of 32 pins. Some devices and 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.

The features for each GPIO pin include:

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

6.1 Instances

Table 6-1 shows the number of GPIO available on each IC package. Some packages and part numbers do not implement all bits of a 32-bit GPIO port. Register fields corresponding to unimplemented GPIO contain indeterminate values and should not be modified.

Table 6-1: MAX32665—MAX32668 GPIO Pin Count

<table>
<thead>
<tr>
<th>PACKAGE</th>
<th>GPIO</th>
<th>PINS</th>
<th>ALTERNATE FUNCTION MATRIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>109 WLP</td>
<td>GPIO0[31:0]</td>
<td>32</td>
<td>Table 6-2</td>
</tr>
<tr>
<td></td>
<td>GPIO1[15:0]</td>
<td>16</td>
<td>Table 6-2</td>
</tr>
<tr>
<td>121 CTBGA</td>
<td>GPIO0[31:0]</td>
<td>32</td>
<td>Table 6-2</td>
</tr>
<tr>
<td></td>
<td>GPIO1[15:0]</td>
<td>16</td>
<td>Table 6-2</td>
</tr>
</tbody>
</table>
Table 6-2 shows the alternate functions mapped to each GPIO pin.

### Table 6-2: MAX32665—MAX32668 GPIO and Alternate Function Matrix, 140 WLP

<table>
<thead>
<tr>
<th>GPIO Port[pin]</th>
<th>GPIO</th>
<th>ALTERNATE FUNCTION 1</th>
<th>ALTERNATE FUNCTION 2</th>
<th>ALTERNATE FUNCTION 3</th>
<th>ALTERNATE FUNCTION 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIO0[0]</td>
<td>P0.0</td>
<td>SPIXF_SS0</td>
<td></td>
<td>UART2_CTS</td>
<td>TMR0</td>
</tr>
<tr>
<td>GPIO0[1]</td>
<td>P0.1</td>
<td>SPIXF_MOSI/SDIO0</td>
<td></td>
<td>UART2_TX</td>
<td>TMR1</td>
</tr>
<tr>
<td>GPIO0[2]</td>
<td>P0.2</td>
<td>SPIXF_MISO/SDIO1</td>
<td></td>
<td>UART2_RX</td>
<td>TMR2</td>
</tr>
<tr>
<td>GPIO0[3]</td>
<td>P0.3</td>
<td>SPIXF_SCK</td>
<td></td>
<td>UART2_RTS</td>
<td>TMR3</td>
</tr>
<tr>
<td>GPIO0[4]</td>
<td>P0.4</td>
<td>SPIXF_SDIO2</td>
<td></td>
<td>OWM_IO</td>
<td>TMR4</td>
</tr>
<tr>
<td>GPIO0[5]</td>
<td>P0.5</td>
<td>SPIXF_SDIO3</td>
<td></td>
<td>OWM_PE</td>
<td>TMR5</td>
</tr>
<tr>
<td>GPIO0[6]</td>
<td>P0.6</td>
<td>I2C0_SCL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPIO0[7]</td>
<td>P0.7</td>
<td>I2C0_SDA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPIO0[8]</td>
<td>P0.8</td>
<td>SPIX SF5_SSO</td>
<td>QSPI0_SS0</td>
<td>UART0_CTS</td>
<td>TMR2</td>
</tr>
<tr>
<td>GPIO0[9]</td>
<td>P0.9</td>
<td>SPIX R_SSD</td>
<td>QSPI0_SSD</td>
<td>UART0_NOW</td>
<td>TMR2</td>
</tr>
<tr>
<td>GPIO0[10]</td>
<td>P0.10</td>
<td>SPIX M_ISO/SDIO0</td>
<td>QSPI0_M_ISO/SDIO0</td>
<td>UART0_TX</td>
<td>TMR3</td>
</tr>
<tr>
<td>GPIO0[11]</td>
<td>P0.11</td>
<td>SPIX R_SCK</td>
<td>QSPI0_SCK</td>
<td>UART0_RTS</td>
<td>TMR5</td>
</tr>
<tr>
<td>GPIO0[12]</td>
<td>P0.12</td>
<td>SPIX R_SDIO2</td>
<td>QSPI0_SDIO2</td>
<td>OWM_IO</td>
<td>TMR0</td>
</tr>
<tr>
<td>GPIO0[13]</td>
<td>P0.13</td>
<td>SPIX R_SDIO3</td>
<td>QSPI0_SDIO3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPIO0[14]</td>
<td>P0.14</td>
<td>I2C1_SCL</td>
<td>QSPI0_SSD3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPIO0[15]</td>
<td>P0.15</td>
<td>I2C1_SDA</td>
<td>QSPI0_SSD3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPIO0[16]</td>
<td>P0.16</td>
<td>AIN0/AIN0N</td>
<td>QSPI1_SSO</td>
<td>UART0_CTS</td>
<td>TMR2</td>
</tr>
<tr>
<td>GPIO0[17]</td>
<td>P0.17</td>
<td>AIN1/AIN10N</td>
<td>QSPI1_MOSI/SDIO0</td>
<td>UART0_NOW</td>
<td>TMR2</td>
</tr>
<tr>
<td>GPIO0[18]</td>
<td>P0.18</td>
<td>AIN2/AIN11N</td>
<td>QSPI1_MISO/SDIO1</td>
<td>UART0_CTS</td>
<td>TMR2</td>
</tr>
<tr>
<td>GPIO0[19]</td>
<td>P0.19</td>
<td>AIN3/AIN1P</td>
<td>QSPI1_SSD</td>
<td>UART0_RTS</td>
<td>TMR3</td>
</tr>
<tr>
<td>GPIO0[20]</td>
<td>P0.20</td>
<td>AIN4/AIN2N</td>
<td>QSPI1_SDIO2</td>
<td>UART1_RX</td>
<td>TMR2</td>
</tr>
<tr>
<td>GPIO0[21]</td>
<td>P0.21</td>
<td>AIN5/AIN2P</td>
<td>QSPI1_SDIO3</td>
<td>UART1_RX</td>
<td>TMR3</td>
</tr>
<tr>
<td>GPIO0[22]</td>
<td>P0.22</td>
<td>AIN6/AIN3N</td>
<td>QSPI1_SSD1</td>
<td>UART1_CTS</td>
<td>TMR4</td>
</tr>
<tr>
<td>GPIO0[23]</td>
<td>P0.23</td>
<td>AIN7/AIN3P</td>
<td>QSPI1_SSD2</td>
<td>UART1_RTS</td>
<td>TMR5</td>
</tr>
<tr>
<td>GPIO0[24]</td>
<td>P0.24</td>
<td>PCM_LRCLK</td>
<td>QSPI2_SSD0</td>
<td>UART0_NOW</td>
<td>TMR2</td>
</tr>
<tr>
<td>GPIO0[25]</td>
<td>P0.25</td>
<td>PCM_DOOUT</td>
<td>QSPI2_MOSI/SDIO0</td>
<td>UART0_NOW</td>
<td>TMR2</td>
</tr>
<tr>
<td>GPIO0[26]</td>
<td>P0.26</td>
<td>PCM_DIN</td>
<td>QSPI2_MISO/SDIO1</td>
<td>UART0_CTS</td>
<td>TMR2</td>
</tr>
<tr>
<td>GPIO0[27]</td>
<td>P0.27</td>
<td>PCM_BCLK</td>
<td>QSPI2_SCD</td>
<td>UART0_CTS</td>
<td>TMR2</td>
</tr>
<tr>
<td>GPIO0[28]</td>
<td>P0.28</td>
<td>PDM_DATA2</td>
<td>QSPI2_SDIO2</td>
<td>UART0_CTS</td>
<td>TMR2</td>
</tr>
<tr>
<td>GPIO0[29]</td>
<td>P0.29</td>
<td>PDM_DATA3</td>
<td>QSPI2_SDIO3</td>
<td>UART0_CTS</td>
<td>TMR2</td>
</tr>
<tr>
<td>GPIO0[30]</td>
<td>P0.30</td>
<td>PDM_RX_CLK</td>
<td>QSPI2_SS1</td>
<td>UART0_CTS</td>
<td>TMR2</td>
</tr>
<tr>
<td>GPIO0[31]</td>
<td>P0.31</td>
<td>PDM_MCLK</td>
<td>QSPI2_SS2</td>
<td>UART0_CTS</td>
<td>TMR2</td>
</tr>
<tr>
<td>GPIO1[0]</td>
<td>P1.0</td>
<td>SDHC_DAT3</td>
<td></td>
<td>SDMA_TMS</td>
<td>PT0</td>
</tr>
<tr>
<td>GPIO1[1]</td>
<td>P1.1</td>
<td>SDHC_CMD</td>
<td></td>
<td>SDMA_TDO</td>
<td>PT1</td>
</tr>
<tr>
<td>GPIO1[2]</td>
<td>P1.2</td>
<td>SDHC_DAT0</td>
<td></td>
<td>SDMA_TDI</td>
<td>PT2</td>
</tr>
<tr>
<td>GPIO1[3]</td>
<td>P1.3</td>
<td>SDHC_CLK</td>
<td></td>
<td>SDMA_TCK</td>
<td>PT3</td>
</tr>
<tr>
<td>GPIO1[4]</td>
<td>P1.4</td>
<td>SDHC_DAT1</td>
<td></td>
<td>UART0_RX</td>
<td>PT4</td>
</tr>
<tr>
<td>GPIO1[5]</td>
<td>P1.5</td>
<td>SDHC_DAT2</td>
<td></td>
<td>UART0_TX</td>
<td>PT5</td>
</tr>
<tr>
<td>GPIO1[6]</td>
<td>P1.6</td>
<td>SDHC_WP</td>
<td></td>
<td>UART0_CTS</td>
<td>PT6</td>
</tr>
</tbody>
</table>
Each device pin can be individually configured as a GPIO or an alternate function as shown in Table 6-3. The correct alternate function setting must be selected for each pin of a given multi-pin peripheral for proper operation.

**Table 6-3: MAX32665—MAX32668 GPIO Pin Configuration**

<table>
<thead>
<tr>
<th>MODE</th>
<th>GPIOn_EN0</th>
<th>GPIOn_EN1</th>
<th>GPIOn_EN2</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AF2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>AF3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>AF4</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>I/O (transition to AF1)</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>I/O (transition to AF2)</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>I/O (transition to AF3)</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>I/O (transition to AF4)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 6-4** shows the configuration options for digital I/O in input mode. Refer to the data sheet for details of specific electrical characteristics.

**Table 6-4: MAX32665—MAX32668 Input Mode Configuration**

<table>
<thead>
<tr>
<th>Input Mode</th>
<th>Mode Select</th>
<th>Pullup/Pulldown Strength</th>
<th>Power Supply</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-impedance</td>
<td>0</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>Weak Pull-up to VDDIO (1MΩ)</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Strong Pull-up to VDDIO (25KΩ)</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Weak Pull-down to VDDIO (1MΩ)</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Strong Pull-down to VDDIO (25KΩ)</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Reserved</td>
<td>1</td>
<td>1</td>
<td>NA</td>
</tr>
</tbody>
</table>

**Table 6-5** shows the configuration options for digital I/O in output mode. Refer to the data sheet for details of specific electrical characteristics.
Table 6-5: MAX32665—MAX32668 Output Mode Configuration

<table>
<thead>
<tr>
<th>Input Mode</th>
<th>Drive Strength</th>
<th>Power Supply</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GPIOn_DS_SEL1[pin]</td>
<td>GPIOn_DS_SEL0[pin]</td>
</tr>
<tr>
<td>Output Drive Strength 0, VDDIO Supply</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Output Drive Strength 1, VDDIO Supply</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Output Drive Strength 2, VDDIO Supply</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Output Drive Strength 3, VDDIO Supply</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Output Drive Strength 0, VDDIOH Supply</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Output Drive Strength 1, VDDIOH Supply</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Output Drive Strength 2, VDDIOH Supply</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Output Drive Strength 3, VDDIOH Supply</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Each GPIO port is assigned a dedicated interrupt vector as shown in the following table.

Table 6-6: MAX32665—MAX32668 GPIO Port Interrupt Vector Mapping

<table>
<thead>
<tr>
<th>GPIO Interrupt Source</th>
<th>GPIO Interrupt Status Register</th>
<th>Device Specific Interrupt Vector Number</th>
<th>GPIO Interrupt Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIO0[31:0]</td>
<td>GPIO0_INT_STAT</td>
<td>40</td>
<td>GPIO0_IRQHandler</td>
</tr>
<tr>
<td>GPIO1[15:0]</td>
<td>GPIO1_INT_STAT</td>
<td>41</td>
<td>GPIO1_IRQHandler</td>
</tr>
</tbody>
</table>
6.2 Usage

6.2.1 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:

- The GPIO Configuration Enable bits shown in Table 6-3 are set to I/O (transition to AF1) mode.
- Input mode is enabled ($GPIOn_IN_EN[pin] = 1$).
- High impedance mode enabled ($GPIOn_PDPU_SEL0[pin] = 0, GPIOn_PDPU_SEL1[pin] = 0$), pullup and pulldown disabled.
- Output mode disabled ($GPIOn_OUT_EN[pin] = 0$)
- Interrupt disabled ($GPIOn_INT_EN[pin] = 0$)

6.2.2 Input Mode Configuration

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

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

6.2.3 Output Mode Configuration

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

1. Set the GPIO Configuration Enable bits shown in Table 6-3 to any one of the I/O mode settings.
2. Configure the electrical characteristics of the pin as desired as shown in Table 6-5
3. Set the output high or low using the $GPIOn_OUT[pin]$ bit.
4. Enable the output buffer for the pin by setting $GPIOn_OUT_EN[pin]$ to 1.

6.2.4 Alternate Function Configuration

Most GPIO support one or more alternate functions which are selected with the GPIO Configuration Enable bits shown in Table 6-3. 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 6-3 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 6-4 if the assigned alternate function will use the pin as an input. See Table 6-5 if the assigned alternate function will use the pin as an output.
3. Set the GPIO Configuration Enable bits shown in Table 6-3 to the desired alternate function.

6.3 Configuring GPIO (External) Interrupts

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

GPIO interrupts can be individually enabled and configured as an edge or level triggered independently on a pin-by-pin basis. The edge trigger can be a rising, falling, or both transitions.
Each GPIO pin has a dedicated status bit in its corresponding \textit{GPIO\textsubscript{n}\_INT\_STAT} register. A GPIO interrupt will occur when the status bit transitions from 0 to 1 if the corresponding bit is set in the corresponding \textit{GPIO\textsubscript{n}\_INT\_EN} register. Note that the interrupt status bit will always be set when the current interrupt configuration event occurs, but an interrupt will only be generated if explicitly enabled.

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

1. Disable interrupts by setting the \textit{GPIO\textsubscript{n}\_INT\_EN}[\textit{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 \textit{GPIO\textsubscript{n}\_IN\_EN} register. To maintain previously enabled interrupts, read the \textit{GPIO\textsubscript{n}\_IN\_EN} register and save the state prior to setting the register to 0.

2. Clear pending interrupts by writing 1 to the \textit{GPIO\textsubscript{n}\_INT\_CLR}[\textit{pin}] bit.

3. Configure the pin for the desired interrupt event

4. Set \textit{GPIO\textsubscript{n}\_INT\_MODE}[\textit{pin}] to select the desired interrupt.
   a. For level triggered interrupts, the interrupt triggers on an input high (\textit{GPIO\textsubscript{n}\_INT\_POL}[\textit{pin}] = 0) or input low level.
   b. For edge triggered interrupts, the interrupt triggers on a transition from low to high(\textit{GPIO\textsubscript{n}\_INT\_POL}[\textit{pin}] = 0) or high to low (\textit{GPIO\textsubscript{n}\_INT\_POL}[\textit{pin}] = 1).

5. Optionally set \textit{GPIO\textsubscript{n}\_INT\_DUAL\_EDGE}[\textit{pin}] to 1 to trigger on both the rising and falling edges of the input signal.

6. Set \textit{GPIO\textsubscript{n}\_INT\_EN}[\textit{pin}] to 1 to enable the interrupt for the pin.

### 6.3.1 GPIO Interrupt Handling

Each GPIO port is assigned its own dedicated interrupt vector as shown in Table 6-6: MAX32665—MAX32668 GPIO Port Interrupt Vector Mapping.

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

1. Read the \textit{GPIO\textsubscript{n}\_INT\_STAT} register to determine the GPIO pin that triggered the interrupt.
2. Complete interrupt tasks associated with the interrupt source pin (application defined).
3. Clear the interrupt flag in the \textit{GPIO\textsubscript{n}\_INT\_STAT} register by writing a 1 to the \textit{GPIO\textsubscript{n}\_INT\_CLR} bit position that triggered the interrupt. This also clears and rearms the edge detectors for edge triggered interrupts.
4. Signal an end-of-interrupt to the interrupt controller by writing to the End-of-Interrupt register.
5. Return from the interrupt vector handler.

### 6.3.2 Using GPIO for Wakeup from Low Power Modes

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

A single wakeup interrupt vector, GPIOWAKE\_IRQHandler, is assigned for all pins of all GPIO ports. When the GPIO wakeup event occurs, the application software must interrogate each \textit{GPIO\textsubscript{n}\_INT\_STAT} register to determine which external port pin caused the wake-up event.

\textit{Table 6-7: MAX32665—MAX32668 GPIO Wakeup Interrupt Vector}
To enable low power mode wakeup (SLEEP, DEEPSLEEP and BACKUP) using an external GPIO interrupt, complete the following steps:

1. Clear pending interrupt flags by writing to \texttt{GPIOn\_INT\_CLR[\textit{pin}].}
2. Activate the GPIO wakeup function by writing 1 to \texttt{GPIOn\_WAKE\_EN[\textit{pin}].}
3. Configure the power manager to use the GPIO as a wakeup source by setting the appropriate \texttt{gpiowken} field to 1.

### 6.4 Registers

Each GPIO port has a unique base address as shown in \textit{Table 3-1: APB Register Base Address Map}. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific reset, if applicable.

\textbf{Table 6-8: GPIO Register Summary}

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>\texttt{GPIOn_EN0}</td>
<td>GPIO Port n Configuration Enable Bit 0 Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>\texttt{GPIOn_EN0_SET}</td>
<td>Atomic set for \texttt{GPIOn_EN0} register</td>
</tr>
<tr>
<td>[0x0008]</td>
<td>\texttt{GPIOn_EN0_CLR}</td>
<td>Atomic clear for \texttt{GPIOn_EN0} register</td>
</tr>
<tr>
<td>[0x000C]</td>
<td>\texttt{GPIOn_OUT_EN}</td>
<td>GPIO Port n Output enable register</td>
</tr>
<tr>
<td>[0x0010]</td>
<td>\texttt{GPIOn_OUT_EN_SET}</td>
<td>Atomic set for \texttt{GPIOn_OUT_EN} register</td>
</tr>
<tr>
<td>[0x0014]</td>
<td>\texttt{GPIOn_OUT_EN_CLR}</td>
<td>Atomic clear for \texttt{GPIOn_OUT_EN} register</td>
</tr>
<tr>
<td>[0x0018]</td>
<td>\texttt{GPIOn_OUT}</td>
<td>GPIO Port n Output register</td>
</tr>
<tr>
<td>[0x001C]</td>
<td>\texttt{GPIOn_OUT_SET}</td>
<td>Atomic set for \texttt{GPIOn_OUT} register</td>
</tr>
<tr>
<td>[0x0020]</td>
<td>\texttt{GPIOn_OUT_CLR}</td>
<td>Atomic clear for \texttt{GPIOn_OUT} register</td>
</tr>
<tr>
<td>[0x0024]</td>
<td>\texttt{GPIOn_IN}</td>
<td>GPIO Port n Input register</td>
</tr>
<tr>
<td>[0x0028]</td>
<td>\texttt{GPIOn_INT_MODE}</td>
<td>GPIO Port n Interrupt mode register</td>
</tr>
<tr>
<td>[0x002C]</td>
<td>\texttt{GPIOn_INT_POL}</td>
<td>GPIO Port n Interrupt polarity register</td>
</tr>
<tr>
<td>[0x0030]</td>
<td>\texttt{GPIOn_IN_EN}</td>
<td>GPIO Port n Input enable register</td>
</tr>
<tr>
<td>[0x0034]</td>
<td>\texttt{GPIOn_INT_EN}</td>
<td>GPIO Port n Interrupt enable register</td>
</tr>
<tr>
<td>[0x0038]</td>
<td>\texttt{GPIOn_INT_EN_SET}</td>
<td>Atomic set for \texttt{GPIOn_INT_EN} register</td>
</tr>
<tr>
<td>[0x003C]</td>
<td>\texttt{GPIOn_INT_EN_CLR}</td>
<td>Atomic clear for \texttt{GPIOn_INT_EN} register</td>
</tr>
<tr>
<td>[0x0040]</td>
<td>\texttt{GPIOn_INT_STAT}</td>
<td>GPIO Port n Interrupt status register</td>
</tr>
<tr>
<td>[0x0048]</td>
<td>\texttt{GPIOn_INT_CLR}</td>
<td>Atomic clear for \texttt{GPIOn_INT_STAT} register</td>
</tr>
<tr>
<td>[0x004C]</td>
<td>\texttt{GPIOn_WAKE_EN}</td>
<td>GPIO Port n Wake from DEEPSLEEP enable register</td>
</tr>
<tr>
<td>[0x0050]</td>
<td>\texttt{GPIOn_WAKE_EN_SET}</td>
<td>Atomic set for \texttt{GPIOn_WAKE_EN} register</td>
</tr>
<tr>
<td>[0x0054]</td>
<td>\texttt{GPIOn_WAKE_EN_CLR}</td>
<td>Atomic clear for \texttt{GPIOn_WAKE_EN} register</td>
</tr>
<tr>
<td>[0x0058]</td>
<td>\texttt{GPIOn_INT_DUAL_EDGE}</td>
<td>GPIO Port n Interrupt dual edge register</td>
</tr>
<tr>
<td>[0x0060]</td>
<td>\texttt{GPIOn_PDPU_SEL0}</td>
<td>GPIO Port n Input mode selection register 0</td>
</tr>
<tr>
<td>[0x0064]</td>
<td>\texttt{GPIOn_PDPU_SEL1}</td>
<td>GPIO Port n Input mode selection register 1</td>
</tr>
<tr>
<td>[0x0068]</td>
<td>\texttt{GPIOn_EN1}</td>
<td>GPIO Port n Configuration Enable Bit 1 Register</td>
</tr>
<tr>
<td>[0x006C]</td>
<td>\texttt{GPIOn_EN1_SET}</td>
<td>Atomic Set for \texttt{GPIOn_EN1} register</td>
</tr>
<tr>
<td>[0x0070]</td>
<td>\texttt{GPIOn_EN1_CLR}</td>
<td>Atomic Clear for \texttt{GPIOn_EN1} register</td>
</tr>
<tr>
<td>[0x0074]</td>
<td>\texttt{GPIOn_EN2}</td>
<td>GPIO Port n Configuration Enable Bit 2 Register</td>
</tr>
<tr>
<td>[0x0078]</td>
<td>\texttt{GPIOn_EN2_SET}</td>
<td>Atomic Set for \texttt{GPIOn_EN2} register</td>
</tr>
<tr>
<td>[0x007C]</td>
<td>\texttt{GPIOn_EN2_CLR}</td>
<td>Atomic Clear for \texttt{GPIOn_EN0} register</td>
</tr>
<tr>
<td>[0x0080]</td>
<td>\texttt{GPIOn_DS_SEL0}</td>
<td>GPIO Port n Output Drive strength selection register 0</td>
</tr>
</tbody>
</table>
### 6.5 Register Details

**Table 6-9: GPIO Port n Configuration Enable Bit 0 Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
<td>1</td>
<td>GPIO Configuration Enable, Bit 0</td>
</tr>
</tbody>
</table>

This bit, in conjunction with bits in Table 6-3: MAX32665—MAX32668 GPIO Pin Configuration, configures the corresponding device pin for digital I/O or an alternate function modes. This field can be modified directly by writing to this register or indirectly through GPIOn_EN0_SET or GPIOn_EN0_CLR.

Some GPIO are not implemented all devices. The bits associated with unimplemented GPIO should not be changed from their default value.

This bit’s setting does not affect input and interrupt functionality of the associated pin.

**Table 6-10: GPIO Port n Configuration Enable Atomic Set Bit 0 Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>GPIO Configuration Enable Atomic Set, Bit 0</td>
</tr>
</tbody>
</table>

Writing 1 to one or more bits sets the corresponding bits in the GPIOn_EN0 register.

0: No effect.
1: Corresponding bits in GPIOn_EN0 register set to 1.

**Table 6-11: GPIO Port n Configuration Enable Atomic Clear Bit 0 Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>-</td>
<td>WO</td>
<td>0</td>
<td>GPIO Configuration Enable Atomic Clear, Bit 0</td>
</tr>
</tbody>
</table>

Writing 1 to one or more bits clears the corresponding bits in the GPIOn_EN0 register.

0: No effect.
1: Corresponding bits in GPIOn_EN0 register cleared to 0.

**Table 6-12: GPIO Port n Output Enable Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>GPIO Output Enable</td>
</tr>
</tbody>
</table>

Set bit to 1 to enable the output driver for the corresponding GPIO pin. A bit can be enabled directly by writing to this register or indirectly through GPIOn_OUT_EN_SET or GPIOn_OUT_EN_CLR.

0: Pin is set to input mode; output driver disabled.
1: Pin is set to output mode.
Table 6-13: GPIO Port n Output Enable Atomic Set Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0 | -     | WO     | 0     | GPIO Output Enable Atomic Set  
Writing 1 to one or more bits sets the corresponding bits in \( \text{GPIO}_n\_\text{OUT\_EN} \).
0: No effect.
1: Corresponding bits in \( \text{GPIO}_n\_\text{OUT\_EN} \) set to 1. |

Table 6-14: GPIO Port n Output Enable Atomic Clear Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0 | -     | WO     | 0     | GPIO Output Enable Atomic Clear  
Writing 1 to one or more bits sets the corresponding bits in \( \text{GPIO}_n\_\text{OUT\_EN} \).
0: No effect.
1: Corresponding bits in \( \text{GPIO}_n\_\text{OUT\_EN} \) cleared to 0. |

Table 6-15: GPIO Port n Output Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0 | -     | R/W    | 0     | GPIO Output  
Set the corresponding output pin high or low.  
0: Drive the corresponding output pin low (logic 0).  
1: Drive the corresponding output pin high (logic 1). |

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

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0 | -     | WO     | 0     | GPIO Output Atomic Set  
Writing 1 to one or more bits in this register sets the corresponding bits in the \( \text{GPIO}_n\_\text{OUT} \) register.  
Writing 1 to one or more bits sets the corresponding bits in \( \text{GPIO}_n\_\text{OUT} \).
0: No effect.
1: Corresponding bits in \( \text{GPIO}_n\_\text{OUT\_EN} \) set to 1. |

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

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0 | -     | WO     | 0     | GPIO Output Atomic Clear  
Writing 1 to one or more bits in this register clears the corresponding bits in the \( \text{GPIO}_n\_\text{OUT} \) register.  
0: No effect.
1: Corresponding bits in \( \text{GPIO}_n\_\text{OUT\_EN} \) cleared to 0. |
Table 6-18: GPIO Port n Input Register

<table>
<thead>
<tr>
<th>GPIO Port n Input</th>
<th>GPIOn_IN</th>
<th>[0x0024]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6-19: GPIO Port n Interrupt Mode Register

<table>
<thead>
<tr>
<th>GPIO Port n Interrupt Mode</th>
<th>GPIOn_INT_MODE</th>
<th>[0x0028]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6-20: GPIO Port n Interrupt Polarity Register

<table>
<thead>
<tr>
<th>GPIO Port n Interrupt Polarity</th>
<th>GPIOn_INT_POL</th>
<th>[0x002C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6-21: GPIO Port n Input Enable Register

<table>
<thead>
<tr>
<th>GPIO Port n Input Enable</th>
<th>GPIOn_IN_EN</th>
<th>[0x0030]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 6-22: GPIO Port n Interrupt Enable Register

<table>
<thead>
<tr>
<th>GPIO Port n Interrupt Enable</th>
<th>GPIOn_INT_EN</th>
<th>[0x0034]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6-23: GPIO Port n Interrupt Enable Atomic Set Register

<table>
<thead>
<tr>
<th>GPIO Port Interrupt Enable Atomic Set</th>
<th>GPIOn_INT_EN_SET</th>
<th>[0x0038]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>-</td>
<td>WO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6-24: GPIO Port n Interrupt Enable Atomic Clear Register

<table>
<thead>
<tr>
<th>GPIO Port Interrupt Enable Atomic Clear</th>
<th>GPIOn_INT_EN_CLR</th>
<th>[0x003C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>-</td>
<td>WO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6-25: GPIO Port n Interrupt Status Register

<table>
<thead>
<tr>
<th>GPIO Port Interrupt Status</th>
<th>GPIOn_INT_STAT</th>
<th>[0x0040]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6-26: GPIO Port n Interrupt Clear Register

<table>
<thead>
<tr>
<th>GPIO Port Interrupt Clear</th>
<th>GPIOn_INT_CLR</th>
<th>[0x0048]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 6-27: GPIO Port n Wakeup Enable Register

<table>
<thead>
<tr>
<th>GPIO Port Wakeup Enable</th>
<th>GPIOn_WAKE_EN</th>
<th>0x004C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6-28: GPIO Port n Wakeup Enable Atomic Set Register

<table>
<thead>
<tr>
<th>GPIO Port Wakeup Enable Atomic Set</th>
<th>GPIOn_WAKE_EN_SET</th>
<th>0x0050</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6-29: GPIO Port n Wakeup Enable Clear Register

<table>
<thead>
<tr>
<th>GPIO Port Wakeup Enable Atomic Clear</th>
<th>GPIOn_WAKE_EN_CLR</th>
<th>0x0054</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6-30: GPIO Port n Interrupt Dual Edge Mode Register

<table>
<thead>
<tr>
<th>GPIO Port n Pullup Pulldown Selection 0</th>
<th>GPIOn_INT_DUAL_EDGE</th>
<th>0x005C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6-31: GPIO Port n Pullup Pulldown Selection 0 Register

<table>
<thead>
<tr>
<th>GPIO Port n Pullup Pulldown Selection 0</th>
<th>GPIOn_PDPU_SEL0</th>
<th>0x0060</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 6-32: GPIO Port n Pullup Pulldown Selection 1 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>GPIO Pullup Pulldown Selection 1&lt;br&gt;Input mode configuration for the associated GPIO pin. Input mode selection and the selection of a weak or strong pullup or weak or strong pulldown resistor are described in Table 6-4..</td>
</tr>
</tbody>
</table>

### Table 6-33: GPIO Port n Configuration Enable Bit 1 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>GPIO Configuration Enable, Bit 1&lt;br&gt;This bit, in conjunction with bits in Table 6-3: MAX32665—MAX32668 GPIO Pin Configuration, configures the corresponding device pin as a GPIO or an alternate function mode.&lt;br&gt;Some GPIO are not implemented on all devices. The bits associated with unimplemented GPIO should not be changed from their default value. See Table 6-1: MAX32665—MAX32668 GPIO Pin Count concerning which pins are available.&lt;br&gt;This bit's setting does not affect input and interrupt functionality of the associated pin.</td>
</tr>
</tbody>
</table>

### Table 6-34: GPIO Port n Configuration Enable Atomic Set, Bit 1 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>GPIO Configuration Enable Atomic Set, Bit 1&lt;br&gt;Writing 1 to one or more bits sets the corresponding bits in the GPIOn_EN1 register.&lt;br&gt;0: No effect.&lt;br&gt;1: Corresponding bits in GPIOn_EN1 register set to 1.</td>
</tr>
</tbody>
</table>

### Table 6-35: GPIO Port n Configuration Enable Atomic Clear, Bit 1 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>GPIO Configuration Enable Atomic Clear, Bit 1&lt;br&gt;Writing 1 to one or more bits clears the corresponding bits in the GPIOn_EN1 register.&lt;br&gt;0: No effect.&lt;br&gt;1: Corresponding bits in GPIOn_EN1 register cleared to 0.</td>
</tr>
</tbody>
</table>
### Table 6-36: GPIO Port n Configuration Enable Bit 2 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>GPIO Configuration Enable, Bit 2</td>
</tr>
</tbody>
</table>

This bit, in conjunction with bits in Table 6-3: MAX32665—MAX32668 GPIO Pin Configuration, configures the corresponding device pin as a GPIO or an alternate function mode.

Some GPIO are not implemented all devices. The bits associated with unimplemented GPIO should not be changed from their default value. See Table 6-1: MAX32665—MAX32668 GPIO Pin Count concerning which pins are available.

This bit’s setting does not affect input and interrupt functionality of the associated pin.

### Table 6-37: GPIO Port n Configuration Enable Atomic Set Bit 2 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>GPIO Alternate Function Select Atomic Set, Bit 2</td>
</tr>
</tbody>
</table>

Writing 1 to one or more bits sets the corresponding bits in the GPIO\_EN2 register.

- 0: No effect.
- 1: Corresponding bits in GPIO\_EN2 register set to 1.

### Table 6-38: GPIO Port n Configuration Enable Atomic Clear Bit 2 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>GPIO Alternate Function Select Atomic Clear, Bit 2</td>
</tr>
</tbody>
</table>

Writing 1 to one or more bits clears the corresponding bits in the GPIO\_EN2 register.

- 0: No effect.
- 1: Corresponding bits in GPIO\_EN2 register cleared to 0.

### Table 6-39: GPIO Port n Output Drive Strength Bit 0 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>GPIO Output Drive Strength Selection 0</td>
</tr>
</tbody>
</table>

See Table 6-5: MAX32665—MAX32668 Output Mode Configuration for details on how to set the GPIO output drive strength and other electrical characteristics.

### Table 6-40: GPIO Port n Output Drive Strength Bit 0 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>GPIO Output Drive Strength Selection 1</td>
</tr>
</tbody>
</table>

See Table 6-5: MAX32665—MAX32668 Output Mode Configuration for details on how to set the GPIO output drive strength and other electrical characteristics.
### Table 6-41: GPIO Port Pulldown/Pullup Strength Select Register

<table>
<thead>
<tr>
<th>GPIO Port Pulldown/Pullup Strength Select</th>
<th>GPIOn_PS</th>
<th>[0x00B8]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 6-42: GPIO Port Supply Voltage Select Register

<table>
<thead>
<tr>
<th>GPIO Port Supply Voltage Select</th>
<th>GPIOn_VSSEL</th>
<th>[0x00C0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
7. Flash Controller (FLC)

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

- Up to 1 MB total internal flash memory
- 128 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

7.1 Instances

The device provides two instances of the FLC.

The 1MB of internal flash memory is organized as two distinct instances of 512KB, each with its own dedicated controller, for storing user application and data. These internal flash memory instances are programmable via serial wire debug interface (in-system) or directly with user application code (in-application).

The flash instances are organized as an array of pages. Each page is 2,048 words by 128 bits, or 8,192 bytes per page. Table 7-1, below, shows the start address and end address for each flash instance. The internal flash memory is mapped with a start address of 0x1000 0000 and an end address of 0x100F FFFF for a total of 1MB.

Table 7-1: MAX32665—MAX32668 Internal Flash Memory Organization

<table>
<thead>
<tr>
<th>Instance Number</th>
<th>Page Number</th>
<th>Size</th>
<th>Start Address</th>
<th>End Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLC0</td>
<td>1</td>
<td>8,192 Bytes</td>
<td>0x1000 0000</td>
<td>0x1000 1FFF</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>8,192 Bytes</td>
<td>0x1000 2000</td>
<td>0x1000 3FFF</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>8,192 Bytes</td>
<td>0x1000 4000</td>
<td>0x1000 5FFF</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>8,192 Bytes</td>
<td>0x1000 6000</td>
<td>0x1000 7FFF</td>
</tr>
<tr>
<td></td>
<td>63</td>
<td>-</td>
<td>0x1007 C000</td>
<td>0x1007 DFFF</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>-</td>
<td>0x1007 E000</td>
<td>0x1007 FFFF</td>
</tr>
<tr>
<td>FLC1</td>
<td>1</td>
<td>-</td>
<td>0x1008 0000</td>
<td>0x1008 1FFF</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>-</td>
<td>0x1008 2000</td>
<td>0x1008 3FFF</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>-</td>
<td>0x1008 4000</td>
<td>0x1008 5FFF</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>-</td>
<td>0x1008 6000</td>
<td>0x1008 7FFF</td>
</tr>
<tr>
<td></td>
<td>63</td>
<td>8,192 Bytes</td>
<td>0x100F C000</td>
<td>0x100F DFFF</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>8,192 Bytes</td>
<td>0x100F E000</td>
<td>0x100F FFFF</td>
</tr>
</tbody>
</table>

7.2 Usage

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

The Flash Controller requires a 1MHz peripheral clock for operation. The input clock to the Flash Controller block is the system clock, \( f_{SYSCLK} \). Use the Flash Controller clock divisor to generate \( f_{FLCnCLK} = 1MHz \), as shown in Equation 7-1 below.

For the 96MHz Oscillator as the system clock, the \( FLCn_CLKDIV.clkdiv \) should be set to 96 (0x60).

\[
f_{FLCnCLK} = \frac{f_{SYSCLK}}{FLCn_CLKDIV.clkdiv} = 1MHz
\]

7.2.2 Lock Protection

A locking mechanism prevents accidental memory writes and erases. All writes and erase operations require the \( FLCn_CTRL.unlock \) field be set to 0x2 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 becoming locked from the unlocked state.

Note: If a write, page erase or mass erase operation is started and the unlock code was not set to 0x2, the flash controller hardware sets the access fail flag, \( FLCn_INTR.access_fail \), to indicate an access violation occurred.

7.2.3 Flash Write Width

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

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>128-bit Write</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Table 7-2: Valid Addresses Flash Writes
7.2.4  **Flash Write**

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

1. If desired, enable flash controller interrupts by setting the FLCn_INTR.access_error_ie and FLCn_INTR.done_ie bits.
2. Read the FLCn_CTRL.busy bit until it returns 0.
3. Configure FLCn_CTRL.clkdiv to match the SYS_CLK frequency.
4. Set the FLCn_ADDR register to a valid target address. Reference Table 7-2.
5. Set FLCn_DATA3, FLCn_DATA2, FLCn_DATA1, and FLCn_DATA0 to the data to write. FLCn_DATA3 is the most significant word and FLCn_DATA0 is the least significant word. Each word of the data to write follows the little-endian format where the least significant byte of the word is stored at the lowest-numbered byte and the most significant byte is stored at the highest-numbered byte.
6. Set FLCn_CTRL.unlock to 0x2 to unlock the flash instance.
7. Set FLCn_CTRL.write to 1. This field is automatically cleared by the Flash Controller when the write operation is finished.
8. FLCn_INTR.done is set by hardware when the write completes and if an error occurred, the FLCn_INTR.access_fail flag is set. These bits generate a flash IRQ if the interrupt enable bits are set.
9. Set FLCn_CTRL.unlock to any value other than 0x2 to re-lock the flash instance.

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

7.2.5  **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.access_error_ie and FLCn_INTR.done_ie bits.
2. Read the FLCn_CTRL.busy 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 Flash Controller to ensure the address is page aligned.
5. Set FLCn_CTRL.unlock to 0x2 to unlock the flash instance.
6. Set FLCn_CTRL.erase_code to 0x55 for page erase.
7. Set FLCn_CTRL.page_erase to 1 to start the page erase operation.
8. The FLCn_CTRL.busy bit is set by the flash controller while the page erase is in progress and the FLCn_CTRL.page_erase and FLCn_CTRL.busy 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.access_fail 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 0x2 to re-lock the flash instance.
7.2.6 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.busy` bit until it returns 0.
2. Configure `FLCn_CLKDIV.clkdiv` to match the SYS_CLK frequency.
3. Set `FLCn_CTRL.unlock` to 0x2 to unlock the internal flash.
4. Set `FLCn_CTRL.erase_code` to 0xAA for mass erase.
5. Set `FLCn_CTRL.mass_erase` to 1 to start the mass erase operation.
6. The `FLCn_CTRL.busy` bit is set by the flash controller while the mass erase is in progress and the `FLCn_CTRL.mass_erase` and `FLCn_CTRL.busy` 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.access_fail` 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 0x2 to re-lock the flash instance.

7.3 Flash Error Correction Coding

The Flash Controller ECC data register `FLCn_ECC_Data` stores the ECC bits from the last flash instance read memory location. The register contains 9 bits of ECC data of the even 128-bit flash memory location `FLCn_ECC_Data.ecc_even` and 9 bits of ECC data of the 128-bit odd flash memory location `FLCn_ECC_Data.ecc_odd`. These 9-bit ECC data fields are dynamic and are valid only immediately after each location read and represent the ECC for 256 bits of flash. The 128-bit even location of this even/odd pair is matched with the 128-bit odd location of the lower-valued memory address. In case of ECC error from internal flash memory read cycles, the `FLCn_ECC_Data` can be used in conjunction with the Table 4-70: Error Correction Coding Interrupt Enable Register to debug the ECC failure.

7.4 Flash Controller Registers

See Table 3-1: APB Peripheral Base Address Map for the Flash Controller 0 and Flash Controller 1 Peripheral Base Addresses.

Table 7-3: Flash Controller Registers

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register Name</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td><code>FLCn_ADDR</code></td>
<td>R/W</td>
<td>Flash Controller Address Pointer Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td><code>FLCn_CLKDIV</code></td>
<td>R/W</td>
<td>Flash Controller Clock Divisor Register</td>
</tr>
<tr>
<td>[0x0008]</td>
<td><code>FLCn_CTRL</code></td>
<td>R/W</td>
<td>Flash Controller Control Register</td>
</tr>
<tr>
<td>[0x0024]</td>
<td><code>FLCn_INTR</code></td>
<td>R/W</td>
<td>Flash Controller Interrupt Register</td>
</tr>
<tr>
<td>[0x0028]</td>
<td><code>FLCn_ECC_Data</code></td>
<td>R/W</td>
<td>Flash Controller Error Correction Code Data</td>
</tr>
<tr>
<td>[0x0030]</td>
<td><code>FLCn_DATA0</code></td>
<td>R/W</td>
<td>Flash Controller Data Register 0</td>
</tr>
<tr>
<td>[0x0034]</td>
<td><code>FLCn_DATA1</code></td>
<td>R/W</td>
<td>Flash Controller Data Register 1</td>
</tr>
<tr>
<td>[0x0038]</td>
<td><code>FLCn_DATA2</code></td>
<td>R/W</td>
<td>Flash Controller Data Register 2</td>
</tr>
<tr>
<td>[0x003C]</td>
<td><code>FLCn_DATA3</code></td>
<td>R/W</td>
<td>Flash Controller Data Register 3</td>
</tr>
</tbody>
</table>
## 7.5 Flash Controller Register Details

### Table 7-4: Flash Controller Address Pointer Register

<table>
<thead>
<tr>
<th>Flash Address Register</th>
<th>FLCn_ADDR</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>addr</td>
<td>R/W</td>
</tr>
</tbody>
</table>

Flash Address
This field contains the target address for a write operation. A valid internal flash memory address is required for all write operations. The reset value for this field is always 0x0010 0000.

### Table 7-5: Flash Controller Clock Divisor Register

<table>
<thead>
<tr>
<th>Flash Controller Clock Divisor Register</th>
<th>FLCn_CLKDIV</th>
<th>[0x0004]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 31:8 | - | RO | - | Reserved for Future Use  
Do not modify this field. |
| 7:0 | clkdiv | R/W | 0x60 | Flash Controller Clock Divisor  
The system clock, SYS_CLK, is divided by the value in this field to generate the FLCn peripheral clock, \( f_{FLCnCLK} \). The FLCn peripheral clock must equal 1MHz. The default on all forms of reset is 96 (0x60), resulting in \( f_{FLCnCLK} = 1 \text{MHz} \). The FLCn peripheral clock is only used during erase and program functions and not during read functions. |

### Table 7-6: Flash Controller Control Register

<table>
<thead>
<tr>
<th>Flash Controller Control Register</th>
<th>FLCn_CTRL</th>
<th>[0x0008]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 31:28 | unlock_code | R/W | 0 | Flash Unlock  
Write the unlock code, 0x2, prior to any flash write or erase operation to unlock the Flash. Writing any other value to this field locks the internal flash.  
0x2: Flash unlock code |
| 27:25 | - | RO | - | Reserved for Future Use  
Do not modify this field. |
| 24 | busy | RO | 0 | Flash Busy Flag  
When this field is set, writes to all flash registers except the FLCn_INTR register are ignored by the Flash Controller.  
Note: If the Flash Controller is busy (FLCn_CTRL.busy = 1), reads, writes and erase operations are not allowed and result in an access failure (FLCn_CTRL.access_fail = 1).  
0: Flash idle  
1: Flash busy |
| 23:16 | - | RO | - | Reserved for Future Use  
Do not modify this field. |
| 15:8 | erase_code | R/W | 0 | Erase Code  
Prior to an erase operation this field must be set to 0x55 for a page erase or 0xAA for a mass erase. The flash must be unlocked prior to setting the erase code.  
This field is automatically cleared after the erase operation is complete.  
0x00: Erase disabled.  
0x55: Page erase code.  
0xAA: Enable mass erase. |
# Flash Controller Control Register

**FLCn_CTRL**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:3</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved for Future Use. Do not modify this field.</td>
</tr>
<tr>
<td>2</td>
<td>page_erase</td>
<td>R/W1O</td>
<td>0</td>
<td>Page Erase</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write a 1 to this field to initiate a page erase</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>at the address in <strong>FLCn_ADDR.addr</strong>. The flash</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>must be unlocked prior to attempting a page</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>erase, see <strong>FLCn_CTRL.unlock</strong> for details.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The Flash Controller hardware clears this bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>when a page erase operation is complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No page erase operation in process or page</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>erase is complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Write a 1 to initiate a page erase. If this</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>field reads 1, a page erase operation is in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>progress.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> This field is protected and cannot be</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>set to 0 by application code.</td>
</tr>
<tr>
<td>1</td>
<td>mass_erase</td>
<td>R/W1O</td>
<td>0</td>
<td>Mass Erase</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write a 1 to this field to initiate a mass erase</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>of the internal flash memory. The flash must be</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>unlocked prior to attempting a mass erase, see</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>FLCn_CTRL.unlock</strong> for details.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The Flash Controller hardware clears this bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>when the mass erase operation completes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Initiate mass erase</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> This field is protected and cannot be</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>set to 0 by application code.</td>
</tr>
<tr>
<td>0</td>
<td>write</td>
<td>R/W1O</td>
<td>0</td>
<td>Write</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If this field reads 0, no write operation is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>pending for the flash. To initiate a write</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>operation, set this bit to 1 and the Flash</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Controller will write to the address set in the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>FLCn_ADDR</strong> register.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No write operation in process or write</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>operation complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Write 1 to initiate a write operation. If this</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>field reads 1, a write operation is in progress.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> This field is protected and cannot be</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>set to 0 by application code.</td>
</tr>
</tbody>
</table>

**Table 7-7: Flash Controller Interrupt Register**

**FLCn_INTR**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:10</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved for Future Use. Do not modify this field.</td>
</tr>
<tr>
<td>9</td>
<td>access_fail_ie</td>
<td>R/W</td>
<td>0</td>
<td>Flash Access Fail Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set this bit to 1 to enable interrupts on flash</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>access failures.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>8</td>
<td>done_ie</td>
<td>R/W</td>
<td>0</td>
<td>Flash Operation Complete Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set this bit to 1 to enable interrupts on flash</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>operations complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>7:2</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved for Future Use. Do not modify this field.</td>
</tr>
</tbody>
</table>
### Flash Controller Interrupt Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>access_fail</td>
<td>R/WOC</td>
<td>0</td>
<td><strong>Flash Access Fail Interrupt Flag</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit is set when an attempt is made to write or erase the flash while</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the flash is busy or locked. Only hardware can set this bit to 1. Writing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>a 1 to this bit has no effect. This bit is cleared by writing a 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No access failure has occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Access failure occurred.</td>
</tr>
</tbody>
</table>

| 0    | done            | R/WOC    | 0     | **Flash Operation Complete Interrupt Flag**                                 |
|      |                 |          |       | This flag is automatically set by hardware after a flash write or erase     |
|      |                 |          |       | operation completes.                                                        |
|      |                 |          |       | 0: Operation not complete or not in process.                                 |
|      |                 |          |       | 1: Flash operation complete.                                                |

### Table 7-8: Flash Controller ECC Data Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:25</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td><strong>Reserved for Future Use</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>24:16</td>
<td>ecc_odd</td>
<td>RO</td>
<td>0</td>
<td><strong>Error Correction Code Odd Data</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9-bit ECC data recorded from the last flash read memory location of odd</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>address of the even/odd pair of 128-bit flash memory content.</td>
</tr>
<tr>
<td>15:9</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td><strong>Reserved for Future Use</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>8:0</td>
<td>ecc_even</td>
<td>RO</td>
<td>0</td>
<td><strong>Error Correction Code Even Data</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9-bit ECC data recorded from the last flash read memory location of even</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>address of the even/odd pair of 128-bit flash memory content.</td>
</tr>
</tbody>
</table>

### Table 7-9: Flash Controller Data Register 0

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>data0</td>
<td>R/W</td>
<td>0</td>
<td><strong>Flash Data 0</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Flash data for bits 31:0.</td>
</tr>
</tbody>
</table>

### Table 7-10: Flash Controller Data Register 1

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>data1</td>
<td>R/W</td>
<td>0</td>
<td><strong>Flash Data 1</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Flash data for bits 63:32</td>
</tr>
</tbody>
</table>

### Table 7-11: Flash Controller Data Register 2

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>data2</td>
<td>R/W</td>
<td>0</td>
<td><strong>Flash Data 2</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Flash data for bits 95:64</td>
</tr>
</tbody>
</table>
### Table 7-12: Flash Controller Data Register 3

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0 | data3 | R/W    | 0     | Flash Data 3  
Flash data for bits 127-96. |
8. External Memory

8.1 Overview

External memory can be accessed via multiple interfaces. There are three external memory interfaces, two of which are backed by 16KB of cache:

- SPI Execute-in-Place FLASH (SPIXF)
  - 16KB dedicated cache
- SPI Execute-in-Place RAM (SPIXR)
  - 16KB dedicated cache
- SD/SDIO/SDHC/MMC

8.2 SPI Execute-in-Place Flash (SPIXF)

The SPIXF provides the following features:

- Up to 48MHz operation in mode 0 and 3
- Single slave select
- Four wire mode for single-bit slave device communication
- Dual and Quad I/O supported
- Programmable SCK frequency and duty cycle
- SS assertion and de-assertion timing with respect to the leading and trailing SCK edge
- Configurable command, address, dummy, and data fields to support a variety of SPI flashes

The SPIXF allows the CPU to transparently execute instructions stored in an external SPI flash. Instructions fetched using the SPIXF are cached just like instructions fetched from internal program memory. You can also use the SPIXF to access large amounts of external static data that would otherwise reside in internal data memory. This device provides support for a wide variety of external SPI flash memory devices.

Prior to using the SPI flash device, you must configure the SPIXF interface.

To prevent disclosure of intellectual property, code and data can optionally be stored in external flash in an encrypted form using the SPIXF. Generation of the encrypted data can be done via user firmware or with the cryptographic accelerator. The SPIXF can transparently decrypt this information in real time using Memory Decryption Integrity Unit (MDOU) with the AES-128 algorithm in ECB mode.

The SPIXF consists of the SPIXF Master and SPIXF Master Controller as shown in the diagram below. The SPIXF Master transparently reads the external SPI flash device while the SPIXF Master Controller is used to manually write data to the external SPI flash and to configure the SPI external flash device registers.
8.2.1 SPIXF Master Controller

The SPIXF Master Controller block (SPIXFC) shown in Figure 8-2 consists of transmit and receive shift registers (supported by FIFOs) and a control unit. Communication and interface configuration are set up using the APB registers. It contains one 16×16 FIFO (Transmit FIFO) to support the transmit direction and one 32×8 FIFO (Receive FIFO) to support the receive direction. These FIFOs are accessible to firmware using an AHB interface to support high-speed data transfers. New data is moved automatically from the Transmit FIFO into the shift register at the start of every new SPI transfer as long as there is
data in the Transmit FIFO. At the end of every SPI transfer, data is moved from the shift register into the Receive FIFO. Status flags and interrupts are available to monitor the data levels in these FIFOs.

When a SPI transfer occurs, a multi-byte (selectable from 1 to 1024 bytes) packet is shifted out if the Transmit FIFO has configured the device to transmit using the Transmit FIFO header entry. The most significant bit is sent first. If the Transmit FIFO configures the device to receive, the device receives data most significant bit first, and places each byte received into the Receive FIFO.

*Figure 8-2. Simplified Block Diagram*

### 8.2.1.1 SPIXFC Configuration

The SPIXF Master Controller shares pins with the SPIXF Master so that the SPI flash is configured for code execution or data transfer. See the SPI Pins Configuration section of the SPIXF Master for more information.

#### 8.2.1.1.1 Configuration Modes Overview

Once the main SPIXF Master Controller clock is set up, the remainder of the configuration and operation for this block is mapped into three categories:

1. Static configuration: Performed during SPI initial setup, when the communicate port is disabled, or both.
   a. `SPIXFC_GEN_CTRL` register: SCK Feedback mode, enable Transmit and Receive FIFOs
   b. `SPIXFC_SS_POL` register: Slave Select signal polarity
   c. `SPIXFC_CFG` register: Active slave, SPI clock polarity and phase that is, mode), clock and slave select timing.

2. Dynamic configuration: Configuration required to communicate with a specific slave device, which may take place while the communication port is enabled but the slave select is de-asserted.
   a. `SPIXFC_CFG` register: SPI page transfer size if using pages. See header information in Table 8-1.

3. Interrupt servicing: Status and control used by an application to efficiently service the SPI data transfer.
   a. `SPIXFC_FIFO_CTRL` register: Transmit and Receive FIFO monitoring levels
   b. `SPIXFC_INT_FL` register: Interrupt flag bits
   c. `SPIXFC_INT_EN` register: Interrupt enable bits

#### 8.2.1.1.2 SPI Master Controller Transaction

Once the SPIXFC is configured to communicate to a specific slave, SPI transactions are initiated by writing to the SPI Transmit FIFO `SPIXFC_FIFO_TX`. The FIFO is 16-bits wide and expects a 16-bit header followed by an optional payload padded out to a word boundary.
The format of the header is shown in Table 8-1. If the transaction generates receive data, this data is pushed into the SPI. The Receive FIFO is SPIFC_FIFO_RX.

A complete access sequence to a SPI device is made up of one or more transactions. In some cases, the slave select signal remains asserted across several transactions. In other cases, the access sequence defined by the slave device might require de-assertion of the slave selection in the middle of the access sequence. In general, any part of the access sequence that requires a change in direction, width, or timing, requires another transaction. Interrupt logic is provided to allow efficient servicing of the SPI Master functionality by firmware.

**Table 8-1: SPI Header Format**

<table>
<thead>
<tr>
<th>Name</th>
<th>Bits</th>
<th>Description</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header Type</td>
<td>15:14</td>
<td><strong>Reserved for Future Use.</strong> This header field should always be set to 0b00.</td>
<td>0b00</td>
</tr>
</tbody>
</table>
| De-assert SS| 13     | Slave Select control.                                       | 0: Maintain assertion of slave select after transaction.  
1: De-assert slave select at the completion of transaction |
| RFU        | 12:11  | **Reserved for Future Use.** This header field should always be set to 0b00. |
| Width      | 10:9   | Number of SDIO pins to use for the transaction.             | 0x00: Single I/O mode  
0x01: Dual I/O mode  
0x02: Quad I/O mode  
0x03: Invalid |
| Size       | 8:4    | Size of transaction in terms of units.                      | 0x00: 32  
0x01: 1  
0x02: 2  
...  
0x0F: 15 |
| Size Units | 3:2    | Defines units to use when interpreting the size field. Bit transactions are available only for Tx (that is, Direction = 1 transactions). | 0: Bits  
1: Bytes  
2: Pages (See the SPIFC_CFG.pgsz field for page size definition) |
| Direction  | 1:0    | Defines direction of information transfer. For headers that do not define a transmission (that is, direction = None or Rx), no payload is required. Conversely, headers that do not define a reception (that is, direction = None or Tx), result in no data pushing into the Receive FIFO. | 0: None  
1: Tx  
2: Rx  
3: Both |
8.2.1.1.3 Sample SPIXF Master Controller Example

Here is an example how to set up the Master Controller:

1. Configure the SPIXF Master Controller mode, number of bytes per page (see SPIFC_CFG.pgsz and Note below), SCK high and low values, and Slave Select (SS) active timing and inactive timing.
   a. Example:
      i. SPIFC_CFG.mode = 0b00 SPI Mode 0
      ii. SPIFC_CFG.pgsz = 0b00 Page size = four bytes
      iii. SPIFC_CFG.loclk = 0b01 SCK low time = 1 system clock period
      iv. SPIFC_CFG.hiclk = 0b01 SCK high time = 1 system clock period
      v. SPIFC_CFG.ssact = 0b10 SS Active time = 4 system clock periods
      vi. SPIFC_CFG.inact = 0b10 SS Inactive stretch time = 8 system clock periods
2. Configure the Almost Empty and Almost Full levels for monitoring the FIFOs.
   a. Example:
      i. SPIFC_FIFO_CTRL.tfifolvl = 0x8 Almost Empty = 8
      ii. SPIFC_FIFO_CTRL.tfifolvl = 0xC Almost Full = 12
3. Enable the Transmit and Receive FIFOs as well as the feedback clock.
   i. SPIFC_GEN_CTRL.rfifoen = 1 Receive FIFO enabled
   ii. SPIFC_GEN_CTRL.tfifoen = 1 Transmit FIFO enabled
4. Write the header then payload data to the Transmit FIFO (SPIFC_FIFO_TX) to send a command to configure the SPI flash for configuration or programming. More than one command may be loaded to the FIFO.
5. Initialize the SPIXF Master Controller interrupt flags by clearing the SPIFC_INT_FL register.
6. Set the interrupt enables for Transmit FIFO stalled and almost empty to make sure that the Transmit FIFO does not stall the AHB bus.
7. Write to the SPIXF Master Controller enable bit to start the transmission (SPIFC_GEN_CTRL.enable = 1).
8. Monitor the receive stalled interrupt status flag to know when data is available in the Receive FIFO if data is received by this command.
9. Monitor the SPIFC_INT_FL.txrdy for indication that the command has completed.
10. Repeat steps 6 through 9 to monitor multiple commands to the SPI flash if necessary.

Note: Page size is used if enabled by the SPIXF Master Controller header to configure the SPIXF Master Controller for larger transaction packet sizes (not to be confused with the SPI flash page size).
Multiple headers and payloads are written to the Transmit FIFO for consecutive execution. As an example, complete the following steps to set up the external SPI flash bus width using the SPIXF Master Controller:

1. Configure the SPIXF Master Controller so it can communicate with the default configuration of the external SPI flash chosen using the appropriate register and header settings.
2. Write the header and initial payload to the Transmit FIFO to send configuration of the data width (single/dual/quad) to the external SPI flash. This might require multiple commands to write status registers of the external flash device or to send specific commands.
3. Write header and payload to the Transmit FIFO to complete subsequent commands (read/write external SPI flash registers and program external SPI flash) using the new external SPI flash IO configuration.
4. Enable the SPIXF Master Controller to send commands to the external SPI flash.

8.2.1.1.4 Clock Phase and Polarity Control

The SPIXF Master Controller and the SPIXF Master support configuration of SCK phase and polarity:

- Clock polarity (CLKPOL) selects an active low/high clock and has no effect on the transfer format
- Clock phase (PHASE) selects one of two different transfer formats

The master always places data on the MOSI line a half cycle before the SCK edge for the slave to latch the data.

Table 8-2 details the SCK phase and polarity combinations supported. See `SPIFC_CFG.mode` register.

<table>
<thead>
<tr>
<th>MODE</th>
<th>PHASE</th>
<th>CLKPOL</th>
<th>SCK Transmit Edge</th>
<th>SCK Receive Edge</th>
<th>SCK Idle State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Falling</td>
<td>Rising</td>
<td>Low</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>Falling</td>
<td>Rising</td>
<td>High</td>
</tr>
</tbody>
</table>

Note: Do not change the clock phase and polarity control while executing or reading from SPIXF space. This configuration should ideally be done prior to SPIXF transactions and remain unchanged while reading or executing from SPIXF space. If the clock phase and polarity need to be changed after the SPIXF slave select is active, the user must not be executing from SPIXF space, and the SPIXF block should be reset by setting `GCR_RST1.spixf` = 1.

8.2.1.1.5 Serial Clock Configuration

The output clock speed and pulse width can be controlled with the `SPIFC_CFG.hiclk` and `SPIFC_CFG.loclk` register fields.

**Target SPI Clock Hi Time** = \( t_{\text{PCLK}} \times \text{SPIFC_CFG.hiclk} \)

**Target SPI Clock Lo Time** = \( t_{\text{PCLK}} \times \text{SPIFC_CFG.loclk} \) where,

\[
\begin{align*}
t_{\text{PCLK}} &= \frac{1}{f_{\text{PCLK}}} = \frac{2}{f_{\text{SYS.CLK}}} \\
\end{align*}
\]

**Target SPI Clock Period** = **Target SPI Clock Hi Time** + **Target SPI Clock Lo Time**

**Target SPI Frequency** = \( \frac{1}{\text{Target SPI Clock Period}} \)

**Target SPI Clock Duty Cycle** = \( \frac{\text{Target SPI Clock Hi Time}}{\text{Target SPI Clock Lo Time} + \text{Target SPI Clock Hi Time}} \)
8.2.1.6  Slave Select Transaction Delay Configuration

The transaction delay and slave select timing with respect to the active or inactive slave select edge are determined by a combination of the following register fields:

- SPIXFC_CFG.ssact
- SPIXFC_CFG.ssiact
- SPIXFC_CFG.hiclk
- SPIXFC_CFG.loclk
- SPIXFC_CFG.mode
- SPIXFC_SP_CTRL.sckinh3 (if in mode 3)

Automatic slave selection de-assert for the SPIXF Master Controller occurs when the Transaction Header Deassert Slave Select field is set. The Slave Select is automatically de-asserted if the SPIXF Master Controller is disabled (SPIXFC_GEN_CTRL.enable = 0) or GCR_RST1.spixip = 1, resetting this peripheral.
Figure 8-3. SPIXFC Transaction Delay

SPIXFC MODE 0

SS

SCK

SPIXFC_CFG.ssact  SPIXFC_CFG.hiclk  SPIXFC_CFG.inact

SPIXFC_CFG.loclk

SPIXFC MODE 3, SPIXFC_SP_CTRL.sckinh3 = 0

An extra pulse is provided at the end of this transaction to comply with some SPI flash timing diagrams, and allow for higher speeds during SPI READ transactions. SCKINH3 = 0 should not be used for SPI WRITE transactions.

SS

SCK

SPIXFC_CFG.ssact  SPIXFC_CFG.hiclk  SPIXFC_CFG.inact

SPIXFC_CFG.loclk

SPIXFC_CFG.ssact

Bit 0

Extra Pulse

SPIXFC MODE 3, SPIXFC_SP_CTRL.sckinh3 = 1

When sending a SPI WRITE transaction in MODE 3, SCKINH3 should always be set in order to provide the correct number of clocks.

SS

SCK

SPIXFC_CFG.ssact  SPIXFC_CFG.hiclk  SPIXFC_CFG.inact

SPIXFC_CFG.loclk

SPIXFC_CFG.ssact

Bit 0
8.2.1.1.7 Slave Select

The SPIXF Master Controller operates with one slave device. A dedicated select pin for slave #0 is provided and controlled by hardware. Both execute-in-place and data storage are supported on slave #0.

8.2.1.1.8 Interrupts

Interrupt logic is provided to allow efficient servicing of the SPIXF Master Controller by firmware. You can group interrupts into the following two categories:

- Keeping the Transmit FIFO full
- Keeping the Receive FIFO empty

Programmable levels in the FIFO_CTRL register allow interrupt events to be issued if the Transmit FIFO falls below a certain level or if the Receive FIFO fills above a certain level. See the FIFO_CTRL register description for more information.

8.2.1.1.9 External SPI Flash Encryption

The user may optionally store encrypted data or code in the external SPI flash. Encryption of the SPI flash data is achieved using the cryptographic accelerator to encrypt the data and the SPIXF Master Controller to write the data. Data should be encrypted using AES-128, ECB mode.

Also, the following cryptographic accelerator control bits should be set when encrypting the SPIXF address space:

- \texttt{CRYPTO_CTRL.bsi}
- \texttt{CRYPTO_CTRL.bso}

Setting \texttt{CRYPTO_CTRL.src} = 0b11 selects the key stored for MDIU use in memory locations 0x4000 5080 to 0x4000 508F.

The data must be pre-processed with an address mask. 128 bits plain data blocks are XORed (\(^\)) with a 128-bit address mask to avoid patterns in encrypted data. The address mask, \(addr\_mask\) below, results in 128-bit aligned addressing by masking off the lower four bits of the input address (\(addr\_in\)) as follows:

\[
addr\_mask = addr\_in \& 0xFFFF \text{FFF0}
\]

For encryption, the data stored in the SPI flash, \(data\_out\) below, is calculated as follows:

\[
data\_out = \text{AES}(data\_in \oplus (addr\_mask << 96) | ((addr\_mask+4) << 64) \mid \ \((addr\_mask+8) << 32) | (addr\_mask+12)))\]

where:

\[
data\_in = \text{word0:word1:word2:word3} \text{ (big endian format)}
\]

When using the cryptographic accelerator, the input data should be loaded as follows:

\[
\begin{align*}
\text{crypto\_din0} &= \text{word0} \oplus (addr\_mask) \\
\text{crypto\_din1} &= \text{word1} \oplus (addr\_mask+4) \\
\text{crypto\_din2} &= \text{word2} \oplus (addr\_mask+8) \\
\text{crypto\_din3} &= \text{word3} \oplus (addr\_mask+12)
\end{align*}
\]

Once the encrypted data is available (either via FIFO or via Crypto Data Output Registers [3:0]), this data may be written to SPI flash using the SPIXF Master Controller.

The available output bytes from the cryptographic accelerator should be written to SPIXF flash memory as shown in Table 8-3.

<table>
<thead>
<tr>
<th>Least Significant Word</th>
<th>Most Significant Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{crypto_dout0}</td>
<td>\text{crypto_dout1}</td>
</tr>
<tr>
<td>\text{crypto_dout2}</td>
<td>\text{crypto_dout3}</td>
</tr>
</tbody>
</table>

\textbf{Table 8-3: Encrypted Data Write Order to SPIX Flash Memory}
8.2.1.2 SPIXF Master Controller Registers

See Table 3-1: APB Peripheral Base Address Map for the SPIXF Peripheral Base Offset Address.

Table 8-4. SPIXF Master Controller Register Offsets, Names, Access and Description

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>SPIXFC_CFG</td>
<td>R/W</td>
<td>SPIXF Controller Configuration Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>SPIXFC_SS_POL</td>
<td>R/W</td>
<td>SPIXF Controller Slave Select Polarity Register</td>
</tr>
<tr>
<td>[0x0008]</td>
<td>SPIXFC_GEN_CTRL</td>
<td>R/W</td>
<td>SPIXF Controller General Controller Register</td>
</tr>
<tr>
<td>[0x000C]</td>
<td>SPIXFC_FIFO_CTRL</td>
<td>R/W</td>
<td>SPIXF Controller FIFO Control and Status Register</td>
</tr>
<tr>
<td>[0x0010]</td>
<td>SPIXFC_SP_CTRL</td>
<td>R/W</td>
<td>SPIXF Controller Special Control Register</td>
</tr>
<tr>
<td>[0x0014]</td>
<td>SPIXFC_INT_FL</td>
<td>R/W</td>
<td>SPIXF Controller Interrupt Status Register</td>
</tr>
<tr>
<td>[0x0018]</td>
<td>SPIXFC_INT_EN</td>
<td>R/W</td>
<td>SPIXF Controller Interrupt Enable Register</td>
</tr>
</tbody>
</table>

8.2.1.3 SPIXF Master Controller Register Details

Table 8-5. SPIXF Controller Configuration Register

<table>
<thead>
<tr>
<th>SPIXF Controller Configuration Register</th>
<th>SPIXFC_CFG</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Access</td>
<td>Reset</td>
</tr>
<tr>
<td>31:24</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td>23:20</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19:18</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17:16</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### SPIXF Controller Configuration Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:12</td>
<td>loclk</td>
<td>R/W</td>
<td>0</td>
<td><strong>SCK Low Clocks</strong>&lt;br&gt;Number of system clocks that SCK is held low when SCK pulses are generated&lt;br&gt;0: 16 system clocks&lt;br&gt;1: 1 system clock&lt;br&gt;2: 2 system clocks&lt;br&gt;3: 3 system clocks&lt;br&gt;All other values: This value defines the number of system clock that SCK is held low.</td>
</tr>
<tr>
<td>11:8</td>
<td>hiclk</td>
<td>R/W</td>
<td>0</td>
<td><strong>SCK High Clocks</strong>&lt;br&gt;Number of system clocks that SCK is held high when SCK pulses are generated.&lt;br&gt;00: 16 system clocks.&lt;br&gt;All other values: This value defines the number of system clock that SCK is held high.</td>
</tr>
<tr>
<td>7:6</td>
<td>pgsz</td>
<td>R/W</td>
<td>0</td>
<td><strong>Page Size</strong>&lt;br&gt;Defines the number of bytes per page for transactions that define transfers in terms of pages.&lt;br&gt;00: 4 bytes&lt;br&gt;01: 8 bytes&lt;br&gt;10: 16 bytes&lt;br&gt;11: 32 bytes</td>
</tr>
<tr>
<td>5:4</td>
<td>mode</td>
<td>R/W</td>
<td>0</td>
<td><strong>SPI Mode.</strong>&lt;br&gt;Defines the SPI mode.&lt;br&gt;00: SPI Mode 0. Clock Polarity = 0, Clock Phase = 0&lt;br&gt;01: Invalid&lt;br&gt;10: Invalid&lt;br&gt;11: SPI Mode 3. Clock Polarity = 1, Clock Phase = 1</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>2:0</td>
<td>ssel</td>
<td>R/W</td>
<td>0</td>
<td><strong>Slave Select.</strong>&lt;br&gt;Only Slave 0 is supported&lt;br&gt;0b000: Slave 0 is selected&lt;br&gt;0b001-0b111: Invalid</td>
</tr>
</tbody>
</table>

### SPIXF Controller Slave Select Polarity Register

<table>
<thead>
<tr>
<th>SPIXF Controller Slave Select Polarity Register</th>
<th>SPIXFC_SS_POL</th>
<th>[0x0004]</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
<td>Reset</td>
</tr>
<tr>
<td>31:1</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>sspol_0</td>
<td>R/W</td>
<td>0</td>
</tr>
</tbody>
</table>
## Table 8-7. SPIXF Controller General Control Register

<table>
<thead>
<tr>
<th>SPIXF Controller General Control Register</th>
<th>SPIXFC_GEN_CTRL</th>
<th>[0x0008]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>31:26</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>sckfbinv</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>sckfb</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>smplss</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>simplerx</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>simple</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19:16</td>
<td>bbdatoe</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:12</td>
<td>bbdat</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:8</td>
<td>sdatain</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### SPIXF Controller General Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 6    | sckdr         | R/W    | 0     | **SCK Drive and State**  
When in Bit-Bang mode (SPIXFC_GEN_CTRL.bbmode = 1), this bit is written to control the output state of the SCK.  
0: SCK is 0.  
1: SCK is 1. |
| 5    | -             | R/W    | 0     | **Reserved for Future Use**  
Do not modify this field. |
| 4    | ssdr          | R/W    | 0     | **Slave Select Drive and State**  
This bit reflects the state of the slave select. This accounts for the polarity as defined in the SPIXFC_SS_POL register. When in Bit-Bang mode, this bit is written to control the output state of the slave select.  
0: Selected Slave Select Output is 0  
1: Selected Slave Select Output is 1. |
| 3    | bbmode        | R/W    | 0     | **Bit-Bang Mode**  
0: Disable Bit-Bang mode  
1: Enable Bit-Bang mode |
| 2    | rfifoen       | R/W    | 0     | **Receive FIFO Enable**  
Setting this bit enables the Receive FIFO. Clearing this bit disables the Receive FIFO and places it into a reset state.  
0: Disable result FIFO.  
1: Enable result FIFO. |
| 1    | tfifoen       | R/W    | 0     | **Transmit FIFO Enable**  
Setting this bit to 1 enables the Transmit FIFO. Clearing this bit disables the Transmit FIFO and places it into reset state.  
0: Disable Transmit FIFO.  
1: Enable Transmit FIFO. |
| 0    | enable        | R/W    | 0     | **SPI Master enable**  
Setting this bit to 1 enables SPI Master for processing transactions. Clearing this bit disables the SPI Master and puts the block into reset state.  
0: Disable SPI Master, putting it into a reset state.  
1: Enable SPI Master for processing transactions. |

### SPIXF Controller FIFO Control and Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:30  | -               | R/W    | 00    | **Reserved for Future Use**  
Do not modify this field. |
| 29:24  | rfifocnt        | R/W    | 0     | **Receive FIFO Entry Count**  
Current number of used entries (bytes) in Receive FIFO. Writes to this field are ignored. |
| 23:21  | -               | R/W    | 0     | **Reserved for Future Use**  
Do not modify this field. |
### SPIXF Controller FIFO Control and Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20:16</td>
<td>rfifolvl</td>
<td>R/W</td>
<td>0</td>
<td>Receive FIFO Almost Full Level</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The Almost Full flag is asserted when the number of used FIFO entries (bytes)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>exceed this value. FIFO depth is 32 bytes.</td>
</tr>
<tr>
<td>15:13</td>
<td></td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>12:8</td>
<td>tfifocnt</td>
<td>R/W</td>
<td>0</td>
<td>Transmit FIFO Entry Count</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Current number of used entries (words) in the Transmit FIFO. Writes to this</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>field are ignored.</td>
</tr>
<tr>
<td>7:4</td>
<td></td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>3:0</td>
<td>tfifolvl</td>
<td>R/W</td>
<td>0</td>
<td>Transmit FIFO Almost Empty Level</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The Almost Empty flag is asserted when the number of unused FIFO entries in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>words exceeds this value. FIFO depth is 16 words.</td>
</tr>
</tbody>
</table>

Table 8-9. SPIXF Controller Special Control Register

<table>
<thead>
<tr>
<th>SPIXF Controller Special Control Register</th>
<th>SPIXFC_SP_CTRL</th>
<th>[0x0010]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:17</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>16</td>
<td>sckinh3</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>In SPI mode 3, some SPI flash read timing diagrams show the last SCK going</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>low prior to de-assertion. The default is to support this additional falling</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>edge of the clock. When this bit is set, and the device is in SPI mode 3,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the SPI clock is held high while slave select is de-asserted. This is to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>support some SPI flash write timing diagrams.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: Allow trailing SCK low pulse prior to slave select de-assertion.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Inhibit trailing SCK low pulse prior to slave select de-assertion.</td>
</tr>
<tr>
<td>15:12</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>11:8</td>
<td>sdiooe</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Defines whether the output is enabled for each SDIO pin.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 11: SDIO[3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 10: SDIO[2]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 9: SDIO[1]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 8: SDIO[0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: SDIO output disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: SDIO output enabled.</td>
</tr>
<tr>
<td>7:4</td>
<td>sdioout</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Defines the values for the SDIO outputs when in Sample Mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(SPIXFC_SP_CTRL.sampl=1).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 7: SDIO[3]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 6: SDIO[2]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 5: SDIO[1]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 4: SDIO[0]</td>
</tr>
</tbody>
</table>
### SPIXF Controller Special Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3:1</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field</td>
</tr>
<tr>
<td>0</td>
<td>sampl</td>
<td>R/W</td>
<td>0</td>
<td>SDIO Sample Mode Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting this bit to a 1 enables the ability to drive SDIO outputs prior to the assertion of Slave Select. This bit must only be set when the SPIXF bus is idle and the transmit FIFO is empty. This bit is automatically cleared by hardware after the next slave select assertion.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Sample Mode disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Sample mode enabled</td>
</tr>
</tbody>
</table>

Table 8-10. SPIXF Controller Interrupt Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:6</td>
<td>-</td>
<td>R/W1C</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field</td>
</tr>
<tr>
<td>5</td>
<td>rfifoaf</td>
<td>R/W1C</td>
<td>0</td>
<td>Receive FIFO Almost Full Flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This flag is set by hardware when the Receive FIFO is almost full as defined by SPIXFC_FIFO_CTRL.rfifolvl.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Receive FIFO level below the Almost Full level</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Receive FIFO level at almost full level</td>
</tr>
<tr>
<td>4</td>
<td>tfifoae</td>
<td>R/W1C</td>
<td>1</td>
<td>Transmit FIFO Almost Empty Flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This flag is set by hardware when the Transmit FIFO is almost empty as defined by SPIXFC_FIFO_CTRL.tfifolvl. This does not depend on block enable or the slave select value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Transmit FIFO not Almost Empty</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Transmit FIFO Almost Empty</td>
</tr>
<tr>
<td>3</td>
<td>rdone</td>
<td>R/W1C</td>
<td>0</td>
<td>Receive Done Interrupt Status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This flag is set by hardware when the Receive FIFO is not empty, and the slave select is deasserted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Receive FIFO ready</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Receive FIFO Not ready</td>
</tr>
<tr>
<td>2</td>
<td>trdy</td>
<td>R/W1C</td>
<td>0</td>
<td>Transmit Ready Interrupt Status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This flag is set by hardware when the Transmit FIFO is empty, and the slave select is deasserted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Transmit FIFO not ready</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Transmit FIFO is ready</td>
</tr>
<tr>
<td>1</td>
<td>rstall</td>
<td>R/W1C</td>
<td>0</td>
<td>Receive Stalled Interrupt Flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This flag is set by hardware when the Receive FIFO is full, and the selected slave select is asserted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Normal FIFO operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Stalled FIFO.</td>
</tr>
</tbody>
</table>
### SPIXF Controller Interrupt Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>tstall</td>
<td>R/W1C</td>
<td>0</td>
<td>Transmit Stalled Interrupt Flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This flag is set by hardware when the Transmit FIFO is empty, and the selected slave select is asserted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Normal FIFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Stalled FIFO.</td>
</tr>
</tbody>
</table>

Table 8-11. SPIXF Controller Interrupt Enable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:6</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>5</td>
<td>rfifoafie</td>
<td>R/W</td>
<td>0</td>
<td>Receive FIFO Almost Full Interrupt Enable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting this bit enables interrupt generation when the SPIXF_INT_FL.rfifoaf flag is set. Clearing this bit means that no interrupt is generated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable Receive FIFO Almost Full Interrupt.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable Receive FIFO Almost Full Interrupt.</td>
</tr>
<tr>
<td>4</td>
<td>tfifoaeie</td>
<td>R/W</td>
<td>1</td>
<td>Transmit FIFO Almost Empty Interrupt Enable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting this bit enables interrupt generation when the SPIXF_INT_FL.tfifoae flag is set. Clearing this bit means that no interrupt is generated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable Transmit FIFO Almost Empty Interrupt.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable Transmit FIFO Almost Empty Interrupt.</td>
</tr>
<tr>
<td>3</td>
<td>rdoneie</td>
<td>R/W</td>
<td>0</td>
<td>Receive Done Interrupt Enable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting this bit enables interrupt generation when the SPIXF_INT_FL.rdone flag is set. Clearing this bit means that no interrupt is generated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable Receive Done Interrupt.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable Receive Done Interrupt.</td>
</tr>
<tr>
<td>2</td>
<td>trdyie</td>
<td>R/W</td>
<td>0</td>
<td>Transmit Ready Interrupt Enable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting this bit enables interrupt generation when the SPIXF_INT_FL.trdy flag is set. Clearing this bit means that no interrupt is generated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable Transmit Ready Interrupt.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable Transmit Ready Interrupt.</td>
</tr>
<tr>
<td>1</td>
<td>rstallie</td>
<td>R/W</td>
<td>0</td>
<td>Receive Stalled Interrupt Enable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting this bit enables the Receive Stalled Interrupt. Clearing this bit means that no interrupt is generated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable Receive Stalled Interrupt.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable Receive Stalled Interrupt.</td>
</tr>
<tr>
<td>0</td>
<td>tstallie</td>
<td>R/W</td>
<td>0</td>
<td>Transmit Stalled Interrupt Enable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting this bit enables interrupt generation when the SPIXF_INT_FL.tstall flag is set. Clearing this bit means that no interrupt is generated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable Transmit Stalled Interrupt.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable Transmit Stalled Interrupt.</td>
</tr>
</tbody>
</table>
8.2.1.4 SPIXF Master Controller FIFO Registers

See Table 3-2: AHB Peripheral Base Address Map for the SPIXF Master Controller FIFO Peripheral Base Address.

Table 8-12. SPIXF Master Controller FIFO Register Offsets, Names, Access and Description

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>SPIXFC_FIFO_TX</td>
<td>WO</td>
<td>SPIXF Master Controller TX FIFO Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>SPIXFC_FIFO_RX</td>
<td>RO</td>
<td>SPIXF Master Controller RX FIFO Register</td>
</tr>
</tbody>
</table>

8.2.1.5 SPIXF Master Controller FIFO Register Details

Table 8-13. SPIXF Master Controller TX FIFO Register

<table>
<thead>
<tr>
<th>SPIXF Master Controller TX FIFO Register</th>
<th>SPIXFC_FIFO_TX [0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
</tr>
<tr>
<td>31:0</td>
<td>txfifo</td>
</tr>
</tbody>
</table>

Table 8-14. SPIXF Master Controller RX FIFO Register

<table>
<thead>
<tr>
<th>SPIXF Master Controller RX FIFO Register</th>
<th>SPIXFC_FIFO_RX [0x0004]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
</tr>
<tr>
<td>31:0</td>
<td>rxfifo</td>
</tr>
</tbody>
</table>

8.2.2 SPIXF Master

The SPIXF Master (SPIXFM) is an AHB slave interface that is driven by a 16KB Unified Instruction and Constant cache to support cache operation. The AHB slave supports either instruction execution or fetching of data from external SPI flash. This interface is accessible to firmware using an AHB interface to support high-speed data transfer. The address for SPI flash access is determined by the AHB access and is mapped from address 0x0800 0000 to 0xFFFF FFFF for a total addressable space of 128MB.

The command used to transfer SPI flash data is configured using firmware. Then, the access to SPI flash space (either code execution or data) may be performed by firmware. The AHB transaction initiated by the firmware provides address and other transaction critical parameters to control the data transfer from the external SPI flash.

You should exercise care in choosing the correct configuration and command to support the speed of data transfer. The SPIXF Master provides SCK periods as fast as the AHB clock speed divided by two. The external SPI flash configuration to support data transfer rates must be performed by the SPIXF Master Controller.
8.2.2.1 SPIXF Pin Configuration

The SPIXF Master and SPIXF Master Controller use a highly-configurable, flexible, and efficient interface supporting single, dual, or quad I/O. Dedicated pins are provided to support high-speed communication. The following pin configurations are supported and shown in Figure 8-4:

- Four-wire SPI: SS, SCK, MOSI on SDIO0, and MISO on SDIO1
- Dual SPI: SS, SCK, SDIO0, and SDIO1
- Quad SPI: SS, SCK, SDIO0, SDIO1, SDIO2, and SDIO3

Figure 8-4. Supported SPI configuration

8.2.2.2 Slave-Select Transaction Delay Configuration

The transaction delay and clock timing with respect to the active or inactive slave-select edge is determined by a combination of the following register fields:

- SPIXF_CFG.ssact
- SPIXF_CFG.ssinact
- SPIXF_CFG.hclk
- SPIXF_CFG.loclk
- SPIXF_CFG.mode

Automatic slave-select de-assertion only occurs when the next flash address fetched is not contiguous to the current flash address that is being read or used for execution. The SPIXF does not automatically de-assert slave selection under any other circumstance including data read or execution of areas outside of the SPIXF space. For these cases, manual control of the slave select is provided. Invoke manual control only when running from internal memory. You can de-assert slave-select safely by setting GCR_RST1.spixip. This resets the SPIXF block (including turning off decryption if previously enabled) and
causes the slave select to de-assert. The SPIXF block requires reconfiguration prior to subsequent access to external SPI flash space either for execution or data reads.

Figure 8-5. SPIXF Delay Configuration

8.2.2.3 SPIXF Read Sequence Configuration and Control

Assertion of SPIXF slave select followed by the read command, then the read address. After the read address is sent 0 or more clocks are generated (called dummy bytes or mode clocks) to allow the flash to access the data being addressed. The remainder of the SPI access is read data. Sequential bytes are read until the de-assertion of SPIXF slave select.

Depending on the read command and the SPI flash configuration, the read command is sent over 1, 2, or 4 bits per clock. The same is true for the address, data, and mode/dummy clocks. Also, configure the device to eliminate the sending of the read command once the command is sent to the SPI flash device. This is enabled and disabled through special data sent during the mode or dummy period between address and read data.
### 8.2.2.4 Sample SPIXF Master Configuration - Execute Code

Complete the following steps to execute the SPIXF Master Configuration sample:

1. Turn on ICache XIP Clock (GCR_PCLK_DIS1.icachexipf = 1).
   a. The cache can be put into different power states. See GCR_MEM_CLK for options.
2. Configure the SPIXF Master mode, slave select polarity, slave number, and slave select timing.
   a. Example:
      i. SPI Mode 0
      ii. Slave select high
      iii. Slave #0
      iv. 1 SPI clock per 2 AHB clocks
      v. SPIXF_M_CFG = 0x1104.
3. Configure the command value, the command, address, data width, and whether the address is three- or four-byte mode.
   a. Example Read command:
      i. command value = 0x03
      ii. command width = single data I/O
      iii. address bit = single data I/O.
      iv. data width = single data I/O
      v. 3-byte address mode
      vi. SPIXF_FETCH_CTRL = 0x0003.
4. Configure the SPIXF mode/dummy field and the data for the mode/dummy field
   a. Example:
      i. Mode clocks = 0 (no dummy field)
      ii. SPIXF_MODE_CTRL = 0x0
5. Enable the SPIXF feedback clock control.
   a. Example:
      i. SPIXF feedback clock enabled using non-inverted serial feedback clock
      ii. SPIXF_FB_CTRL = 0x0001.
6. Jump to the start of application code in SPI flash space.
   a. Example pseudo code:
      i. jump_to_external_flash = (void) (0x08000001)
      ii. jump_to_external_flash()
SPIXF slave select is low, the user must not be executing from SPIXF space, and the SPIXF block should be reset by setting `GCR_RST1.spixip = 1`.

### 8.2.2.7 External SPI Flash Decryption

If data in the SPI flash is encrypted when written, it might be transparently decrypted on read back using either code execution or data reads. Decryption is not enabled by default. Setting `SPIXF_SEC_CTRL.dec_en = 1` enables the Memory Decryption Integrity Unit (MDIU). The MDIU uses an AES-128 algorithm in ECB mode. This key is written by the user to the register file locations 0x4000 5020 to 0x4000 502F, which is automatically used by the MDIU for decryption.

See *SPIF Master Controller* for information about data encryption for external SPI flash.

### 8.2.2.8 SPIXF Master Registers

See *Table 3-1: APB Peripheral Base Address Map* for the SPIXF Master Peripheral Base Offset Address.

Reserved register bits should only be written as 0.

#### Table 8-15. SPIXF Master Register Offsets, Names, Access and Description

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>SPIXF_CFG</td>
<td>R/W</td>
<td>SPIXF Configuration Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>SPIXF_FETCH_CTRL</td>
<td>R/W</td>
<td>SPIXF Fetch Control Register</td>
</tr>
<tr>
<td>[0x0008]</td>
<td>SPIXF_MODE_CTRL</td>
<td>R/W</td>
<td>SPIXF Mode Control Register</td>
</tr>
<tr>
<td>[0x000C]</td>
<td>SPIXF_MODE_DATA</td>
<td>R/W</td>
<td>SPIXF Mode Data Register</td>
</tr>
<tr>
<td>[0x0010]</td>
<td>SPIXF_FB_CTRL</td>
<td>R/W</td>
<td>SPIXF SCK Feedback Control Register</td>
</tr>
<tr>
<td>[0x0014]</td>
<td>SPIXF_IO_CTRL</td>
<td>R/W</td>
<td>SPIXF I/O Control Register</td>
</tr>
<tr>
<td>[0x0020]</td>
<td>SPIXF_SEC_CTRL</td>
<td>R/W</td>
<td>SPIXF Memory Security Register</td>
</tr>
<tr>
<td>[0x0024]</td>
<td>SPIXF_BUS_IDLE</td>
<td>R/W</td>
<td>SPIXF Bus Idle Detection</td>
</tr>
</tbody>
</table>

### 8.2.2.9 SPIXF Master Register Details

#### Table 8-16. SPIXF Configuration Register

<table>
<thead>
<tr>
<th>SPIXF Configuration Register</th>
<th>SPIXF_CFG</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>--------</td>
</tr>
<tr>
<td>31:20</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td>19:18</td>
<td>ssiact</td>
<td>R/W</td>
</tr>
</tbody>
</table>
### Table 8-17. SPIXF M Configuration Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>17:16</td>
<td>ssact</td>
<td>R/W</td>
<td>0</td>
<td>Slave Select Active Timing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Controls delay from assertion of slave select to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>start of the SCK pulse and delay from the end of</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SCK pulses to de-assertion of slave select. See</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8.2.1.1.6, above, for details on slave select</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>transaction delay configuration.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00: 0 system clocks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01: 2 system clocks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b10: 4 system clocks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b11: 8 system clocks</td>
</tr>
<tr>
<td>15:12</td>
<td>hiclk</td>
<td>R/W</td>
<td>0</td>
<td>SCK High Clocks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Number of system clocks that SCK is held high</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>when SCK pulses are generated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Invalid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All other values: The number of system clocks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>that SCK is held high.</td>
</tr>
<tr>
<td>11:8</td>
<td>loclk</td>
<td>R/W</td>
<td>0</td>
<td>SCK Low Clocks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Number of system clocks that SCK is held low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>when SCK pulses are generated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Invalid</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All other values: The number of system clocks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>that SCK is held low.</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>6:4</td>
<td>ssel</td>
<td>R/W</td>
<td>0</td>
<td>Slave Select</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Only valid value is zero.</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>2</td>
<td>sspol</td>
<td>R/W</td>
<td>1</td>
<td>Slave Select Polarity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit controls the polarity of the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>slave select.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Slave Select active high</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Slave Select active low</td>
</tr>
<tr>
<td>1:0</td>
<td>mode</td>
<td>R/W</td>
<td>0</td>
<td>SPI mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set this field to the required SPI mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00: SPI mode 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01: Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b10: Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b11: SPI mode 3</td>
</tr>
</tbody>
</table>

### Table 8-17. SPIXF M Fetch Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:17</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>16</td>
<td>addr4</td>
<td>R/W</td>
<td>0</td>
<td>Four-Byte Address mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Enables 4-byte Flash Address mode. Defaults to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>value as defined by parameter in instantiation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>User can override.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: 3-byte address mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: 4-byte address mode</td>
</tr>
</tbody>
</table>
### SPIXF M Fetch Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:14</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>13:12</td>
<td>data_width</td>
<td>R/W</td>
<td>0</td>
<td>Data Width</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Number of data I/O used to receive data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00: Single SDIO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01: Dual SDIO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b10: Quad SDIO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b11: Reserved</td>
</tr>
<tr>
<td>11:10</td>
<td>addr_width</td>
<td>R/W</td>
<td>0</td>
<td>Address Width</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Number of data I/O used to send address and mode/dummy clocks.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00: Single SDIO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01: Dual SDIO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b10: Quad SDIO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b11: Reserved</td>
</tr>
<tr>
<td>9:8</td>
<td>cmdwth</td>
<td>R/W</td>
<td>0</td>
<td>Command Width</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Number of data I/O used to send commands.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00: Single SDIO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01: Dual SDIO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b10: Quad SDIO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b11: Reserved</td>
</tr>
<tr>
<td>7:0</td>
<td>cmdval</td>
<td>R/W</td>
<td>0</td>
<td>Command Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Command value sent to target to initiate fetching from SPI flash.</td>
</tr>
</tbody>
</table>

### SPIXF M Mode Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:10</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>9</td>
<td>mode_send</td>
<td>R/W</td>
<td>0</td>
<td>Mode Send</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting this field ensures that the next SPI flash transaction will send the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mode byte as defined in the SPIXF_MODE_DATA.mddata field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When this field is set, the next SPI flash read operation exits continuous</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>mode cleanly. This field and the SPIXF_MODE_CTRL.nocmd field is automatically</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>cleared by hardware after the next SPI transaction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No Action.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Send Mode Byte on next transaction</td>
</tr>
<tr>
<td>8</td>
<td>nocmd</td>
<td>R/W</td>
<td>0</td>
<td>No Command Mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Read command sent only once after this bit is set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Send read command every time SPI transaction is initiated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Send read command on first transaction only and not on subsequent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>transactions.</td>
</tr>
<tr>
<td>7:4</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
</tbody>
</table>

Table 8-18. SPIXF M Mode Control Register
### SPIXF Mode Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3:0</td>
<td>mdclk</td>
<td>R/W</td>
<td>0</td>
<td>Mode Clocks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Number of SPI clocks needed during the mode/dummy phase of fetch.</td>
</tr>
</tbody>
</table>

**Table 8-19. SPIXF Mode Data Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>mdoe</td>
<td>R/W</td>
<td>0</td>
<td>Mode Output Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Output enable state for each corresponding data bit in <code>SPIXF_MODE_DATA.mddata</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Output enable off, I/O is tristate stated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Output enable on, I/O is driving <code>SPIXF_MODE_DATA.mddata</code>.</td>
</tr>
<tr>
<td>15:0</td>
<td>mddata</td>
<td>R/W</td>
<td>0</td>
<td>Mode Data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Specifies the data to send with the dummy/mode clocks.</td>
</tr>
</tbody>
</table>

### SPIXF SCK Feedback Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:2</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>1</td>
<td>fbinv</td>
<td>R/W</td>
<td>0</td>
<td>SCK feedback Clock inversion.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The feedback clock can be phase selected to increase the timing margin for</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>input data from the slave external flash device.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Non-inverted SCK is used for feedback clock</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Inverted SCK is used for feedback clock</td>
</tr>
<tr>
<td>0</td>
<td>fbmd</td>
<td>R/W</td>
<td>0</td>
<td>SCK Feedback Mode Enable. Enable SCK feedback mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable SCK feedback mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable SCK feedback mode</td>
</tr>
</tbody>
</table>

### SPIXF I/O Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:5</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>4:3</td>
<td>pupdctrl</td>
<td>R/W</td>
<td>1</td>
<td>IO Pullup/Pulldown Control</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>These bits control the pullups and pulldowns associated with all SPIXF SDIO pins.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00: tristate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01: pullup</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b10: pulldown</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b11: pullup</td>
</tr>
<tr>
<td>2</td>
<td>sdio_ds</td>
<td>R/W</td>
<td>1</td>
<td>SDIO Drive Strength</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit controls the drive strength of all SDIO pins.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Low Drive Strength.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Hi Drive Strength.</td>
</tr>
</tbody>
</table>
### SPIXM I/O Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ss_ds</td>
<td>R/W</td>
<td>1</td>
<td><strong>Slave Select Drive Strength</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit controls the drive strength on the dedicated slave select pin.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Low Drive Strength.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Hi Drive Strength.</td>
</tr>
<tr>
<td>0</td>
<td>sck_ds</td>
<td>R/W</td>
<td>1</td>
<td><strong>SCK Drive Strength</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit controls the drive strength on the SCK pin.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Low Drive Strength.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Hi Drive Strength.</td>
</tr>
</tbody>
</table>

### SPIXM Memory Security Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:2</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>1</td>
<td>auth_disable</td>
<td>R/W</td>
<td>0</td>
<td><strong>Integrity Enable.</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Integrity checking enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Integrity checking disabled.</td>
</tr>
<tr>
<td>0</td>
<td>dec_en</td>
<td>R/W</td>
<td>0</td>
<td><strong>Decryption Enable.</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable decryption of SPIXF data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable decryption of SPIXF data.</td>
</tr>
</tbody>
</table>

### SPIXM Bus Idle Detection

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>15:0</td>
<td>busidle</td>
<td>R/W</td>
<td>0</td>
<td><strong>Bus Idle Timer Limit</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A 16-bit timer will be triggered for each external access. The timer will be restarted if another access is performed before the timer expires. When the timer expires, slave select will be deactivated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This register contains the limit (expiration) value for the timer. A value of 0 will disable the bus idle detection and non-zero values enable bus idle detection.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This feature is useful when fetching code out of I-cache, ROM or in SLEEP and DEEPSLEEP modes. When this number is too small, Slave Select will be deactivated on every access, which may reduce current consumption, but decreases performance.</td>
</tr>
</tbody>
</table>
8.3 SPI Execute-in-Place RAM (SPIXR)

The SPI Execute-in-Place RAM Master Controller (SPIXR) is an instantiation of the Quad SPI Interface with the following features:

- Four SPI modes (mode 0, 1, 2, and 3)
- Master mode only support
- Dual SPI Mode with two bidirectional serial data I/O (SDIO) lines
- High Performance Quad SPI Mode with four bidirectional SDIO lines
- Programmable Serial Clock (SCK) frequency and duty cycle with 48MHz maximum
- 32-byte Transmit FIFO, 32-byte Receive FIFO with DMA support backed by a 16KB Data Cache

The SPIXR Master Controller allows the CPU to transparently execute instructions stored in an external SPI SRAM device. Instructions fetched using the SPIXR Master Controller are cached just like instructions fetched from internal program memory. You can also use the SPIXR Master Controller to access large amounts of external data that would otherwise reside in internal data memory.

Prior to using the SPI SRAM device, you must configure the SPIXR interface.

The command used to transfer SPI SRAM data is configured using firmware. Then, the access to SPI SRAM space (either code execution or data) may be performed by firmware. The AHB transaction initiated by the firmware provides address and other transaction critical parameters to control the data transfer from the external SPI SRAM.

Care should be exercised when choosing the correct configuration and command to support the speed of data transfer. The SPIXR Master Controller provides SCK periods as fast as the AHB clock speed divided by two. The external SPI SRAM configuration to support data transfer rates must be performed by the SPIXR Master Controller.
**8.3.1 SPIXR Master Controller Registers**

See *Table 3-1: APB Peripheral Base Address Map* for the SPIXR Peripheral Base Address

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>SPIXR_DATA</td>
<td>R/W</td>
<td>SPIXR FIFO Data Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>SPIXR_CTRL1</td>
<td>R/W</td>
<td>SPIXR Master Signals Control Register</td>
</tr>
<tr>
<td>[0x0008]</td>
<td>SPIXR_CTRL2</td>
<td>R/W</td>
<td>SPIXR Transmit Packet Size Register</td>
</tr>
<tr>
<td>[0x000C]</td>
<td>SPIXR_CTRL3</td>
<td>R/W</td>
<td>SPIXR Static Configuration Register</td>
</tr>
<tr>
<td>[0x0010]</td>
<td>SPIXR_SS_TIME</td>
<td>R/W</td>
<td>SPIXR Slave Select Timing Register</td>
</tr>
<tr>
<td>[0x0014]</td>
<td>SPIXR_BRG_CTRL</td>
<td>R/W</td>
<td>SPIXR Master Baud Rate Register</td>
</tr>
<tr>
<td>[0x001C]</td>
<td>SPIXR_DMA</td>
<td>R/W</td>
<td>SPIXR DMA Control Register</td>
</tr>
<tr>
<td>[0x0020]</td>
<td>SPIXR_INT_FL</td>
<td>R/W1C</td>
<td>SPIXR Interrupt Status Flags Register</td>
</tr>
<tr>
<td>[0x0024]</td>
<td>SPIXR_INT_EN</td>
<td>R/W</td>
<td>SPIXR Interrupt Enable Register</td>
</tr>
<tr>
<td>[0x0028]</td>
<td>SPIXR_WAKE_FL</td>
<td>R/W1C</td>
<td>SPIXR Wakeup Status Flags Register</td>
</tr>
<tr>
<td>[0x002C]</td>
<td>SPIXR_WAKE_EN</td>
<td>R/W</td>
<td>SPIXR Wakeup Enable Register</td>
</tr>
</tbody>
</table>
### SPIXR Register Details

**Table 8-25. SPIXR FIFO Data Register**

<table>
<thead>
<tr>
<th>SPIXR FIFO Data Register</th>
<th>SPIXR_DATA</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 31:0 | data | R/W | 0 | SPIXR FIFO Data  
FIFO data for the SPIXR. |

**Table 8-26. SPIXR Master Signals Control Register**

<table>
<thead>
<tr>
<th>SPIXR Master Signals Control Register</th>
<th>SPIXR_CTRL1</th>
<th>[0x0004]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 31:24 | - | R/W | 0 | Reserved for Future Use  
Do not modify this field. |
| 23:16 | ss | R/W | 0 | Master Slave Select  
This field selects the slave select pin for the SPIXR interface.  
0: The slave select pin is not selected for the SPIXR.  
1: The SPIXR slave select pin is used for the SPIXR.  
*Note: This field must be set to 1 for SPIXR operation.* |
| 15:9 | - | R/W | 0 | Reserved for Future Use  
Do not modify this field. |
| 8 | ss_ctrl | R/W | 0 | Master Slave Select Control  
Setting this field to 1 leaves the Slave Select signal asserted at the end of the transmission. This enables multiple transmissions to occur without the Slave Select signal being deasserted. At the completion of all transmissions with the SPIXR device, this field must be set to 0 to deassert the Slave Select line.  
0: Slave Select is deasserted at the end of a transmission  
1: Slave Select stays asserted at the end of a transmission |
| 7:6 | - | R/W | 0 | Reserved for Future Use  
Do not modify this field. |
| 5 | start | R/W1AC | 0 | Master Start Data Transmission  
Set this field to 1 to start the transaction with the slave device. Hardware automatically clears this field after the transaction is started.  
0: No SPIXR data transmission is in process.  
1: Master initiates a data transmission.  
*Note: Ensure that all pending transactions are complete before writing a 1.*  
*Warning: If the transmit FIFO is enabled, there must be at least one byte in the TX FIFO before setting this bit.* |
### SPIXR Master Signals Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>ss_io</td>
<td>R/W</td>
<td>0</td>
<td>Master Slave Select Signal Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This field must be set to 0 for SPIXR operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Slave Select is an output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> The SPIXR only operates as a SPI master in single master mode. Writing 1 to this</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>field is invalid.</td>
</tr>
<tr>
<td>3:2</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>1</td>
<td>master</td>
<td>R/W</td>
<td>0</td>
<td>SPIXR Master Mode Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This field must be set to 1 to use the SPIXR peripheral.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Master Mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> The SPIXR peripheral only operates in Master Mode. Writing 0 to this field is invalid.</td>
</tr>
<tr>
<td>0</td>
<td>enable</td>
<td>R/W</td>
<td>0</td>
<td>SPIXR Enable/Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set this field to 1 to enable the SPIXR peripheral.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: SPIXR is disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: SPIXR is enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> Setting this field to 0 disables the SPIXR but maintains all registers and the FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>data.</td>
</tr>
</tbody>
</table>

**Table 8-27. SPIXR Transmit Packet Size Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>rx_num_char</td>
<td>R/W</td>
<td>0</td>
<td>Number of characters to receive in RX FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The number of characters in the RX FIFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> This field is only used if the SPIXR is configured for Three-Wire SPI operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><em>SPIXR_CTRL3</em>.three_wire = 1.</td>
</tr>
<tr>
<td>15:0</td>
<td>tx_num_char</td>
<td>R/W</td>
<td>0</td>
<td>Number of characters to transmit from TX FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The number of characters in the TX FIFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> If the SPIXR is set to Four-wire mode, <em>SPIXR_CTRL3</em>.three_wire = 0, this field</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>represents both the RX and TX FIFO character count.</td>
</tr>
</tbody>
</table>

**Table 8-28. SPIXR Static Configuration Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:17</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>16</td>
<td>sspol</td>
<td>R/W</td>
<td>0</td>
<td>Slave Select Polarity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: SS is active low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: SS is active high</td>
</tr>
<tr>
<td>15</td>
<td>three_wire</td>
<td>R/W</td>
<td>0</td>
<td>Three-Wire Mode Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Four-wire mode enabled (Single Mode only)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Three-wire mode enabled</td>
</tr>
</tbody>
</table>
### SPIR Static Configuration Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>13:12</td>
<td>data_width</td>
<td>R/W</td>
<td>0</td>
<td><strong>SPIR Data Width</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sets the number of data lines (SDIO pins) for communication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: 1-data pin (Single Mode)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: 2-data pins (Dual Mode)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2: 4-data pins (Quad Mode)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3: Reserved for Future Use</td>
</tr>
<tr>
<td>11:8</td>
<td>numbits</td>
<td>R/W</td>
<td>0</td>
<td><strong>Number of Bits per Character</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sets the number of bits per character for an SPIR transaction.</td>
</tr>
<tr>
<td>7:5</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>4</td>
<td>sck_inv</td>
<td>R/W</td>
<td>0</td>
<td><strong>SCK Inverted</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This field must always be set to 0 for SPIR operation. SCK inversion for a</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>specific mode is not supported by the SPIR peripheral. Use the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SPIR_CTRL3.spol field to set the polarity of the clock for a given mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Normal SCK output.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Invalid, not supported.</td>
</tr>
<tr>
<td>3:2</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>1</td>
<td>cpol</td>
<td>R/W</td>
<td>0</td>
<td><strong>Clock Polarity</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sets the SPIR SCK clock polarity for the supported modes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Normal clock. Use when in SPI Mode 0 and Mode 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Inverted clock. Use when in SPI Mode 2 and Mode 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> This field is set depending on the SPI Mode configuration.</td>
</tr>
<tr>
<td>0</td>
<td>cpha</td>
<td>R/W</td>
<td>0</td>
<td><strong>Clock Phase</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sets the SPIR SCK clock phase.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Data sampled on clock rising edge. Use when in SPI Mode 0 and Mode 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Data sampled on clock falling edge. Use when in SPI Mode 1 and Mode 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> This field must be set based on the SPI Mode configuration.</td>
</tr>
</tbody>
</table>

### SPIR Slave Select Timing Register

<table>
<thead>
<tr>
<th>SPIR Slave Select Timing Register</th>
<th>SPIR_SS_TIME</th>
<th>[0x0010]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Access</td>
<td>Reset</td>
</tr>
<tr>
<td>31:24</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:16</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### SPIXR Slave Select Timing Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:8</td>
<td>ssact2</td>
<td>R/W</td>
<td>0</td>
<td><strong>Slave Select Active After Last SCK</strong>&lt;br&gt;Number of system clock cycles that SS is active from the last SCK edge to when SS is inactive.</td>
</tr>
<tr>
<td>7:0</td>
<td>ssact1</td>
<td>R/W</td>
<td>0</td>
<td><strong>Slave Select Active Before SCK</strong>&lt;br&gt;Number of system clock cycles between the time SS is asserted until the first SCK edge.</td>
</tr>
</tbody>
</table>

### SPIXR Master Baud Rate Generator

<table>
<thead>
<tr>
<th>SPIXR Master Baud Rate Generator Register</th>
<th>SPIXR_BRG_CTRL</th>
<th>[0x0014]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
<td>Reset</td>
</tr>
<tr>
<td>31:20</td>
<td>-</td>
<td>R/W</td>
<td></td>
</tr>
<tr>
<td>19:16</td>
<td>scale</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>scale</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>9 - 15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 15:8 | hi   | R/W    | 0 | **SCK Hi Clock Cycles Control**<br>Setting this field to 0 disables the high duty cycle control for SCK.<br>Setting this field to any non-zero value sets the high cycle time to: $SCK_{HIGH} = hi \times SPIXR_{CLK}$

*Note: if SPIXR_BRG_CTRL.scale = 0, SPIXR_BRG_CTRL.hi = 0, and SPIXR_BRG_CTRL.lo = 0, character sizes of 2 and 10 bits are not supported.*
### SPIXR Master Baud Rate Generator Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:0</td>
<td>lo</td>
<td>R/W</td>
<td>0</td>
<td><strong>SCK Low Clock Cycles Control</strong>&lt;br&gt;Setting this field to 0 disables the low duty cycle control for SCK.&lt;br&gt;Setting this field to any non-zero value sets the high cycle time to:&lt;br&gt;&lt;br&gt;SCK_Low = ( lo \times SPIXR_{CLK} )&lt;br&gt;&lt;br&gt;Note: If SPIXR_BRG_CTRL.scale = 0, SPIXR_BRG_CTRL.hi = 0, and SPIXR_BRG_CTRL.lo = 0, character sizes of 2 and 10 bits are not supported.</td>
</tr>
</tbody>
</table>

### SPIXR DMA Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>rx_dma_en</td>
<td>R/W</td>
<td>0</td>
<td><strong>RX DMA Enable</strong>&lt;br&gt;Enable or disable the RX DMA.&lt;br&gt;&lt;br&gt;0: RX DMA is disabled. Any pending DMA requests are cleared&lt;br&gt;1: RX DMA is enabled</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>29:24</td>
<td>rx_fifo_cnt</td>
<td>RO</td>
<td>0</td>
<td><strong>Number of Bytes in the RX FIFO</strong>&lt;br&gt;Reading this field returns the number of bytes currently in the RX FIFO</td>
</tr>
<tr>
<td>23</td>
<td>rx_fifo_clear</td>
<td>R/W0</td>
<td>0</td>
<td><strong>Clear the RX FIFO</strong>&lt;br&gt;Set this field to clear the RX FIFO and all related RX FIFO flags in the SPIXR_INT_FL register. When cleared, the SPIXR_INT_FL.rx_fifo_empty flag is set by hardware.&lt;br&gt;&lt;br&gt;1: Clear the RX FIFO and any pending RX FIFO flags in SPIXR_INT_FL. This should be done when the RX FIFO is inactive.&lt;br&gt;&lt;br&gt;Note: Writing 0 has no effect.</td>
</tr>
<tr>
<td>22</td>
<td>rx_fifo_en</td>
<td>R/W</td>
<td>0</td>
<td><strong>RX FIFO Enabled</strong>&lt;br&gt;Set this field to 1 to enable the RX FIFO.&lt;br&gt;&lt;br&gt;0: RX FIFO disabled&lt;br&gt;1: RX FIFO enabled</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>20:16</td>
<td>rx_fifo_level</td>
<td>R/W</td>
<td>0</td>
<td><strong>RX FIFO Threshold Level</strong>&lt;br&gt;When the RX FIFO has more than this field, a DMA request is triggered, and the SPIXR_INT_FL.rx_level interrupt flag is set. Valid values are 0x00 to 0x1E.&lt;br&gt;&lt;br&gt;0x1F is not a valid value.</td>
</tr>
<tr>
<td>15</td>
<td>tx_dma_en</td>
<td>R/W</td>
<td>0</td>
<td><strong>TX DMA Enable</strong>&lt;br&gt;0: TX DMA is disabled. Any pending DMA requests are cleared.&lt;br&gt;1: TX DMA is enabled</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>13:8</td>
<td>tx_fifo_cnt</td>
<td>RO</td>
<td>0</td>
<td><strong>Number of Bytes in the TX FIFO</strong>&lt;br&gt;Read returns the number of bytes currently in the TX FIFO</td>
</tr>
</tbody>
</table>
### SPIXR DMA Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 7    | tx_fifo_clear    | WO     | 0     | Clear the TX FIFO  
Set this field to 1 to clear the TX FIFO and all TX FIFO related flags in the SPIXR_INT_FL register. When the TX FIFO is cleared, the SPIXR_INT_FL.tx_fifo_empty flag is set by hardware.  
1: Clear the TX FIFO and any pending TX FIFO flags in SPIXR_INT_FL. This should be done when the TX FIFO is inactive.  
Note: Writing a 0 has no effect. |
| 6    | tx_fifo_en       | R/W    | 0     | TX FIFO Enabled  
Set to 1 to enable the TX FIFO.  
0: TX FIFO disabled  
1: TX FIFO enabled |
| 5    | -                | R/W    | 0     | Reserved for Future Use  
Do not modify this field. |
| 4:0  | tx_fifo_level    | R/W    | 0x10  | TX FIFO Threshold Level  
When the TX FIFO has fewer than this field, a DMA request is triggered and the SPIXR_INT_FL.tx_level interrupt flag is set. |

For all read-only fields, writes have no effect.

### SPIXR Interrupt Status Flag Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>-</td>
<td>R/W1C</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
</tbody>
</table>
| 15   | rx_und          | R/W1C  | 0     | RX FIFO Underrun Flag  
Set when a read is attempted from an empty RX FIFO. |
| 14   | rx_ovr          | R/W1C  | 0     | RX FIFO Overrun Flag  
Set if SPI is in Slave Mode, and a write to a full RX FIFO is attempted. If the SPI is in Master Mode, this bit is not set as the SPI stalls the clock until data is read from the RX FIFO |
| 13   | tx_und          | R/W1C  | 0     | TX FIFO Underrun Flag  
Set if SPI is in Slave Mode, and a read from empty TX FIFO is attempted. If SPI is in Master Mode, this bit is not set as the SPI stalls the clock until data is written to the empty TX FIFO. |
| 12   | tx_ovr          | R/W1C  | 0     | TX FIFO Overrun Flag  
Set when a write is attempted to a full TX FIFO. |
| 11   | m_done          | R/W1C  | 0     | Master Data Transmission Done Flag  
Set if SPI is in Master Mode, and all transactions have completed. |
| 10   | -               | R/W1C  | 0     | Reserved for Future Use  
Do not modify this field. |
| 9    | abort           | R/W1C  | 0     | Slave Mode Transaction Abort Detected Flag  
Set if the SPI is in Slave Mode, and SS is deasserted before a complete character is received. |
## SPIXR Interrupt Status Flag Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>fault</td>
<td>R/W1C</td>
<td>0</td>
<td>Multi-Master Fault Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set if the SPI is in Master Mode, Multi-Master Mode is enabled, and a Slave</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Select input is asserted. A collision also sets this flag.</td>
</tr>
<tr>
<td>7:6</td>
<td>-</td>
<td>R/W1C</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>5</td>
<td>ssd</td>
<td>R/W1C</td>
<td>0</td>
<td>Slave Select Deasserted Flag</td>
</tr>
<tr>
<td>4</td>
<td>ssa</td>
<td>R/W1C</td>
<td>0</td>
<td>Slave Select Asserted Flag</td>
</tr>
<tr>
<td>3</td>
<td>rx_full</td>
<td>R/W1C</td>
<td>0</td>
<td>RX FIFO Full Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set when the RX FIFO is full.</td>
</tr>
<tr>
<td>2</td>
<td>rx_level</td>
<td>R/W1C</td>
<td>0</td>
<td>RX FIFO Threshold Level Crossed Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set when the RX FIFO exceeds the value in \textit{SPIXR_DMA.rx_fifo_level}.</td>
</tr>
<tr>
<td>1</td>
<td>tx_empty</td>
<td>R/W1C</td>
<td>1</td>
<td>TX FIFO Empty Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set when the TX FIFO is empty.</td>
</tr>
<tr>
<td>0</td>
<td>tx_level</td>
<td>R/W1C</td>
<td>0</td>
<td>TX FIFO Threshold Level Crossed Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set when the TX FIFO is less than the value in \textit{SPIXR_DMA.tx_fifo_level}.</td>
</tr>
</tbody>
</table>

### Table 8-33. SPIXR Interrupt Enable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>15</td>
<td>rx_und</td>
<td>R/W</td>
<td>0</td>
<td>RX FIFO Underrun Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td>14</td>
<td>rx_ovr</td>
<td>R/W</td>
<td>0</td>
<td>RX FIFO Overrun Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td>13</td>
<td>tx_und</td>
<td>R/W</td>
<td>0</td>
<td>TX FIFO Underrun Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td>12</td>
<td>tx_ovr</td>
<td>R/W</td>
<td>0</td>
<td>TX FIFO Overrun Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td>11</td>
<td>m_done</td>
<td>R/W</td>
<td>0</td>
<td>Master Data Transmission Done Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>9</td>
<td>abort</td>
<td>R/W</td>
<td>0</td>
<td>Slave Mode Transaction Abort Detected Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
</tbody>
</table>
### SPIXR Interrupt Enable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>fault</td>
<td>R/W</td>
<td>0</td>
<td>Multi-Master Fault Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td>7:6</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td>5</td>
<td>ssd</td>
<td>R/W</td>
<td>0</td>
<td>Slave Select Deasserted Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td>4</td>
<td>ssa</td>
<td>R/W</td>
<td>0</td>
<td>Slave Select Asserted Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td>3</td>
<td>rx_full</td>
<td>R/W</td>
<td>0</td>
<td>RX FIFO Full Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td>2</td>
<td>rx_level</td>
<td>R/W</td>
<td>0</td>
<td>RX FIFO Threshold Level Crossed Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>tx_empty</td>
<td>R/W</td>
<td>1</td>
<td>TX FIFO Empty Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td>0</td>
<td>tx_level</td>
<td>R/W</td>
<td>0</td>
<td>TX FIFO Threshold Level Crossed Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
</tbody>
</table>

Table 8-34. SPIXR Wakeup Flag Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:4</td>
<td>-</td>
<td>R/W1C</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>3</td>
<td>rx_full</td>
<td>R/W1C</td>
<td>0</td>
<td>Wake on RX FIFO Full Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If set, the RX FIFO Full condition caused the wake event.</td>
</tr>
<tr>
<td>2</td>
<td>rx_level</td>
<td>R/W1C</td>
<td>0</td>
<td>Wake on RX FIFO Threshold Level Crossed Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If set, the RX FIFO Threshold Level Crossed condition caused the wake event.</td>
</tr>
<tr>
<td>1</td>
<td>tx_empty</td>
<td>R/W1C</td>
<td>0</td>
<td>Wake on TX FIFO Empty Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If set, the TX FIFO empty condition caused the wake event.</td>
</tr>
<tr>
<td>0</td>
<td>tx_level</td>
<td>R/W1C</td>
<td>0</td>
<td>Wake on TX FIFO Threshold Level Crossed Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If set, the TX FIFO threshold level crossed caused the wake event.</td>
</tr>
</tbody>
</table>
### Table 8-35. SPIXR Wakeup Enable Register

<table>
<thead>
<tr>
<th>SPIXR Wakeup Enable Register</th>
<th>SPIXR_WAKE_EN</th>
<th>[0x002C]</th>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
|                              |               |         | 31:4 | -      | R/W    | 0     | Reserved for Future Use  
Do not modify this field.                                                      |
|                              |               |         | 3    | rx_full| R/W    | 0     | Wake on RX FIFO Full Enable  
Set to 1 to wake up the device when this RX FIFO is full.  
0: Wakeup Disabled for this condition.  
1: Wakeup Enabled for this condition. |
|                              |               |         | 2    | rx_level| R/W    | 0     | Wake on RX FIFO Threshold Level Crossed Enable  
Set to 1 to wake up the device when this RX FIFO is full.  
0: Wakeup Disabled for this condition.  
1: Wakeup Enabled for this condition. |
|                              |               |         | 1    | tx_empty| R/W    | 0     | Wake on TX FIFO Empty Enable  
Set to 1 to wake up the device when this TX FIFO is full.  
0: Wakeup Disabled for this condition.  
1: Wakeup Enabled for this condition. |
|                              |               |         | 0    | tx_level| R/W    | 0     | Wake on TX FIFO Threshold Level Crossed Enable  
Set to 1 to wake up the device when this TX FIFO is full.  
0: Wakeup Disabled for this condition.  
1: Wakeup Enabled for this condition. |

### Table 8-36. SPIXR Active Status Register

<table>
<thead>
<tr>
<th>SPIXR Active Status Register</th>
<th>SPIXR_STAT</th>
<th>[0x0030]</th>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
|                              |            |         | 31:1 | -      | RO     | 0     | Reserved for Future Use  
Do not modify this field.                                                      |
|                              |            |         | 0    | busy   | RO     | 0     | SPI Active Status  
This field returns the status of the SPIXR communications.  
Hardware sets and clears this field automatically when SPI communications are active or complete.  
0: SPI is not active.  
1: SPI is active.  
Set when transmit starts. |

### Table 8-37. SPIXR External Memory Control Register

<table>
<thead>
<tr>
<th>SPIXR External Memory Control Register</th>
<th>SPIXR_XMEM_CTRL</th>
<th>[0x0034]</th>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
|                                        |                 |         | 31   | xmem_en   | R/W    | 0     | Enable External Memory  
0: XMEM disabled  
1: XMEM enabled |
|                                        |                 |         | 30:24| -         | R/W    | 0     | Reserved for Future Use  
Do not modify this field.                                                      |
|                                        |                 |         | 23:16|xmem_dclks| R/W    | 0     | Number of dummy characters between address phase and read data from the  
external memory.  
0: no delay between address and read data  
1:255 delay number of characters |
### SPIXR External Memory Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:8</td>
<td>xmem_wr_cmd</td>
<td>R/W</td>
<td>0</td>
<td>Write command to be received at the external memory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Vendor specific value</td>
</tr>
<tr>
<td>7:0</td>
<td>xmem_rd_cmd</td>
<td>R/W</td>
<td>0</td>
<td>Read command to be received at the external memory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Vendor specific value</td>
</tr>
</tbody>
</table>

#### 8.4 SPIXR Cache Controller (SRCC)

The SPIXR Cache Controller is an AHB block that has multiple interfaces. The address and data interface is connected to the AHB and the SRCC register interface is connected via the APB.

The SRCC is a 16KB 2-way set-associative cache. It operates with the LRU replacement policy and has write-through implementation used for caching instructions and data from an external SPI-XiP RAM device. The SRCC includes tag RAM, cache RAM and a line fill buffer as shown in Figure 4-6: MAX32665—MAX32668 Cache Controllers Control. Write allocate and critical word first are options controlled by the application. Each cache line is 256-bits wide with the lower 5-bits of the address used as the cache line index. The SRCC uses tag cache RAM with 8-bits of the address index and a 5-bit line offset to access the tag cache RAM. 16-bits of the address are stored in tag RAM for hit/miss checking enabling the SRCC to access up to 512MB of external memory. The SRCC interfaces to the address range of 0x8000 0000 to 0x9FFF FFFF for a maximum of 512MB external.

#### 8.4.1 Features

- 2-way set associative, LRU (Least-Recently Used) replacement policy
- Write-no-allocate with option to Write-allocate
- Write-through
- Read critical word first and streaming
- 512MB addressable range
- 16KB size

#### 8.4.2 Enabling the SRCC

Enable the SRCC as follows:

1. Set the `GCR_SCON.dcache_dis` field to 0.
2. Set the `SRCC_CACHE_CTRL.enable` field to 1.

Once enabled, the cache is empty and begins filling when a read from or write to (if write allocate is enabled) the external memory is performed.

After a Power-On-Reset event, the cache tag RAM is cleared by hardware ensuring that no corrupted data is accessed from the initial cache read.

#### 8.4.3 Disabling the SRCC

Disabling the SRCC cache automatically invalidates the cache contents. All access to the external memory while the SRCC cache is disabled are performed using the Line Buffer.

Disable the SRCC by setting `SRCC_CACHE_CTRL.enable` to 0.

The SRCC cache and Line Buffer can both be bypassed by setting `GCR_SCON.dcache_dis` to 1. Bypassing the SRCC enables direct access to the external memory from the application firmware.
8.4.4 SRCC Registers

See Table 3-1: APB Peripheral Base Address Map for the SRCC Peripheral Base Address.

Table 8-38: External Memory Cache Controller Register Addresses and Descriptions

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register Name</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>SRCC_CACHE_ID</td>
<td>RO</td>
<td>Cache ID Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>SRCC_MEM_SIZE</td>
<td>RO</td>
<td>Cache Memory Size Register</td>
</tr>
<tr>
<td>[0x0100]</td>
<td>SRCC_CACHE_CTRL</td>
<td>R/W</td>
<td>Cache Control Register</td>
</tr>
<tr>
<td>[0x0700]</td>
<td>SRCC_INVALIDATE</td>
<td>WO</td>
<td>Invalidate Register</td>
</tr>
</tbody>
</table>

8.4.5 SRCC Register Details

Table 8-39: SRCC Cache ID Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:16  | -           | RO     | -     | Reserved for Future Use  
|        |             |        |       | Do not modify this field.                         |
| 15:10  | cchid       | RO     | -     | Cache ID  
|        |             |        |       | Returns the Cache ID for this Cache instance.    |
| 9:6    | partnum     | RO     | -     | Cache Part Number  
|        |             |        |       | Returns the part number indicator for this Cache instance. |
| 5:0    | relnum      | RO     | -     | Cache Release Number  
|        |             |        |       | Returns the release number for this Cache instance. |

Table 8-40: SRCC Memory Size Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:16  | memsz | RO     | -     | Addressable Memory Size  
|        |       |        |       | Indicates the size of addressable memory by this cache controller instance in 128KB units. |
| 15:0   | cchsz | RO     | -     | Cache Size  
|        |       |        |       | Returns the size of the cache RAM memory in 1KB units.  
|        |       |        |       | 16: 16KB Cache RAM |

Table 8-41: SRCC Cache Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:17  | -     | R/W    | -     | Reserved for Future Use  
|        |       |        |       | Do not modify this field.                         |
| 16     | ready | RO     | -     | Ready  
|        |       |        |       | This field is cleared by hardware anytime the cache as a whole is invalidated (including a Power-On Reset event). Hardware automatically sets this field to 1 when the invalidate operation is complete and the cache is ready.  
|        |       |        |       | 0: Cache Invalidate in process.  
|        |       |        |       | 1: Cache is ready.  
<p>|        |       |        |       | Note: While this field reads 0, the cache is bypassed and reads come directly from the line fill buffer. |</p>
<table>
<thead>
<tr>
<th>bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:3</td>
<td>-</td>
<td>R/W</td>
<td></td>
<td><strong>Reserved for Future Use</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>2</td>
<td>cwfst_dis</td>
<td>R/W</td>
<td>0</td>
<td><strong>Critical Word First (CWF) Disable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting this field to 1 disables Critical Word First operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When CWF is disabled, the cache fills the cache line before sending the data to the Arm Cortex core. When CWF is enabled, any data fetch that results in a cache miss immediately sends the data read to the Arm Cortex core prior to filling the cache line.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Enable Critical Word First.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Critical Word First Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> This field is only writable when the EMCC is disabled (SRCC_CACHE_CTRL.enable = 0).</td>
</tr>
<tr>
<td>1</td>
<td>write_alloc</td>
<td>R/W</td>
<td>0</td>
<td><strong>Write Allocate Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set this field to enable write allocate for the cache.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When this is enabled, writes to the memory update the external memory and the cache line associated with the write is filled from the external memory. Disabling write allocate, default mode, performs a write to the external memory on any write operation, but the associated cache line is not refilled. When disabled, writes to successive memory locations are more efficient.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Write allocate disabled (default)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Write allocate enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> The EMCC is a write-through cache resulting in any write to the external memory performing an immediate write to the external device.</td>
</tr>
<tr>
<td>0</td>
<td>enable</td>
<td>R/W</td>
<td>0</td>
<td><strong>Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set this field to 1 to enable the cache.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting this field to 0 automatically invalidates the cache contents.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When this cache is disabled, reads are handled by the line fill buffer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable Cache</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable Cache</td>
</tr>
</tbody>
</table>

**Table 8-42: SRCC Invalidate Register**

<table>
<thead>
<tr>
<th>bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>-</td>
<td>WO</td>
<td></td>
<td><strong>Invalidate</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Any write to this register of any value invalidates the cache.</td>
</tr>
</tbody>
</table>
8.5 Secure Digital Host Controller

The Secure Digital Host Controller (SDHC) provides an interface between the AHB and Embedded MultiMediaCards (e.MMCs), Secure Digital I/O (SDIO) cards, Standard Capacity SD Memory Cards and High-Capacity SD Memory Cards. The SDHC handles the SDIO/SD protocol at the transmission level, packing data, adding cyclic redundancy check (CRC), Start/End bit, and checking for transaction format correctness. Details of the SD communication and protocol are not part of the scope of this document. The MAX32665—MAX32668 SDHC only supports a single SD card.

SD memory card and SDIO card specifications are available at https://www.sdcard.org.

The e.MMC specifications are available from JEDEC at http://www.jedec.org.

Compliance

- SD Host Controller Standard Specification Version 3.00
- SDIO Card Specification Version 3.0
- SD Memory Card Specification Version 3.01
- SD Memory Card Security Specification version 1.01
- e.MMC Specification version 4.51

SD/SDIO Card Interface

- Supports SDR50 with SDHC clock of up to 48MHz (24MB/sec)
- Supports DDR50 with SDHC clock of up to 24MHz (24MB/sec)
- Designed to work with I/O cards, Read-Only cards, and Read/Write cards
- 1-bit and 4-bit data transfers in SD modes and SPI mode
- Double buffer for transfers configurable from 512B to 1KB.
- Auto Command (AutoCMD12 or AutoCMD23) support
- Multi-block transfers
- Variable-length data transfers
- Default and high-speed mode transfers
- Card insertion/removal events
- Read Wait Control, Suspend/Resume operation
- CRC7 for command and CRC16 for data integrity
- Single Operation DMA (SDMA) for data transfer
- Advanced DMA (ADMA) support
Figure 8-7: SDHC Block Diagram

8.5.1 Instances

The SDHC pin mapping for the SD Host Controller Standard Specification Version 3.0 are shown in Table 8-43, below.

Table 8-43: MAX32665—MAX32668 SDHC Alternate Function Mapping to SDHC Specification Pin Names

<table>
<thead>
<tr>
<th>Alternate Function</th>
<th>Alternate Function Number</th>
<th>Pin Name</th>
<th>SDHC Specification Pin Name</th>
<th>Direction</th>
<th>Signal Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDHC_CDN</td>
<td>AF1</td>
<td>P1.7</td>
<td>SDCD#</td>
<td>I</td>
<td>Card present, active low.</td>
</tr>
<tr>
<td>SDHC_CLK</td>
<td>AF1</td>
<td>P1.3</td>
<td>SDCLK</td>
<td>O</td>
<td>SD clock signal.</td>
</tr>
<tr>
<td>SDHC_WP</td>
<td>AF1</td>
<td>P1.6</td>
<td>SDWP</td>
<td>I</td>
<td>Write protect signal, active high.</td>
</tr>
<tr>
<td>SDHC_CMD</td>
<td>AF1</td>
<td>P1.1</td>
<td>CMD</td>
<td>I/O</td>
<td>SD bus command signal.</td>
</tr>
<tr>
<td>SDHC_DAT0</td>
<td>AF1</td>
<td>P1.2</td>
<td>DAT[0]</td>
<td>I/O</td>
<td>SD data bus bit 0.</td>
</tr>
<tr>
<td>SDHC_DAT1</td>
<td>AF1</td>
<td>P1.4</td>
<td>DAT[1]</td>
<td>I/O</td>
<td>SD data bus bit 1.</td>
</tr>
<tr>
<td>SDHC_DAT2</td>
<td>AF1</td>
<td>P1.5</td>
<td>DAT[2]</td>
<td>I/O</td>
<td>SD data bus bit 2.</td>
</tr>
</tbody>
</table>
### Alternate Function  
<table>
<thead>
<tr>
<th>Alternate Function</th>
<th>Alternate Function Number</th>
<th>Pin Name</th>
<th>SDHC Specification Pin Name</th>
<th>Direction</th>
<th>Signal Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDHC_DAT3</td>
<td>AF1</td>
<td>P1.0</td>
<td>DAT[3]</td>
<td>I/O</td>
<td>SD data bus bit 3.</td>
</tr>
</tbody>
</table>

For configuration of the GPIO for SDHC peripheral usage see the General-Purpose I/O and Alternate Function Pins chapter.

### 8.5.2 SDHC Peripheral Clock Selection

The input clock to the SDHC peripheral is driven by the high speed system oscillator always, 96MHz. This 96MHz input clock is either divided by 2 (default) or by 4 to drive the SDHC peripheral. Set the SDHC peripheral clock divisor using the `GCR_PCLK_DIV` bit as shown.

**Equation 8-1: SDHC Peripheral Clock**

\[
f_{SDHC_{-}CLK} = \frac{96MHz}{2^{GCR_{-}PCLK_{DIV}_{sdhcfreq}}}
\]

### 8.5.3 Usage

Communication over the SD bus is based on command and data bit streams/blocks that are initiated by a start bit and terminated by a stop bit.

- **Command**: A command is a token that starts an operation and is sent by the SDHC to the card in the embedded card slot. A command is transferred serially using the `SDHC_CMD` pin.
- **Response**: A response is a token sent from the card to the SDHC in response to a previously received command and is transferred serially using the `SDHC_CMD` pin.
- **Data**: You can transfer data from the card to the SDHC or vice versa using the `SDHC_DAT[3:0]` pins.

*Figure 8-8, Figure 8-9, and Figure 8-10* show the basic types of SD operations as described in the Physical Layer Simplified Specification Version 6.00 from the SD Card Association.

*Figure 8-8: SD Bus Protocol - No Response and No Data Operations*
8.5.4 SD Command Generation

Table 8-44 shows the registers required for three transaction types: SDMA generated transactions, ADMA generated transactions, and CPU transactions (includes data transfers and Non-DAT transfers). When initiating a transaction, you should program the registers sequentially starting with the SDHC_SDMA register and finishing with the SDHC_CMD register. When the upper byte of the SDHC_CMD register is written, it triggers the SDHC to issue the SD command.

<table>
<thead>
<tr>
<th>Register</th>
<th>SDMA Command</th>
<th>ADMA Command</th>
<th>CPU Data Transfer</th>
<th>Non-DAT (No Data) Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDMA System Address / Argument 2 SDHC_SDMA</td>
<td>Yes/No</td>
<td>No/Auto CMD23</td>
<td>No/AutoCMD23</td>
<td>No/No</td>
</tr>
<tr>
<td>Block Size SDHC_BLK_SIZE</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No (Protected)</td>
</tr>
<tr>
<td>Block Count SDHC_BLK_CNT</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No (Protected)</td>
</tr>
</tbody>
</table>
### SDHC Registers

See Table 3-1: APB Peripheral Base Address Map for the SDHC Peripheral Base Address

**Table 8-45: SDHC Register Offsets, Names and Descriptions**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>SDHC_SDMA</td>
<td>SDMA System Address / Argument 2</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>SDHC_BLK_SIZE</td>
<td>Block Size register</td>
</tr>
<tr>
<td>[0x0006]</td>
<td>SDHC_BLK_CNT</td>
<td>Block Count register</td>
</tr>
<tr>
<td>[0x0008]</td>
<td>SDHC_ARG_1</td>
<td>Argument 1 register</td>
</tr>
<tr>
<td>[0x000C]</td>
<td>SDHC_TRANS</td>
<td>Transfer Mode register</td>
</tr>
<tr>
<td>[0x000E]</td>
<td>SDHC_CMD</td>
<td>Command register</td>
</tr>
<tr>
<td>[0x0010]</td>
<td>SDHC_RESP_0</td>
<td>Response register 0</td>
</tr>
<tr>
<td>[0x0012]</td>
<td>SDHC_RESP_1</td>
<td>Response register 1</td>
</tr>
<tr>
<td>[0x0014]</td>
<td>SDHC_RESP_2</td>
<td>Response register 2</td>
</tr>
<tr>
<td>[0x0016]</td>
<td>SDHC_RESP_3</td>
<td>Response register 3</td>
</tr>
<tr>
<td>[0x0018]</td>
<td>SDHC_RESP_4</td>
<td>Response register 4</td>
</tr>
<tr>
<td>[0x001A]</td>
<td>SDHC_RESP_5</td>
<td>Response register 5</td>
</tr>
<tr>
<td>[0x001C]</td>
<td>SDHC_RESP_6</td>
<td>Response register 6</td>
</tr>
<tr>
<td>[0x001E]</td>
<td>SDHC_RESP_7</td>
<td>Response register 7</td>
</tr>
<tr>
<td>[0x0020]</td>
<td>SDHC_BUFFER</td>
<td>Buffer Data Port register</td>
</tr>
<tr>
<td>[0x0024]</td>
<td>SDHC_PRESENT</td>
<td>Present State register</td>
</tr>
<tr>
<td>[0x0028]</td>
<td>SDHC_HOST_CN_1</td>
<td>Host Control 1 register</td>
</tr>
<tr>
<td>[0x0029]</td>
<td>SDHC_PWR</td>
<td>Power Control register</td>
</tr>
<tr>
<td>[0x002A]</td>
<td>SDHC_BLK_GAP</td>
<td>Block Gap Control register</td>
</tr>
<tr>
<td>[0x002B]</td>
<td>SDHC_WAKEUP</td>
<td>Wakeup Control register</td>
</tr>
<tr>
<td>[0x002C]</td>
<td>SDHC_CLK_CN</td>
<td>Clock Control register</td>
</tr>
<tr>
<td>[0x002E]</td>
<td>SDHC_TO</td>
<td>Timeout Control register</td>
</tr>
<tr>
<td>[0x002F]</td>
<td>SDHC_SW_RESET</td>
<td>Software Reset register</td>
</tr>
<tr>
<td>[0x0030]</td>
<td>SDHC_INT_STAT</td>
<td>Normal Interrupt Status register</td>
</tr>
<tr>
<td>[0x0032]</td>
<td>SDHC_ER_INT_STAT</td>
<td>Error Interrupt Status register</td>
</tr>
<tr>
<td>[0x0034]</td>
<td>SDHC_INT_EN</td>
<td>Normal Interrupt Status Enable register</td>
</tr>
<tr>
<td>[0x0036]</td>
<td>SDHC_ER_INT_EN</td>
<td>Error Interrupt Status Enable register</td>
</tr>
<tr>
<td>[0x0038]</td>
<td>SDHC_INT_SIGNAL</td>
<td>Normal Interrupt Signal Enable register</td>
</tr>
<tr>
<td>[0x003A]</td>
<td>SDHC_ER_INT_SIGNAL</td>
<td>Error Interrupt Signal Enable register</td>
</tr>
<tr>
<td>[0x003C]</td>
<td>SDHC_AUTO_CMD_ER</td>
<td>Auto CMD Error Status register</td>
</tr>
<tr>
<td>[0x003E]</td>
<td>SDHC_HOST_CN_2</td>
<td>Host Control 2 register</td>
</tr>
<tr>
<td>[0x0040]</td>
<td>SDHC_CFG_0</td>
<td>Capabilities register 0</td>
</tr>
<tr>
<td>[0x0044]</td>
<td>SDHC_CFG_1</td>
<td>Capabilities register 1</td>
</tr>
<tr>
<td>[0x0048]</td>
<td>SDHC_MAX_CURR_CFG</td>
<td>Maximum Current Capabilities register</td>
</tr>
</tbody>
</table>

8.5.5 SDHC Registers
### SDHC Register Details

#### Table 8-46: SDHC SDMA System Address / Argument Register

<table>
<thead>
<tr>
<th>SDMA System Address / Argument 2 Register</th>
<th>SDHC_SDMA</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Offset</strong></td>
<td><strong>Register Name</strong></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td>[0x0050]</td>
<td><strong>SDHC_FORCE_CMD</strong></td>
<td>Force Event Register for Auto CMD Error Status</td>
</tr>
<tr>
<td>[0x0052]</td>
<td><strong>SDHC_FORCE_EVENT_INT_STAT</strong></td>
<td>Force Event Register for Error Interrupt Status</td>
</tr>
<tr>
<td>[0x0054]</td>
<td><strong>SDHC_ADMA_ER</strong></td>
<td>ADMA Error Status register</td>
</tr>
<tr>
<td>[0x0058]</td>
<td><strong>SDHC_ADMA_ADDR_0</strong></td>
<td>ADMA System Address register 0</td>
</tr>
<tr>
<td>[0x005C]</td>
<td><strong>SDHC_ADMA_ADDR_1</strong></td>
<td>ADMA System Address register 1</td>
</tr>
<tr>
<td>[0x0060]</td>
<td><strong>SDHC_PRESET_0</strong></td>
<td>Preset Value for Initialization</td>
</tr>
<tr>
<td>[0x0062]</td>
<td><strong>SDHC_PRESET_1</strong></td>
<td>Preset Value for Default Speed</td>
</tr>
<tr>
<td>[0x0064]</td>
<td><strong>SDHC_PRESET_2</strong></td>
<td>Preset Value for High Speed</td>
</tr>
<tr>
<td>[0x0066]</td>
<td><strong>SDHC_PRESET_3</strong></td>
<td>Preset Value for SDR12</td>
</tr>
<tr>
<td>[0x0068]</td>
<td><strong>SDHC_PRESET_4</strong></td>
<td>Preset Value for SDR25</td>
</tr>
<tr>
<td>[0x006A]</td>
<td><strong>SDHC_PRESET_5</strong></td>
<td>Preset Value for SDR50</td>
</tr>
<tr>
<td>[0x006C]</td>
<td><strong>SDHC_PRESET_6</strong></td>
<td>Preset Value for SDR104</td>
</tr>
<tr>
<td>[0x006E]</td>
<td><strong>SDHC_PRESET_7</strong></td>
<td>Preset Value for DDR50</td>
</tr>
<tr>
<td>[0x00FC]</td>
<td><strong>SDHC_SLOT_INT</strong></td>
<td>Slot Interrupt Status register</td>
</tr>
<tr>
<td>[0x00FE]</td>
<td><strong>SDHC_HOST_CN_VER</strong></td>
<td>Host Controller Version register</td>
</tr>
</tbody>
</table>

**8.5.6 SDHC Register Details**

- **SDMA System Address**
  - This register is the address of the buffer used for a SDMA transfer. You must set this register to a valid data buffer address prior to starting an SDMA transfer. A SDHC DMA interrupt (SDHC_INT_STAT.dma = 1) is generated if the total size of the SDMA transfer exceeds the Host SDMA Buffer Size (SDHC_BLK_SIZE.host_buf). The card driver must update the SDMA System Address (SDHC_SDMA) with the address of the next data to transfer and clear the SDHC DMA interrupt flag prior to the transfer resuming.
  - When the SDMA transfer is complete, this register contains the address of the next contiguous data address.
  - When resuming a SDMA transfer, using the Resume command or by setting the SDHC_BLK_GAP.gap_cont bit to 1, the SDHC resumes using the address in this register for the data to transfer.
  - Reading this register during a SDMA transfer might return an invalid value unless the transfer is paused as the result of a SDHC DMA interrupt. This field is not used for ADMA transfers.

- **Argument 2**
  - This register is used with Auto CMD23 to set a 32-bit block count value to the argument of CMD23 while executing Auto CMD23.
  - If Auto CMD23 is used with ADMA, then the full 32-bit block count value is used. If Auto CMD23 is used without AMDA, the available block count value is limited by the SDHC_BLK_GAP register to 65,535 blocks.
### Table 8-47: SDHC SDMA Block Size Register

<table>
<thead>
<tr>
<th>SDMA Block Size Register</th>
<th>SDHC_BLK_SIZE</th>
<th>[0x0004]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 31:15                    | -            | R/W     | 0     | Reserved for Future Use  
Do not modify this field. |
| 14:12                    | host_buf     | R/W     | 0     | Host SDMA Buffer Size  
This field specifies the size of the contiguous buffer in the system memory for SDMA transfers. SDMA transfers larger than this buffer generates a SDHC DMA interrupt (SDHC_INT_STAT.dma) when the transfer reaches the host_buf size boundary. The SDMA transfer pauses until the card driver updates the SDMA System Address (SDHC_SDMA) register with the next buffer address to transfer and clears the SDHC DMA interrupt flag. When the SDMA transfer is complete, a SDHC transfer complete interrupt (SDHC_INT_STAT.trans_comp = 1) is generated. The SDHC DMA interrupt flag is not set when the SDMA transfer completes. |
| 11:0                     | trans        | R/W     | 0x0200 | Data Transfer Block Size  
Sets the block size of data transfers for CMD17, CMD18, CMD24, CMD25, and CMD53. You can set values ranging from 1 up to the maximum buffer size. Setting this field to 0 indicates there is no data to transfer. During a transfer, reading this field might return an invalid value, and writes to this field are ignored. |

<table>
<thead>
<tr>
<th>host_buf Value</th>
<th>Host SDMA Buffer Size (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>4</td>
</tr>
<tr>
<td>0001</td>
<td>8</td>
</tr>
<tr>
<td>0010</td>
<td>16</td>
</tr>
<tr>
<td>0011</td>
<td>32</td>
</tr>
<tr>
<td>0100</td>
<td>64</td>
</tr>
<tr>
<td>0101</td>
<td>128</td>
</tr>
<tr>
<td>0110</td>
<td>256</td>
</tr>
<tr>
<td>0111</td>
<td>512</td>
</tr>
</tbody>
</table>

Note: This field is used for SDMA transfers only.

<table>
<thead>
<tr>
<th>trans Value</th>
<th>Block Size in Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>00004</td>
<td>4</td>
</tr>
<tr>
<td>00003</td>
<td>3</td>
</tr>
<tr>
<td>00002</td>
<td>2</td>
</tr>
<tr>
<td>00001</td>
<td>1</td>
</tr>
<tr>
<td>00000</td>
<td>No data transfer</td>
</tr>
</tbody>
</table>
Table 8-48: SDHC SDMA Block Count Register

<table>
<thead>
<tr>
<th>SDMA Block Count Register</th>
<th>SDHC_BLK_CNT</th>
<th>[0x0006]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>31:16</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:0</td>
<td>trans</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trans Value</td>
<td>Block Count</td>
<td></td>
</tr>
<tr>
<td>0xFFFF</td>
<td>65,535</td>
<td></td>
</tr>
<tr>
<td>0xFFFE</td>
<td>65,534</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>0x0002</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>0x0001</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0x0000</td>
<td>Stop count</td>
<td></td>
</tr>
<tr>
<td></td>
<td>or no block transfer</td>
<td></td>
</tr>
</tbody>
</table>

Table 8-49: SDHC SDMA Argument 1 Register

<table>
<thead>
<tr>
<th>SDMA Argument 1 Register</th>
<th>SDHC_ARG_1</th>
<th>[0x0008]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>cmd</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8-50: SDHC SDMA Transfer Mode Register

<table>
<thead>
<tr>
<th>SDMA Transfer Mode Register</th>
<th>SDHC_TRANS</th>
<th>[0x000C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>31:6</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>multi</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Maxim Integrated
<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>read_write</td>
<td>R/W</td>
<td>0</td>
<td><strong>Data Transfer Direction Select</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sets the direction for DAT line data transfers. Set to 1 to transfer data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>from the SD card to the SDHC (Read). For all other commands, set this bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>to 0 (Write). 1: Read (from card to host) 0: Write (from host to card)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Auto CMD Enable / Function Selection</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00: Auto Command Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01: Auto CMD12 Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b10: Auto CMD23 Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b11: Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td>auto_cmd_en</td>
<td>R/W</td>
<td>0</td>
<td><strong>Auto CMD12 Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When auto_cmd_en is set to 1, the SDHC issues CMD12 automatically after</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>completion of the last block transfer. If an error occurs from Auto CMD12,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>then the error is saved to the SDHC_AUTO_CMD_ER register.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note: Do not set to 1 if an Auto CMD12 is not required.</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Auto CMD23 Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When this bit field is set to 0b10, the Host Controller issues a CMD23</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>automatically before issuing the command specified in the SDHC_CMD (</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Command) register. The following conditions are required to use Auto CMD23:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Auto CMD23 support (Host Controller Version is 3.00 or later)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• A memory card that supports CMD23 (SCR[33] = 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• If using DMA, ADMA mode only</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Only when CMD18 or CMD25 is issued</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>You can use Auto CMD23 with or without ADMA. By writing to the Command</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>register, the SDHC issues a CMD23 first, and then issues the command</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>specified by the Command Index (SDHC_CMD_idx) in the Command register. If</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>response errors are detected from CMD23, then the second command is not</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>issued. A CMD23 error is indicated in the Auto CMD Error Status regis</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ter (SDHC_AUTO_CMD_ER). The 32-bit block count value for CMD23 is set to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the SDMA System Address / Argument 2 register (SDHC SDMA).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note: The SDHC does not check the command index.</strong></td>
</tr>
<tr>
<td></td>
<td>blk_cnt_en</td>
<td>R/W</td>
<td>0</td>
<td><strong>Block Count Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set to enable the Block Count register (SDHC_BLK_CNT) for multiple block</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>transfers. When this bit is 0, the Block Count register (SDHC_BLK_CNT) is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>disabled, which is useful if executing an infinite transfer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable SDHC_BLK_CNT register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable SDHC_BLK_CNT register</td>
</tr>
<tr>
<td></td>
<td>dma_en</td>
<td>R/W</td>
<td>0</td>
<td><strong>DMA Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Enables DMA functionality per the Capabilities register.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If this bit is set to 1, a DMA operation begins when the card driver</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>writes to the upper byte of the Command register (SDHC_CMD).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: DMA mode is enabled as specified in the SDHC_HOST_CN_1.dma_select field</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: DMA mode disabled.</td>
</tr>
</tbody>
</table>
Table 8-51: Summary of how register settings determine type of data transfer

<table>
<thead>
<tr>
<th>Multi/Single Block Select</th>
<th>Block Count Enable</th>
<th>Block Count</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDHC_TRANS.multi</td>
<td>SDHC_TRANS.blk_cnt_en</td>
<td>SDHC_BLK_CNT.trans</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>N.A.</td>
<td>Single transfer</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>Infinite transfer</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>≠0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>Multiple transfer</td>
</tr>
</tbody>
</table>

Table 8-52: SDHC Command Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:14</td>
<td>-</td>
<td>R/W</td>
<td>NA</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>13:8</td>
<td>idx</td>
<td>R/W</td>
<td>0</td>
<td>Command Index</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Valid command number (CMD0-63, ACMD0-63) per the SD Physical Specification and SDIO Card Specification.</td>
</tr>
<tr>
<td>7:6</td>
<td>type</td>
<td>R/W</td>
<td>0</td>
<td>Command Type</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The following table lists the values for this field, the type of command, and provides notes about what the command type is typically used for:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><img src="image" alt="Command Type Table" /></td>
</tr>
<tr>
<td>5</td>
<td>data_pres_sel</td>
<td>R/W</td>
<td>0</td>
<td>Data Present Select</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Set to indicate data is present and transferable using the DAT line.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Commands that only use the CMD line (for example, CMD52), commands with no data transfer but are using the busy signal on SDHC_DAT[0], or a Resume command.</td>
</tr>
<tr>
<td>4</td>
<td>idx_chk_en</td>
<td>R/W</td>
<td>0</td>
<td>Command Index Check Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: SDHC checks the index field in the response and sets a Command Index Error if it does not match the value in the SDHC_CMD.idx field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Index of response is not checked.</td>
</tr>
<tr>
<td>3</td>
<td>crc_chk_en</td>
<td>R/W</td>
<td>0</td>
<td>Command CRC Check Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: SDHC verifies the CRC field in the response, and if an error is detected, it is reported as a Command CRC Error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: CRC not checked by hardware.</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>1:0</td>
<td>resp_type</td>
<td>R/W</td>
<td>0</td>
<td>Response Type Select</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00: No Response</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01: Response Length 136</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b10: Response Length 48</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b11: Response Length 48, and check if busy after response</td>
</tr>
</tbody>
</table>
Table 8-53: Relationship between Parameters and the Name of Response Type

<table>
<thead>
<tr>
<th>Response Type</th>
<th>Index Check Enable</th>
<th>CRC Check Enable</th>
<th>Name of Response Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDHC_CMD.resp_type</td>
<td>SDHC_CMD.idx_chk_en</td>
<td>SDHC_CMD.crc_chk_en</td>
<td></td>
</tr>
<tr>
<td>0b00</td>
<td>0</td>
<td>0</td>
<td>No Response</td>
</tr>
<tr>
<td>0b01</td>
<td>0</td>
<td>1</td>
<td>R2</td>
</tr>
<tr>
<td>0b10</td>
<td>0</td>
<td>0</td>
<td>R3, R4</td>
</tr>
<tr>
<td>0b10</td>
<td>1</td>
<td>1</td>
<td>R1, R5, R6, R7</td>
</tr>
<tr>
<td>0b11</td>
<td>1</td>
<td>1</td>
<td>R1b, R5b</td>
</tr>
</tbody>
</table>

Table 8-54: SDHC Response 0 Register

<table>
<thead>
<tr>
<th>Response 0 Register</th>
<th>SDHC_RESP_0</th>
<th>[0x0010]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>15:0</td>
<td>cmd_resp</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8-55: SDHC Response 1 Register

<table>
<thead>
<tr>
<th>Response 1 Register</th>
<th>SDHC_RESP_1</th>
<th>[0x0012]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>15:0</td>
<td>cmd.resp</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8-56: SDHC Response 2 Register

<table>
<thead>
<tr>
<th>Response 2 Register</th>
<th>SDHC_RESP_2</th>
<th>[0x0014]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>15:0</td>
<td>cmd.resp</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 8-57: SDHC Response 3 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:0</td>
<td>cmd_resp</td>
<td>RO</td>
<td>0</td>
<td>Response Register 3 Response 7 to Response 0 registers are referenced as a contiguous, single register in the SD Host Controller Spec V3.0. Table 8-62 shows the mapping from the Response Registers to the SD Host Controller Standard Specification REP[127:0] notation for the MAX32665—MAX32668. Table 8-63 shows the SD types of response mapped to the MAX32665—MAX32668 Response registers.</td>
</tr>
</tbody>
</table>

### Table 8-58: SDHC Response 4 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:0</td>
<td>cmd_resp</td>
<td>RO</td>
<td>0</td>
<td>Response Register 4 Response 7 to Response 0 registers are referenced as a contiguous, single register in the SD Host Controller Spec V3.0. Table 8-62 shows the mapping from the Response Registers to the SD Host Controller Standard Specification REP[127:0] notation for the MAX32665—MAX32668. Table 8-63 shows the SD types of response mapped to the MAX32665—MAX32668 Response registers.</td>
</tr>
</tbody>
</table>

### Table 8-59: SDHC Response 5 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:0</td>
<td>cmd_resp</td>
<td>RO</td>
<td>0</td>
<td>Response Register 5 Response 7 to Response 0 registers are referenced as a contiguous, single register in the SD Host Controller Spec V3.0. Table 8-62 shows the mapping from the Response Registers to the SD Host Controller Standard Specification REP[127:0] notation for the MAX32665—MAX32668. Table 8-63 shows the SD types of response mapped to the MAX32665—MAX32668 Response registers.</td>
</tr>
</tbody>
</table>

### Table 8-60: SDHC Response 6 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:0</td>
<td>cmd_resp</td>
<td>RO</td>
<td>0</td>
<td>Response Register 6 Response 7 to Response 0 registers are referenced as a contiguous, single register in the SD Host Controller Spec V3.0. Table 8-62 shows the mapping from the Response Registers to the SD Host Controller Standard Specification REP[127:0] notation for the MAX32665—MAX32668. Table 8-63 shows the SD types of response mapped to the MAX32665—MAX32668 Response registers.</td>
</tr>
</tbody>
</table>
Table 8-61: SDHC Response 7 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 15:0 | cmd_resp | RO    | 0     | Response Register 7  
Response 7 to Response 0 registers are referenced as a contiguous, single register in the SD Host Controller Spec V3.0. Table 8-62 shows the mapping from the Response Registers to the SD Host Controller Standard Specification REP[127:0] notation for the MAX32665—MAX32668. Table 8-63 shows the SD types of response mapped to the MAX32665—MAX32668 Response registers. |

Table 8-62: SDHC Response Register Mapping to SD Host Controller Response Register Convention

<table>
<thead>
<tr>
<th>Register</th>
<th>Register Name</th>
<th>Register Offset</th>
<th>SDHC REP[] Bit Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDHC_RESP_0</td>
<td>Response 0</td>
<td>0x10</td>
<td>REP[15:0]</td>
</tr>
<tr>
<td>SDHC_RESP_1</td>
<td>Response 1</td>
<td>0x12</td>
<td>REP[31:16]</td>
</tr>
<tr>
<td>SDHC_RESP_2</td>
<td>Response 2</td>
<td>0x14</td>
<td>REP[47:32]</td>
</tr>
<tr>
<td>SDHC_RESP_3</td>
<td>Response 3</td>
<td>0x16</td>
<td>REP[63:48]</td>
</tr>
<tr>
<td>SDHC_RESP_4</td>
<td>Response 4</td>
<td>0x18</td>
<td>REP[79:64]</td>
</tr>
<tr>
<td>SDHC_RESP_5</td>
<td>Response 5</td>
<td>0x1A</td>
<td>REP[95:80]</td>
</tr>
<tr>
<td>SDHC_RESP_6</td>
<td>Response 6</td>
<td>0x1C</td>
<td>REP[111:96]</td>
</tr>
<tr>
<td>SDHC_RESP_7</td>
<td>Response 7</td>
<td>0x1E</td>
<td>REP[127:112]</td>
</tr>
</tbody>
</table>

Table 8-63: Kind of SD Card Response Mapping to SDHC Response Registers

<table>
<thead>
<tr>
<th>Kind of Response</th>
<th>Meaning of Response</th>
<th>REP[] Specification Mapping</th>
<th>SDHC Response Register MSW</th>
<th>SDHC Response Register LSW</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1, R1b (normal response)</td>
<td>Card Status</td>
<td>REP[31:0]</td>
<td>SDHC_RESP_1</td>
<td>SDHC_RESP_0</td>
</tr>
<tr>
<td>R1b (Auto CMD12 response)</td>
<td>Card Status for Auto CMD12</td>
<td>REP[127:96]</td>
<td>SDHC_RESP_7</td>
<td>SDHC_RESP_6</td>
</tr>
<tr>
<td>R2 (CID, CSD register)</td>
<td>CID or CSD reg. incl.</td>
<td>REP [119:0]</td>
<td>SDHC_RESP_7</td>
<td>SDHC_RESP_0</td>
</tr>
<tr>
<td>R3 (OCR register)</td>
<td>OCR register for memory</td>
<td>REP [31:0]</td>
<td>SDHC_RESP_1</td>
<td>SDHC_RESP_0</td>
</tr>
<tr>
<td>R4 (OCR register)</td>
<td>OCR register for I/O, etc</td>
<td>REP [31:0]</td>
<td>SDHC_RESP_1</td>
<td>SDHC_RESP_0</td>
</tr>
<tr>
<td>R5, R5b</td>
<td>SDIO response</td>
<td>REP [31:0]</td>
<td>SDHC_RESP_1</td>
<td>SDHC_RESP_0</td>
</tr>
<tr>
<td>R6 (Published RCA response)</td>
<td>Newly published RCA[31:16], etc</td>
<td>REP [31:0]</td>
<td>SDHC_RESP_1</td>
<td>SDHC_RESP_0</td>
</tr>
</tbody>
</table>

Table 8-64: SDHC Buffer Data Port Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0 | data | R/W    | 0     | Buffer Data  
Pointer to the SDHC internal data buffer. |

Table 8-65: SDHC Present State Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:25| -                  | RO     | 0     | Reserved for Future Use  
Do not modify this field. |
| 24   | cmd_signal_level   | RO     | 0     | CMD Line Signal Level  
Indicates the CMD line level for error recovery and debugging. |
<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>23:20</td>
<td>dat_signal_level</td>
<td>RO</td>
<td>-</td>
<td>SDHC_DAT[3:0] Line Signal Level</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indicates the DAT line level for error recovery and debugging. Use to detect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the busy signal level as indicated on SDHC_DAT[0].</td>
</tr>
<tr>
<td>19</td>
<td>wp</td>
<td>RO</td>
<td>-</td>
<td>Write Protect Switch Pin Level</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The write protect switch is supported for memory and combo cards. This bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>reflects the state of the SDHC_WP pin.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Write enabled (SDHC_WP = 1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Write protected (SDHC_WP = 0)</td>
</tr>
<tr>
<td>18</td>
<td>card_detect</td>
<td>RO</td>
<td>-</td>
<td>Card Detect Pin Level</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit reflects the inverted state of the SDHC_CDN pin.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Debouncing is not performed on this bit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When Card State Stable is set to 1, this bit might be valid, but is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>not guaranteed. To use this bit, the card driver must debounce the bit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Card present (SDHC_CDN = 0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No card present (SDHC_CDN = 1)</td>
</tr>
<tr>
<td>17</td>
<td>card_state</td>
<td>RO</td>
<td>-</td>
<td>Card State Stable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Used for debugging only. If this bit reads 0, the SDHC_CDN pin level is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>not stable. If this bit reads 1, the SDHC_CDN pin level is stable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: No card or card inserted</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Reset or debouncing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><em>Note: This bit is not valid unless the SDHC_PRESENT.card_inserted bit reads 1.</em></td>
</tr>
<tr>
<td>16</td>
<td>card_inserted</td>
<td>RO</td>
<td>-</td>
<td>Card Inserted</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Indicates if a card is inserted. This signal is debounced by the SDHC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>hardware. A change in state from 0 to 1 on this bit generates an SDHC_IRQ</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>with the SDHC_INT_STAT.card_insertion flag set. Conversely, a transition of</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>this bit from a 1 to a 0 generates an SDHC_IRQ interrupt with the SDHC_INT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>STAT.card_removal field set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Card Inserted</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Reset, debouncing, or no card inserted</td>
</tr>
<tr>
<td>15:12</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>11</td>
<td>buffer_read</td>
<td>RO</td>
<td>0</td>
<td>Buffer Read Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If this bit reads 1, then data is available in the buffer for non-DMA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>transfers. This bit is cleared when all available block data is read from</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the buffer. This bit transitions from 0 to 1 when block data is ready in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the buffer resulting in a SDHC_IRQ interrupt, if enabled, with the SDHC_INT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>STAT.buffer_rd_ready flag set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Read data available</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No data to read</td>
</tr>
<tr>
<td>10</td>
<td>buffer_write</td>
<td>RO</td>
<td>0</td>
<td>Buffer Write Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If this bit reads 1, then space is available in the buffer for write data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit is cleared when no space is available in the buffer. This bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>transitions from a 0 to a 1 when top-of-block data is written to the buffer,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>resulting in a SDHC_IRQ interrupt, if enabled, with the SDHC_INT_STAT.buffer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>wr_ready flag set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Space available in the buffer for write data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No space available in the buffer for write data</td>
</tr>
</tbody>
</table>
### Present State Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>read_transfer</td>
<td>RO</td>
<td>0</td>
<td><strong>Read Transfer Active</strong>&lt;br&gt;Indicates completion of a read transfer.&lt;br&gt;This bit is set to 1 for either of the following conditions:&lt;br&gt;1) After the end bit of a Read command. 2) When a read operation is restarted by setting the <code>SDHC_BLK_GAP.cont</code> bit (Continue Request).&lt;br&gt;This bit is set to 0 for either of the following conditions:&lt;br&gt;1) The last data block as specified by the block length is transferred to the SDHC. 2) When all valid data blocks are transferred to the system, and no current block transfers are sent because the Stop At Block Gap Request register field is set to 1. A SDHC_IRQ interrupt is generated, if enabled. 1: Transferring data 0: No valid data</td>
</tr>
<tr>
<td>8</td>
<td>write_transfer</td>
<td>RO</td>
<td>0</td>
<td><strong>Write Transfer Active</strong>&lt;br&gt;This bit is set to 1 for either of the following conditions:&lt;br&gt;1) After the end bit of the Write command. 2) When a write operation is restarted by setting the <code>SDHC_BLK_GAP.cont</code> bit to 1. This bit is cleared to 0 for either of the following conditions:&lt;br&gt;1) After getting the CRC status of the last data block transfer as specified by the transfer count, single and multiple block, <code>SDHC_BLK_CNT</code> register. 2) After getting the CRC status of any block where data transmission is stopped by a Stop At Block Gap Request (<code>SDHC_BLK_GAP.stop</code>).&lt;br&gt;When <code>SDHC_BLK_GAP.stop</code> (stop at block gap request) is set, a change in <code>write_transfer</code> from 1 to 0 causes an SDHC_IRQ interrupt, if enabled, with the <code>SDHC_INT_STAT.blk_gap_event</code> flag set to 1. The <code>blk_gap_event</code> field indicates to the card driver that a non-DAT command can be issued during an active write. 1: Transferring data 0: No valid data for transfer</td>
</tr>
<tr>
<td>7:4</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>3</td>
<td>retuning</td>
<td>RO</td>
<td>0</td>
<td><strong>Re-Tuning Request</strong>&lt;br&gt;If this field reads 1, a retuning request was received from the external device. 0: Re-tuning request has not been received. 1: Re-tuning request received.</td>
</tr>
<tr>
<td>2</td>
<td>dat_line_active</td>
<td>RO</td>
<td>0</td>
<td><strong>DAT Line Active</strong>&lt;br&gt;A value of 1 indicates one or more DAT lines (SDHC_DAT[3:0]) are in use on the SD Bus. 0: No SD Bus DAT lines in use. 1: 1 or more DAT lines are in use.</td>
</tr>
</tbody>
</table>
### Present State Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dat</td>
<td>RO</td>
<td>0</td>
<td><strong>Command Inhibit (DAT)</strong>&lt;br&gt;This bit is set if DAT Line Active or the Read Transfer Active bits are set. A SDHC_IRQ interrupt is generated, if enabled, when this bit transitions from a 1 to a 0 with the SDHC_INT_STAT.trans_comp flag set. The card driver can save registers in the range of 0x000 to 0x00D for a suspend transaction after the SDHC_INT_STAT.trans_comp interrupt event.&lt;br&gt;1: Command that uses DAT line cannot be issued.&lt;br&gt;0: Command that uses DAT line can be issued.</td>
</tr>
<tr>
<td>0</td>
<td>cmd_comp</td>
<td>RO</td>
<td>0</td>
<td><strong>Command Inhibit (CMD)</strong>&lt;br&gt;If this bit reads 0, the CMD line is not in use. This bit is set to 1 by the SDHC immediately after the SDHC_CMD register is written, and the bit is cleared to 0 when the Command Response is received. Auto CMD12 and Auto CMD23 consist of two responses, and this bit is not cleared until the read/write portion of the sequence is complete.&lt;br&gt;1: Command cannot be issued.&lt;br&gt;0: Can issue command using only CMD line.</td>
</tr>
</tbody>
</table>

### SDHC_PRESENT [0x0024]

#### Table 8-66: SDHC Host Control 1 Register

<table>
<thead>
<tr>
<th>Host Control 1 Register</th>
<th>SDHC_HOST_CN_1</th>
<th>[0x0028]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>card_detect_signal</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>card_detect_test</td>
<td>R/W</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>ext_data_transfer_width</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td>4:3</td>
<td>dma_select</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>hs_en</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>data_transfer_width</td>
<td>R/W</td>
<td>0</td>
</tr>
</tbody>
</table>
### Host Control 1 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>led_cn</td>
<td>R/W</td>
<td>0</td>
<td>LED Control</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: LED on</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: LED off</td>
</tr>
</tbody>
</table>

### Table 8-67: SDHC Power Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:4</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>3:1</td>
<td>bus_volt_sel</td>
<td>R/W</td>
<td>6</td>
<td><strong>SD Bus Voltage Select</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sets the voltage level for the SD card. Validate the setting against the Capabilities Register (SDHC_CFG_0).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7: 3.3V typical</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6: 3.0V typical</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5: 1.8V typical</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4: Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3: Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2: Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Reserved for Future Use</td>
</tr>
<tr>
<td>0</td>
<td>bus_power</td>
<td>R/W</td>
<td>0</td>
<td><strong>SD Bus Power</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Before setting this bit, configure the SDHC_PWR.bus_volt_sel field. If no card is detected, then this bit is automatically set to 0 by the SDHC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Power Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Power Disabled</td>
</tr>
</tbody>
</table>

### Table 8-68: SDHC Block Gap Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:4</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>3</td>
<td>intr</td>
<td>R/W</td>
<td>0</td>
<td><strong>Interrupt at Block Gap</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting this bit to 1 enables interrupt detection at the block gap for a multiple block transfer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: This bit is only valid if SDHC_PWR.data_transfer_width=1 (4-bit mode).</td>
</tr>
</tbody>
</table>
### Block Gap Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>read_wait</td>
<td>R/W</td>
<td>0</td>
<td><strong>Read Wait Control</strong>&lt;br&gt;If the card supports read wait (optional for SDIO cards), setting this bit enables use of the read wait protocol to stop reading data using the SDHC_DAT[2] line. If the card does not support read wait, the SDHC stops the SD Clock to hold read data, preventing command generation. When a card is inserted, the card driver must set this field based on the CCCR of the SDIO card inserted.&lt;br&gt;Suspend/Resume is not supported when this bit is set to 0.&lt;br&gt;1: Enable Read Wait Control&lt;br&gt;0: Disable Read Wait Control&lt;br&gt;<em>Note: If the SDIO card does not support read wait, then you must not set this bit to 1. Setting it to 1 when read wait is not supported might cause a SDHC_DAT line conflict.</em></td>
</tr>
<tr>
<td>1</td>
<td>cont</td>
<td>R/W</td>
<td>0</td>
<td><strong>Continue Request</strong>&lt;br&gt;This bit is used to restart a transaction that was stopped using the Stop At Block Gap Request (SDHC_BLK_GAP.stop). To cancel a stop at the block gap, set SDHC_BLK_GAP.stop to 0, and set this bit, SDHC_BLK_GAP.cont, to 1 to restart the transfer.&lt;br&gt;This bit is automatically cleared by hardware for either of the following conditions:&lt;br&gt;- During a read transaction, the DAT Line Active changes from 0 to 1 as the write transaction restarts.&lt;br&gt;- During a write transaction, the Write Transfer Active changes from 0 to 1 as the write transaction restarts.&lt;br&gt;1: Restart&lt;br&gt;0: No effect</td>
</tr>
<tr>
<td>0</td>
<td>stop</td>
<td>R/W</td>
<td>0</td>
<td><strong>Stop At Block Gap Request</strong>&lt;br&gt;Setting this bit stops executing read and write transactions at the next block gap for non-DMA, SDMA, and ADMA transfers. This bit must remain set to 1 until the SDHC_INT_STAT.trans_comp bit is set to 1.&lt;br&gt;For write transfers where the card driver writes data to the Buffer Data Port Register (SDHC_BUFFER), the card driver must set this bit after all block data is written.&lt;br&gt;1: Stop&lt;br&gt;0: Transfer&lt;br&gt;This bit affects the following fields:&lt;br&gt;- Read Transfer Active, SDHC_PRESENT.read_transfer&lt;br&gt;- Write Transfer Active, SDHC_PRESENT.write_transfer&lt;br&gt;- SDHC_DAT Line Active, SDHC_PRESENT.dat_line_active&lt;br&gt;- Command Inhibit (DAT), SDHC_PRESENT.dat&lt;br&gt;<em>Note: If this bit is set to 1, the card driver must not write data to the Buffer Data Port Register (SDHC_BUFFER).</em>&lt;br&gt;<em>Note: Clearing both the SDHC_BLK_GAP.stop and SDHC_BLK_GAP.cont fields does not cause a transaction to restart.</em>&lt;br&gt;<em>Note: You can set this bit to 1 regardless of whether the card inserted supports Read Wait Control. The SDHC stops the card through Read Wait Control or by stopping the SD clock.</em></td>
</tr>
</tbody>
</table>
### Table 8-69: SDHC Wakeup Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:3</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>2</td>
<td>card_rem</td>
<td>R/W</td>
<td>0</td>
<td><strong>Wakeup Event Enable on SD Card Removal</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Enable wakeup event interrupt when the <code>SDHC_INT_STAT.card_removal</code> flag occurs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable Interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable Interrupt</td>
</tr>
<tr>
<td>1</td>
<td>card_ins</td>
<td>R/W</td>
<td>0</td>
<td><strong>Wakeup Event Enable on SD Card Insertion</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Enable wakeup event interrupt when the <code>SDHC_INT_STAT.card_inserted</code> flag occurs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable Interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable Interrupt</td>
</tr>
<tr>
<td>0</td>
<td>card_int</td>
<td>R/W</td>
<td>0</td>
<td><strong>Wakeup Event Enable On Card Interrupt</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Enable wakeup event interrupt when the <code>SDHC_INT_STAT.card_intr</code> flag occurs.</td>
</tr>
</tbody>
</table>

### Table 8-70: SDHC Clock Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:8</td>
<td>sdclk_freq_sel</td>
<td>R/W</td>
<td>0</td>
<td><strong>SDCLK Frequency Select</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Selects the SD Clock Frequency output on the SDHC_CLK pin.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The SD Clock Frequency Select is a total of 10 bits. The divisors shown below consist of the <code>upper_sdclk_freq_sel</code> bits as bits 9:8, and the <code>sdclk_freq_sel</code> bits as bits 7:0 of the divisor.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><code>upper_sdclk_freq_sel</code></th>
<th><code>sdclk_freq_sel</code></th>
<th>SDCLK Divisor (N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0b11</td>
<td>0b11111111</td>
<td>1023</td>
</tr>
<tr>
<td>0b11</td>
<td>0b00000000</td>
<td>768</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0b10</td>
<td>0b01010101</td>
<td>597</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0b00</td>
<td>0b00000001</td>
<td>2</td>
</tr>
<tr>
<td>0b00</td>
<td>0b00000000</td>
<td>1</td>
</tr>
<tr>
<td>0b00</td>
<td>0b00000000</td>
<td>0 (MAX)</td>
</tr>
</tbody>
</table>

Setting `upper_sdclk_freq_sel` and `sdclk_freq_sel` to 0 results in the maximum SDCLK frequency of $f_{SDHC_CLK\_FRQ}$. All other settings for `upper_sdclk_freq_sel` and `sdclk_freq_sel` follow the equation below:

$$SDHC\_CLK = \frac{f_{SDHC\_CLK\_FRQ}}{(2 \times N)}$$

*Note: The SD Clock Enable must be disabled (`SDHC_CLK_CN.sd_clk_en = 0`) prior to modification of this field.*
<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:6</td>
<td>upper_sdclk_freq_sel</td>
<td>R/W</td>
<td>0</td>
<td><strong>Upper Bits of SDCLK Frequency Select</strong>&lt;br&gt;Bits 9 and 8 of the 10-bit SDCLK frequency select. See the SDHC_CLK_CN.sdclk_freq_sel field for details about the clock select calculation.&lt;br&gt;Note: The SD Clock Enable must be disabled (SDHC_CLK_CN.sd_clk_en = 0) prior to modification of this field.</td>
</tr>
<tr>
<td>5</td>
<td>clk_gen_sel</td>
<td>RO</td>
<td>0</td>
<td><strong>Clock Generator Select</strong>&lt;br&gt;Reads 0 indicating Divided Clock mode only for SD Clock Frequency generation.&lt;br&gt;0: Divided clock mode</td>
</tr>
<tr>
<td>4:3</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>2</td>
<td>sd_clk_en</td>
<td>R/W</td>
<td>0</td>
<td><strong>SD Clock Enable</strong>&lt;br&gt;Enable/disable SD Clock generation.&lt;br&gt;1: Enable the SD Clock and output on the SDHC_CLK pin.&lt;br&gt;0: SD Clock is disabled.&lt;br&gt;Note: This bit is cleared by the SDHC if the card-inserted field in the Present State register is cleared.&lt;br&gt;Note: The internal_clk_en bit must be set to 1, and the internal_clk_stable bit must read 1 prior to setting this bit to 1.</td>
</tr>
<tr>
<td>1</td>
<td>internal_clk_stable</td>
<td>RO</td>
<td>0</td>
<td><strong>Internal Clock Stable</strong>&lt;br&gt;This bit is set to 1 when the internal clock is stable.&lt;br&gt;Note: The internal clock must be enabled (SDHC_CLK_CN.internal_clk_en = 1) before this field is used.</td>
</tr>
<tr>
<td>0</td>
<td>internal_clk_en</td>
<td>R/W</td>
<td>0</td>
<td><strong>Internal Clock Enable</strong>&lt;br&gt;Enable the internal clock.&lt;br&gt;Note: This bit must be set, and the internal_clk_stable bit must read 1 prior to setting the SD Clock Enable (SDHC_CLK_CN.sd_clk_en) bit.&lt;br&gt;Note: This bit is set to 0 by the SDHC if waiting for a wakeup interrupt.</td>
</tr>
</tbody>
</table>
### Table 8-71: SDHC Timeout Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:4</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
</tbody>
</table>
| 3:0    | data_count_value | R/W    | 0     | **Data Timeout Counter Value**<br>Determines the interval for DAT line timeout detection. The timeout clock frequency is generated by dividing PCLK by the value calculated using this register. See Capabilities 0 Register (SDHC_CFG_0) for the definition of TMCLK. The calculation for Data Timeout is shown in the following equation:<br>

\[
\text{Data Timeout} = \text{TMCLK} \times 2^{(13 + \text{data_count_value})}
\]

<table>
<thead>
<tr>
<th>Setting</th>
<th>Data Timeout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0b1111</td>
<td>Reserved</td>
</tr>
<tr>
<td>0b1110</td>
<td>TMCLK \times 2^{(27)}</td>
</tr>
<tr>
<td>0b1101</td>
<td>TMCLK \times 2^{(26)}</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0b0010</td>
<td>TMCLK \times 2^{(15)}</td>
</tr>
<tr>
<td>0b0001</td>
<td>TMCLK \times 2^{(14)}</td>
</tr>
<tr>
<td>0b0000</td>
<td>TMCLK \times 2^{(13)}</td>
</tr>
</tbody>
</table>

**Note:** Disable the Data Timeout Error Status Enable in the Error Interrupt Status Enable register (SDHC_ER_INT_EN.data_to).
### Table 8-72: SDHC Software Reset Register

<table>
<thead>
<tr>
<th>Software Reset Register</th>
<th>SDHC_SW_RESET</th>
<th>[0x002F]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 7:3 | - | R/W | 0 | **Reserved for Future Use**  
Do not modify this field. |
| 2 | reset_dat | RWAC | 0 | **Software Reset for DAT Line**  
1: Reset  
0: Ready  
The following registers and fields are clearedinitialized when this bit is set:  
<table>
<thead>
<tr>
<th>Register</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDHC_BUFFER</td>
<td>data</td>
</tr>
</tbody>
</table>
| SDHC_PRESENT | buffer_read  
buffer_write  
read_transfer  
write_transfer  
dat_line_active  
dat  
cmd |
| SDHC_BLK_GAP | cont  
stop |
| SDHC_INT_STAT | buff_rd_ready  
buff_wr_ready  
dma  
blk_gap_event  
trans_comp |
| Note: After setting this bit to 1, the Card Driver must poll this bit until it reads 0 to determine reset completion. |
| 1 | reset_cmd | RWAC | 0 | **Software Reset for CMD Line**  
1: Reset  
0: Ready  
The following registers and fields are cleared by setting this bit.  
<table>
<thead>
<tr>
<th>Register</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDHC_PRESENT</td>
<td>cmd</td>
</tr>
<tr>
<td>SDHC_INT_STAT</td>
<td>cmd_comp</td>
</tr>
<tr>
<td>Note: After setting this bit to 1, the card driver must poll this bit for 0 to determine when the reset is complete.</td>
<td></td>
</tr>
</tbody>
</table>
| 0 | reset_all | RWAC | 0 | **Software Reset for All**  
Reset the SDHC except for the card detection interface. All registers are reset to their Reset/POR state.  
1: Reset  
0: Ready  
*Note: After the Card Driver sets this bit to 1, the Card Driver should poll this bit until it reads 0 to determine when the SDHC completes the reset all request.* |
8.5.6.1 Normal Interrupt Status Register

The Normal Interrupt Status Enable affects reads of this register, but Normal Interrupt Signal Enable does not. An interrupt is generated when the Normal Interrupt Signal Enable is enabled, and at least one of the status bits is set to 1. Writing 1 to a bit of the RW1C attribute clears it. Writing 0 keeps the bit unchanged. Writing 1 to a bit of the ROC attribute keeps the bit unchanged. You can clear more than one status with a single register write. The Card Interrupt (SDHC_INT_STAT.card_intr) is cleared when the card stops asserting the interrupt after the Card Driver services the interrupt condition.

Table 8-73: SDHC Normal Interrupt Status Register

<table>
<thead>
<tr>
<th>Normal Interrupt Status Register</th>
<th>SDHC_INT_STAT</th>
<th>[0x0030]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 15                              | err_intr     | ROC     | 0     | Error Interrupt  
  If any of the bits in the Error Interrupt Status register are set, then this bit is set. Therefore, the Host Driver can efficiently test for an error by checking this bit first. This bit is read only.  
  1: Error  
  0: No Error |
| 14:13                           | -            | ROC     | 0     | Reserved for Future Use |
| 12                              | retuning     | ROC     | 0     | Re-Tuning Event  
  This status is set if the Re-Tuning Request bit in the Present State register changes from 0 to 1. The SDHC requests the Host Driver to perform re-tuning for the next data transfer. However, you can complete the current data transfer (not large block count) without re-tuning.  
  1: Perform re-tuning before the next data transfer  
  0: Re-tuning is not required |
| 11:9                            | -            | ROC     | 0     | Reserved for Future Use |
| 8                               | card_intr    | ROC     | 0     | Card Interrupt  
  In one-bit mode, the SDHC detects the Card interrupt without the SD Clock to support wake up. In four-bit mode, the card interrupt signal is sampled during the interrupt cycle resulting in a delay between the interrupt signal from the memory card and the interrupt signal to the host driver.  
  1: Generate Card Interrupt  
  0: No Card Interrupt  
  Note: Writing a 1 to this bit does not clear this bit. It is cleared by resetting the SDHC_INT_EN.card_int flag. |
| 7                               | card_removal | RW1C    | 0     | Card Removal  
  Set if the Card Inserted field in the Present State register (SDHC_PRESENT.card_inserted) changes from 1 to 0.  
  1: Card removed  
  0: Card state stable or hardware debouncing |
| 6                               | card_insertion | RW1C   | 0     | Card Inserted  
  Set if the Card Inserted field in the Present State register (SDHC_PRESENT.card_inserted) changes from 0 to 1.  
  1: Card inserted  
  0: Card state stable or hardware debouncing |
### Normal Interrupt Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 5    | buff_rd_ready   | RW1C   | 0     | **Buffer Read Ready**  
Set if the Buffer Read Enable field in the Present State register
(SDHCPRESENT.buff_rd_ready) changes from 0 to 1.  
1: Ready to read buffer  
0: Not ready to read buffer  
*Note: This field is set to 1 for every CMD19 execution while performing a tuning procedure (SDHC_HOST_CN_2.execute = 1).* |
| 4    | buff_wr_ready   | RW1C   | 0     | **Buffer Write Ready**  
Set if the Buffer Write Enable field in the Present State register
(SDHCPRESENT.buff_wr_ready) changes from 0 to 1.  
1: Ready to write buffer  
0: Not ready to write buffer |
| 3    | dma             | RW1C   | 0     | **DMA Interrupt**  
Set when the SDHC encounters the DMA buffer boundary set in the
SDHC_BLK_SIZE.trans field during a SDMA transfer. The Card Driver must update the
SDHC_SDMA register with the address of the next block to transfer before the SDHC
continues the transfer.  
1: SDHC DMA Interrupt is generated  
0: No SDHC DMA Interrupt |
| 2    | blk_gap_event   | RW1C   | 0     | **Block Gap Event**  
If the Stop at Block Gap Request field is set in the Block Gap Control register
(SDHCBLK_GAP.stop), this bit is set when a read or write transaction is stopped at a
block gap. If Stop at Block Gap Request is not set to 1, then this bit is not
meaningless.  
1: Transaction stopped at block gap  
0: No Block Gap Event |
| 1    | trans_comp      | RW1C   | 0     | **Transfer Complete**  
Set when a read/write transfer and a command with busy is complete. This bit has
higher priority than Data Timeout Error. If both bits are set to 1, execution of a
command is complete. See Table 8-74 for Transfer Complete and Data Timeout Error
priority and meaning.  
1: Command execution is complete  
0: Not complete  
*Note: This field is not set while performing a tuning procedure (SDHC_HOST_CN_2.execute = 1).* |
| 0    | cmd_cmp         | RW1C   | 0     | **Command Complete**  
Set when the end bit of the command response is received. Auto CMD12 and Auto
CMD23 consist of two responses. This flag is not set by the card’s response to the
CMD12 or CMD23, but by the card’s response to the read or write command you
send to complete the Auto CMD12 or Auto CMD23. See Command Inhibit (CMD) in
the Present State (SDHPRESENT.cmd_comp) register for how to control this bit.  
*Table 8-75* illustrates the relationship between Command Complete and Command
Timeout Error bits. If both bits are set, then the response was not received within 64
SD clock cycles.  
1: Command execution is complete  
0: Not complete |
Table 8-74: Transfer Complete and Data Timeout Error Priority and Status

<table>
<thead>
<tr>
<th>Transfer Complete</th>
<th>Data Timeout Error</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDHC_INT_STAT.trans_comp</td>
<td>SDHC_ER_INT_STAT.data_to</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Interrupted by another event</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Timeout occurred during transfer</td>
</tr>
<tr>
<td>1</td>
<td>N/A</td>
<td>Command execution complete</td>
</tr>
</tbody>
</table>

Table 8-75: Command Complete and Command Timeout Error Priority and Status

<table>
<thead>
<tr>
<th>Transfer Complete</th>
<th>Data Time Error</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDHC_INT_STAT.cmd_comp</td>
<td>SDHC_ER_INT_STAT.cmd_to</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Interrupted by another event.</td>
</tr>
<tr>
<td>N/A</td>
<td>1</td>
<td>Response not received within 64 SD Clock cycles.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Response received.</td>
</tr>
</tbody>
</table>

8.5.6.2 Error Interrupt Status Register

The interrupts defined in this register are enabled by the corresponding fields in the Error Interrupt Status Enable (SDHC_ER_INT_EN) register. Setting any field in the SDHC_ER_INT_SIGNAL register enables SDHC error interrupt generation using the SDHC_IRQ. The interrupt occurs when any field in the SDHC_ER_INT_STAT register is set to 1.

Table 8-76: SDHC Error Interrupt Status Register

<table>
<thead>
<tr>
<th>Error Interrupt Status Register</th>
<th>SDHC_ER_INT_STAT</th>
<th>[0x0032]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>15:13</td>
<td>-</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>dma</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:10</td>
<td>-</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>adma</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Error Interrupt Status Register

### SDHC_ER_INT_STAT

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 8    | auto_cmd_12     | R/W1C  | 0     | **Auto CMD Error**  
Auto CMD12 and Auto CMD23 use this error status. This bit is set when detecting that one of the bits D00-D04 in Auto CMD Error Status \( SDHC.AUTO.CMD.ER \) register changed from a 0 to a 1.  
\[ 1: \text{Error} \]  
\[ 0: \text{No error} \]  
**Note:** For Auto CMD12, this bit is set to 1 not only when an error occurs in Auto CMD12, but also when Auto CMD12 is not executed due to a previous command error. |
| 7    | current_limit   | R/W1C  | 0     | **Current Limit Error**  
Not supported on MAX32665—MAX32668 |
| 6    | data_end_bit    | R/W1C  | 0     | **Data End Bit Error**  
Set if a 0 is detected at the end bit position of read data that uses the DAT line or the end-bit position of the CRC status.  
\[ 1: \text{Error} \]  
\[ 0: \text{No error} \] |
| 5    | data_crc        | R/W1C  | 0     | **Data CRC Error**  
Set when a CRC error is detected when receiving read data that uses the DAT line or when detecting a Write CRC status with a value other than 010.  
\[ 1: \text{Error} \]  
\[ 0: \text{No error} \] |
| 4    | data_to         | R/W1C  | 0     | **Data Timeout Error**  
Set for any of the following timeout conditions:  
- Busy timeout for R1b and R5b response types. See Table 8-53 for more information about response types.  
- Busy timeout after Write CRC status  
- Write CRC status Timeout  
- Read Data Timeout  
\[ 1: \text{Error} \]  
\[ 0: \text{No error} \] |
| 3    | cmd_idx         | R/W1C  | 0     | **Command Index Error**  
Set if a Command Index error is detected in the Command Response.  
\[ 1: \text{Error} \]  
\[ 0: \text{No error} \] |
| 2    | cmd_end_bit     | R/W1C  | 0     | **Command End Bit Error**  
Set if the end bit of a Command Response is 0.  
\[ 1: \text{Error} \]  
\[ 0: \text{No error} \] |
Error Interrupt Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1    | cmd_crc | R/W1C  | 0     | Command CRC Error  
Set for the following cases:  
1) If a response is returned, and the Command Timeout Error is set to 0, then this error flag is set if a CRT error is detected in the Command Response.  
2) The SDHC detects a CMD-line conflict by monitoring the SDHC_CMD line when a command is issued. The SDHC sets the Command Timeout Error flag if a CMD line conflict is detected. A CMD line conflict indicates the CMD line was driven to 1, and the SDHC detected a 0 on the CMD line on the next SDCLK.  
1: Error  
0: No error |
| 0    | cmd_to | R/W1C  | 0     | Command Timeout Error  
Set if there is not response within 64 SDCLK cycles from the end bit of a command.  
1: Error  
0: No error  
Note: If both the SDHC_ER_INT_STAT.cmd_crc and SDHC_ER_INT_STAT.cmd_to flags are set, then the SDHC detected a CMD-line conflict. See SDHC_ER_INT_STAT.cmd_crc for more information about a CMD-line conflict. |

Table 8-77: SDHC Normal Interrupt Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 15:13|              | R/W    | 0     | Reserved for Future Use  
Do not modify this field. |
| 12   | retuning     | R/W    |       | Re-Tuning Event Status Enable           |
|      |              |        |       | 1: Enabled  
0: Disabled |
| 11:9 |              | R/W    | 0     | Reserved for Future Use  
Do not modify this field. |
| 8    | card_int     | R/W    | 0     | Card Interrupt Status Enable            |
|      |              |        |       | Set to enable card interrupt detection. The Card Driver should clear this bit prior to servicing a card interrupt status event and re-enable this bit after all interrupts from the card are serviced.  
1: Enabled  
0: Disabled |
| 7    | card_removal | R/W    | 0     | Card Removal Status Enable              |
|      |              |        |       | Set to enable card removal event.  
1: Enabled  
0: Disabled |
| 6    | card_insert  | R/W    | 0     | Card Insertion Status Enable            |
|      |              |        |       | Set to enable card insertion event.  
1: Enabled  
0: Disabled |
### Normal Interrupt Status Enable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>buffer_rd</td>
<td>R/W</td>
<td>0</td>
<td>Buffer Read Ready Status Enable Set to enable Buffer Read Ready status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>4</td>
<td>buffer_wr</td>
<td>R/W</td>
<td>0</td>
<td>Buffer Write Ready Status Enable Set to enable Buffer Write Ready status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>3</td>
<td>dma</td>
<td>R/W</td>
<td>0</td>
<td>DMA Interrupt Status Enable Set to enable DMA status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>2</td>
<td>blk_gap</td>
<td>R/W</td>
<td>0</td>
<td>Block Gap Event Status Enable Set to enable Block Gap status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>1</td>
<td>trans_comp</td>
<td>R/W</td>
<td>0</td>
<td>Transfer Complete Status Enable Set to enable Transfer Complete status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>0</td>
<td>cmd_comp</td>
<td>R/W</td>
<td>0</td>
<td>Command Complete Status Enable Set to enable Command Complete status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
</tbody>
</table>

### SDHC_INT_EN [0x0034]

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>buffer_rd</td>
<td>R/W</td>
<td>0</td>
<td>Buffer Read Ready Status Enable Set to enable Buffer Read Ready status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>4</td>
<td>buffer_wr</td>
<td>R/W</td>
<td>0</td>
<td>Buffer Write Ready Status Enable Set to enable Buffer Write Ready status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>3</td>
<td>dma</td>
<td>R/W</td>
<td>0</td>
<td>DMA Interrupt Status Enable Set to enable DMA status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>2</td>
<td>blk_gap</td>
<td>R/W</td>
<td>0</td>
<td>Block Gap Event Status Enable Set to enable Block Gap status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>1</td>
<td>trans_comp</td>
<td>R/W</td>
<td>0</td>
<td>Transfer Complete Status Enable Set to enable Transfer Complete status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>0</td>
<td>cmd_comp</td>
<td>R/W</td>
<td>0</td>
<td>Command Complete Status Enable Set to enable Command Complete status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
</tbody>
</table>

### SDHC_ER_INT_EN [0x0036]

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:13</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use Do not modify this field.</td>
</tr>
<tr>
<td>12</td>
<td>vendor</td>
<td>R/W</td>
<td>0</td>
<td>Target Response Error/Host Error Status Enable Set to enable Target Response/Host Error status interrupts.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>11</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use Do not modify this field.</td>
</tr>
<tr>
<td>10</td>
<td>tuning</td>
<td>R/W</td>
<td>0</td>
<td>Tuning Error Status Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>9</td>
<td>adma</td>
<td>R/W</td>
<td>0</td>
<td>ADMA Error Status Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
</tbody>
</table>

Table 8-78: SDHC Error Interrupt Status Enable Register
### Error Interrupt Status Enable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>auto_cmd_12</td>
<td>R/W</td>
<td>0</td>
<td>Auto CMD12 Error Status Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>6</td>
<td>data_end_bit</td>
<td>R/W</td>
<td>0</td>
<td>Data End Bit Error Status Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>5</td>
<td>data_crc</td>
<td>R/W</td>
<td>0</td>
<td>Data CRC Error Status Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>4</td>
<td>data_to</td>
<td>R/W</td>
<td>0</td>
<td>Data Timeout Error Status Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>3</td>
<td>cmd_idx</td>
<td>R/W</td>
<td>0</td>
<td>Command Index Error Status Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>2</td>
<td>cmd_end_bit</td>
<td>R/W</td>
<td>0</td>
<td>Command End Bit Error Status Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>1</td>
<td>cmd_crc</td>
<td>R/W</td>
<td>0</td>
<td>Command CRC Error Status Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable</td>
</tr>
<tr>
<td>0</td>
<td>cmd_to</td>
<td>R/W</td>
<td>0</td>
<td>Command Timeout Error Status Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
</tbody>
</table>

### Table 8-79: SDHC Normal Interrupt Signal Enable Register

<table>
<thead>
<tr>
<th>Normal Interrupt Signal Enable Register</th>
<th>SDHC_INT_SIGNAL</th>
<th>[0x0038]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>15:13</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>retuning</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:9</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>card_int</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>card_removal</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Normal Interrupt Signal Enable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>card_insert</td>
<td>R/W</td>
<td>0</td>
<td>Card Insertion Signal Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>5</td>
<td>buffer_rd</td>
<td>R/W</td>
<td>0</td>
<td>Buffer Read Ready Signal Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>4</td>
<td>buffer_wr</td>
<td>R/W</td>
<td>0</td>
<td>Buffer Write Ready Signal Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>3</td>
<td>dma</td>
<td>R/W</td>
<td>0</td>
<td>DMA Interrupt Signal Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>2</td>
<td>blk_gap</td>
<td>R/W</td>
<td>0</td>
<td>Block Gap Signal Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>1</td>
<td>trans_comp</td>
<td>R/W</td>
<td>0</td>
<td>Transfer Complete Signal Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>0</td>
<td>cmd_comp</td>
<td>R/W</td>
<td>0</td>
<td>Command Complete Signal Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
</tbody>
</table>

### SDHC_INT_SIGNAL

Table 8-80: SDHC Error Interrupt Signal Enable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:13</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>12</td>
<td>tar_resp</td>
<td>R/W</td>
<td>0</td>
<td>Target Response Error Signal Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>11</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>10</td>
<td>tuning</td>
<td>R/W</td>
<td>0</td>
<td>Tuning Error Signal Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>9</td>
<td>adma</td>
<td>R/W</td>
<td>0</td>
<td>ADMA Error Signal Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>8</td>
<td>auto_cmd_12</td>
<td>R/W</td>
<td>0</td>
<td>Auto CMD12 Error Signal Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>7</td>
<td>current_limit</td>
<td>R/W</td>
<td>0</td>
<td>Current Limit Error Signal Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>Error Interrupt Signal Enable Register</td>
<td>SDHC_ER_INT_SIGNAL</td>
<td>[0x003A]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-------------------</td>
<td>----------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
<td>Reset</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
<td>--------</td>
<td>-------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td>6</td>
<td>data_end_bit</td>
<td>R/W</td>
<td>0</td>
<td>Data End Bit Error Signal Enable</td>
</tr>
</tbody>
</table>
|      |                  |        |       | 1: Enabled
|      |                  |        |       | 0: Disabled                                                   |
| 5    | data_crc         | R/W    | 0     | Data CRC Error Signal Enable                                  |
|      |                  |        |       | 1: Enabled
|      |                  |        |       | 0: Disabled                                                   |
| 4    | data_to          | R/W    | 0     | Data Timeout Error Signal Enable                              |
|      |                  |        |       | 1: Enabled
|      |                  |        |       | 0: Disabled                                                   |
| 3    | cmd_idx          | R/W    | 0     | Command Index Error Signal Enable                             |
|      |                  |        |       | 1: Enabled
|      |                  |        |       | 0: Disabled                                                   |
| 2    | cmd_end_bit      | R/W    | 0     | Command End Bit Error Signal Enable                           |
|      |                  |        |       | 1: Enabled
|      |                  |        |       | 0: Disabled                                                   |
| 1    | cmd_crc          | R/W    | 0     | Command CRC Error Signal Enable                               |
|      |                  |        |       | 1: Enabled
|      |                  |        |       | 0: Disabled                                                   |
| 0    | cmd_to           | R/W    | 0     | Command Timeout Error Signal Enable                           |
|      |                  |        |       | 1: Enabled
|      |                  |        |       | 0: Disabled                                                   |

### 8.5.6.3 Auto CMD Error Status Register

This register is used to indicate response errors for Auto CMD12 and Auto CMD23. The contents of this register are only valid when the Auto CMD Error is set (SDHC_ER_INT_STAT.auto_cmd_12). For Auto CMD23 errors, the error code is stored in SDHC_AUTO_CMD_ER[4:1].

**Table 8-81: SDHC Auto CMD Error Status Register**

<table>
<thead>
<tr>
<th>Auto CMD Error Status Register</th>
<th>SDHC_AUTO_CMD_ER</th>
<th>[0x003C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>------</td>
<td>------------------</td>
<td>-----------</td>
</tr>
</tbody>
</table>
| 15:8 | -                 | R/W       | 0     | Reserved for Future Use
|      |                  |           |       | Do not modify this field.                                                   |
| 7    | not_issued       | ROC       | 0     | Command Not Issued by Auto CMD12 Error                                     |
|      |                  |           |       | 1: Command not issued due to Auto CMD12 error as indicated in bits 4:1 of  |
|      |                  |           |       | this register.                                                             |
|      |                  |           |       | 0: Auto CMD Error issued by Auto CMD23                                     |
| 6:5  | -                 | ROC       | 0     | Reserved for Future Use
|      |                  |           |       | Do not modify this field.                                                   |
| 4    | index             | ROC       | 0     | Auto CMD Index Error                                                        |
|      |                  |           |       | Command Index error occurred in response to a command.                      |
|      |                  |           |       | 1: Command Index Error                                                      |
|      |                  |           |       | 0: No Error                                                                 |
**Auto CMD Error Status Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>end_bit</td>
<td>ROC</td>
<td>0</td>
<td><strong>Auto CMD End Bit Error</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set if the end bit of the Command Response is 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 1: End Bit Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 0: No Error</td>
</tr>
<tr>
<td>2</td>
<td>crc</td>
<td>ROC</td>
<td>0</td>
<td><strong>Auto CMD CRC Error</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set if CRC error in command response.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 1: CRC Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 0: No Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> If both SDHC_AUTO_CMD_ER.crc and SDHC_AUTO_CMD_ER.to are set, then a CMD-line conflict occurred.</td>
</tr>
<tr>
<td>1</td>
<td>to</td>
<td>ROC</td>
<td>0</td>
<td><strong>Auto CMD Timeout Error</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set if no response is returned within 64 SDCLK cycles from the end bit of the command. If set, then ignore bits 4:2 of this register.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 1: Timeout Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 0: No Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> If both SDHC_AUTO_CMD_ER.crc and SDHC_AUTO_CMD_ER.to are set, then a CMD-line conflict occurred.</td>
</tr>
<tr>
<td>0</td>
<td>not_execute</td>
<td>ROC</td>
<td>0</td>
<td><strong>Auto CMD12 Not Executed Error</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Auto CMD12 was not issued to stop a multi-block memory transfer due to an error with a prior command.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 1: Not Executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 0: No Error or error generated by Auto CMD23</td>
</tr>
</tbody>
</table>

**Table 8-82: SDHC Host Control 2 Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>preset_val_en</td>
<td>R/W</td>
<td>0</td>
<td><strong>Preset Value Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When set to 0, the following fields must be set by the Card Driver:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• SDCLK Frequency Select (SDHC_CLK_CN.sclk_freq_sel)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Clock Generator Select (SDHC_CLK_CN.clk_gen_sel)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Driver Strength Select (SDHC_HOST_CN_2.driver_strength)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If set to 1, the Host Controller hardware sets the above fields based on the values in the Preset Value registers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Card Driver must set the SDCLK Frequency Select, Clock Generator Select and Driver Strength Select fields.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: The Host Controller hardware sets the above fields using the Preset Value register settings.</td>
</tr>
<tr>
<td>14</td>
<td>asynch_int</td>
<td>R/W</td>
<td>0</td>
<td><strong>Asynchronous Interrupt Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Always reads 0. Asynchronous Interrupt Enable is not supported by the MAX32665—MAX32668. Writes to this field have no effect.</td>
</tr>
<tr>
<td>13:8</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
</tbody>
</table>
### Host Control 2 Register (SDHC_HOST_CN_2)

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 7    | sampling_clk    | R/W    | 0     | **Sampling Clock Select**  
This field is automatically set by hardware when Execute Tuning (SDHC_HOST_CN_2.execute) is cleared.  
0: The fixed clock is used to sample data  
1: The tuned clock is used to sample data  
**Note:** The Card Driver cannot write 1 to this bit. Writing this bit to 0 can only be done if the Host Controller is not receiving a response or reading a data block. |
| 6    | execute         | R/WAC  | 0     | **Execute Tuning**  
Setting this bit to 1 starts the tuning procedure and the bit is automatically cleared when the Host Controller completes the tuning procedure. Writing a 0 to this bit when it is set to 1 aborts the tuning procedure.  
1: Execute tuning  
0: Tuning complete or not tuned |
| 5:4  | driver_strength | R/W    | 0     | **Driver Strength Select**  
If using 3.3V signaling, this field is ignored. For 1.8V signaling, the output driver strength is set using this field.  
If SDHC_HOST_CN_2.preset_val_en = 0, this field is controlled by the Host Driver. If SDHC_HOST_CN_2.preset_val_en = 1, this field is automatically set by the hardware using the Preset Value registers.  
00b: Driver Type B is selected  
01b: Driver Type A is selected  
10b: Driver Type C is selected  
11b: Driver Type D is selected |
| 3    | 1_8v_signal     | R/W    | 0     | **1.8V Signaling Enable**  
If the card inserted supports UHS-I, this bit can be set to 1. No matter the value set, 3.3V is used for the card’s supply.  
1: 1.8V signaling  
0: 3.3V signaling |
| 2:0  | uhs             | R/W    | 0     | **UHS Mode Select**  
Used to select the UHS-I mode. This field is only used if 1.8V signaling is set to 1 (SDHC_HOST_CN_2.1_8v_signal = 1).  
000b: SDR12  
001b: SDR25  
010b: SDR50  
011b: SDR104 (Not supported)  
100b: DDR50  
101b – 111b: Reserved for Future Use |

### SDHC Capabilities Register 0 (SDHC_CFG_0)

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:30| slot_type | RO     | 0b00  | **Slot Type**  
0b00: Support for a single slot with support for a removable card |
| 29   | async_int | RO     | 1     | **Asynchronous Interrupt Support**  
1: Asynchronous Interrupt Supported |
| 28   | 64_bit_sys_bus | RO | 0     | **64-bit System Bus Support**  
0: 64-bit system bus not supported |
<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td>26</td>
<td>1_8v</td>
<td>RO</td>
<td>1</td>
<td>Voltage Support 1.8V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: 1.8V supported</td>
</tr>
<tr>
<td>25</td>
<td>3_0v</td>
<td>RP</td>
<td>1</td>
<td>Voltage Support 3.0V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: 3.0V supported</td>
</tr>
<tr>
<td>24</td>
<td>3_3v</td>
<td>RO</td>
<td>1</td>
<td>Voltage Support 3.3V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: 3.3V supported</td>
</tr>
<tr>
<td>23</td>
<td>suspend</td>
<td>RO</td>
<td>1</td>
<td>Suspend/Resume Support</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Suspend / Resume functionality is supported</td>
</tr>
<tr>
<td>22</td>
<td>sdma</td>
<td>RO</td>
<td>1</td>
<td>SDMA Support</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SDMA is supported and can transfer data between system memory and the SDHC directly.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: SDMA supported</td>
</tr>
<tr>
<td>21</td>
<td>hs</td>
<td>RO</td>
<td>1</td>
<td>High Speed Support</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The SDHC supports High speed mode with f_{CLK}=96MHz/2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: High speed mode supported</td>
</tr>
<tr>
<td>20</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>19</td>
<td>adma2</td>
<td>RO</td>
<td>1</td>
<td>ADMA2 Support</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The SDHC supports ADMA2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: ADMA2 supported</td>
</tr>
<tr>
<td>18</td>
<td>8_bit</td>
<td>RO</td>
<td>0</td>
<td>8-bit Support for Embedded Device</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The SDHC supports 8-bit bus width mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: 8-bit Bus width not supported</td>
</tr>
<tr>
<td>17:16</td>
<td>max_blk_len</td>
<td>RO</td>
<td>0b10</td>
<td>Max Block Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This value indicates the maximum block size that the Host Driver can read and write to the buffer in the SDHC without wait cycles. The transfer block length is always 512 bytes for SD memory cards regardless of this field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b10: 2048 bytes</td>
</tr>
<tr>
<td>15:8</td>
<td>clk_freq</td>
<td>RO</td>
<td>0x00</td>
<td>Base Clock Frequency for SD Clock</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x00: Get information using another method</td>
</tr>
<tr>
<td>7</td>
<td>clk_unit</td>
<td>RO</td>
<td>1</td>
<td>Timeout Clock Unit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: MHz base clock unit</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>5:0</td>
<td>clk_freq</td>
<td>RO</td>
<td>0x01</td>
<td>Timeout Clock Frequency</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The base clock frequency used to detect Data Timeout errors. The Timeout Clock Unit defines the units of this field's value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: 1 [MHz]</td>
</tr>
</tbody>
</table>
### Table 8-84: SDHC Capabilities Register 1

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:24| -               | RO     | 1     | Reserved for Future Use  
Do not modify this field.                                                     |
| 23:16| clk_multi       | RO     | 0     | Clock Multiplier  
Always reads 0x00.  
0: Programmable clock generation is not supported.                          |
| 15:14| retuning        | RO     | 0     | Re-Tuning Modes  
Always reads 0b00. The SDHC supports Mode 1 Re-Tuning only with timer  
controlled by the host driver and a maximum of 4MB data length.             |
| 13   | tuning_sdr50    | RO     | 0     | Use Tuning for SDR50  
1: Tuning required for SDR50  
0: SDR50 does not require tuning                                             |
| 12   | -               | RO     | 0     | Reserved for Future Use  
Do not modify this field.                                                    |
| 11:8 | timer_cnt_tuning| RO     | 0     | Timer Count for Re-Tuning  
0x0: Re-Tuning Timer disabled  
0x1: 1 second  
0x2: 2 seconds  
0x3: 4 seconds  
0x4: 8 seconds  
...: 2(n-1) seconds  
...: n seconds  
0xB: 1024 seconds  
0xC: Reserved  
0xD: Reserved  
0xE: Reserved  
0xF: Get information from another source                                    |
| 7    | -               | RO     | 0     | Reserved for Future Use  
Do not modify this field.                                                    |
| 6    | driver_d        | RO     | 1     | Driver Type D Support  
1: Driver Type D is supported                                                |
| 5    | driver_c        | RO     | 1     | Driver Type C Support  
1: Driver Type C is supported                                                |
| 4    | driver_a        | RO     | 1     | Driver Type A Support  
1: Driver Type A is supported                                                |
| 3    | -               | RO     | 0     | Reserved for Future Use  
Do not modify this field.                                                    |
| 2    | ddr50           | RO     | 1     | DDR50 Support  
1: DDR50 is support                                                         |
| 1    | sdr104          | RO     | 1     | SDR104  
1: SDR104 is supported                                                      |
| 0    | sdr50           | RO     | 1     | SDR50  
1: SDR50 is supported                                                       |
### Table 8-85: SDHC Maximum Current Capabilities Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>23:16</td>
<td>1_8v</td>
<td>RO</td>
<td>0</td>
<td>Maximum Current for 1.8V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x00: System dependent</td>
</tr>
<tr>
<td>15:8</td>
<td>3_0v</td>
<td>RO</td>
<td>0</td>
<td>Maximum Current for 3.0V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x00: System dependent</td>
</tr>
<tr>
<td>7:0</td>
<td>3_3v</td>
<td>RO</td>
<td>0</td>
<td>Maximum Current for 3.3V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x00: System dependent</td>
</tr>
</tbody>
</table>

### Table 8-86: SDHC Force Event Register for Auto CMD Error Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:8</td>
<td>-</td>
<td>WO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>7</td>
<td>not_issued</td>
<td>WO</td>
<td>0</td>
<td>Force Event for Command Not Issued By Auto CMD12 Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is generated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No interrupt generated</td>
</tr>
<tr>
<td>6:5</td>
<td></td>
<td>WO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify field.</td>
</tr>
<tr>
<td>4</td>
<td>index</td>
<td>WO</td>
<td>0</td>
<td>Force Event for Auto CMD Index Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is generated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No interrupt generated</td>
</tr>
<tr>
<td>3</td>
<td>end_bit</td>
<td>WO</td>
<td>0</td>
<td>Force Event for Auto CMD End Bit Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is generated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No interrupt generated</td>
</tr>
<tr>
<td>2</td>
<td>crc</td>
<td>WO</td>
<td>0</td>
<td>Force Event for Auto CMD CRC Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is generated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No interrupt generated</td>
</tr>
<tr>
<td>1</td>
<td>to</td>
<td>WO</td>
<td>0</td>
<td>Force Event for Auto CMD Timeout Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is generated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No interrupt generated</td>
</tr>
<tr>
<td>0</td>
<td>not_excu</td>
<td>WO</td>
<td>0</td>
<td>Force Event for Auto CMD12 Not Executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is generated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No interrupt generated</td>
</tr>
</tbody>
</table>

### Table 8-87: SDHC Force Event Register for Error Interrupt Status

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:12</td>
<td>stat_vendor</td>
<td>R/W</td>
<td>0</td>
<td>Force Event for Vendor Specific Error Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is generated</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No interrupt generated</td>
</tr>
</tbody>
</table>
### Force Event Register for Error Interrupt Status

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 11:10 | -                   | R/W    | 0     | Reserved for Future Use
Do not modify this field.                                                        |
| 9     | adma                | R/W    | 0     | Force Event for ADMA Error
1: Interrupt is generated
0: No interrupt generated                                                        |
| 8     | auto_cmd            | R/W    | 0     | Force Event for Auto CMD Error
1: Interrupt is generated
0: No interrupt generated                                                        |
| 7     | curr_limit          | R/W    | 0     | Force Event for Current Limit Error
1: Interrupt is generated
0: No interrupt generated                                                        |
| 6     | data_end_bit        | R/W    | 0     | Force Event for Data End Bit Error
1: Interrupt is generated
0: No interrupt generated                                                        |
| 5     | data_crc            | R/W    | 0     | Force Event for Data CRC Error
1: Interrupt is generated
0: No interrupt generated                                                        |
| 4     | data_to             | R/W    | 0     | Force Event for Data Timeout Error
1: Interrupt is generated
0: No interrupt generated                                                        |
| 3     | cmd_index           | R/W    | 0     | Force Event for Command Index Error
1: Interrupt is generated
0: No interrupt generated                                                        |
| 2     | cmd_end_bit         | R/W    | 0     | Force Event for Command End Bit Error
1: Interrupt is generated
0: No interrupt generated                                                        |
| 1     | cmd_crc             | R/W    | 0     | Force Event for Command CRC Error
1: Interrupt is generated
0: No interrupt generated                                                        |
| 0     | cmd_to              | R/W    | 0     | Force Event for Command Timeout Error
1: Interrupt is generated
0: No interrupt generated                                                        |

Table 8-88: SDHC ADMA Error Status Register

<table>
<thead>
<tr>
<th>ADMA Error Status Register</th>
<th>SDHC_ADMA_ER</th>
<th>[0x0054]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td>--------</td>
</tr>
</tbody>
</table>
| 7:3   | -    | RO     | 0     | Reserved for Future Use
Do not modify this field.                                                        |
### ADMA Error Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>len_mismatch</td>
<td>ROC</td>
<td>0</td>
<td>ADMA Length Mismatch Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This error occurs in the following two cases:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1) When setting Block Count Enable, the total data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>length specified by the Descriptor Table is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>different from that specified by the Block</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Count and Block Length fields.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2) Total data length is not divisible by the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Block Length field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No Error</td>
</tr>
<tr>
<td>1:0</td>
<td>state</td>
<td>ROC</td>
<td>0b00</td>
<td>ADMA Error State</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The state of the ADMA when the error condition</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>occurred. Only valid during data transfer for</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ADMA.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The following table shows the possible state</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>values, the associated ADMA Error State, and the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>contents of the SDHC_SDMA register.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>state</th>
<th>ADMA Error State when the error occurred</th>
<th>SYS_SDR register contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0b00</td>
<td>ST_STOP (Stop DMA)</td>
<td>Points next to the error descriptor</td>
</tr>
<tr>
<td>0b01</td>
<td>ST_FDS (Fetch Descriptor)</td>
<td>Points to the error descriptor</td>
</tr>
<tr>
<td>0b10</td>
<td>N.A.</td>
<td>N.A.</td>
</tr>
<tr>
<td>0b11</td>
<td>ST_TFR (Transfer Data)</td>
<td>Points next to the error descriptor</td>
</tr>
</tbody>
</table>

Note: 0b10 is not a valid error state and is never set.

### SDHC_ADMA_ADDR_0

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>addr</td>
<td>R/W</td>
<td>0</td>
<td>ADMA System Address 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Holds the byte address of the executing command</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>for the Descriptor Table. The Host Driver must</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>set this address, made up of</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&lt;SDHC_ADMA_ADDR_1&gt;&lt;SDHC_ADMA_ADDR_0&gt;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>, to the start address of the Descriptor Table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The ADMA increments this register address when</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>fetching a descriptor line to point to the next</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>address. When an ADMA Error Interrupt occurs,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>this register holds a valid descriptor address</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>depending on the ADMA state. The following table</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>shows the 64-bit System Address for ADMA using</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>&lt;SDHC_ADMA_ADDR_1&gt;&lt;SDHC_ADMA_ADDR_0&gt;.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SDHC_ADMA_ADDR_1</th>
<th>SDHC_ADMA_ADDR_0</th>
<th>64-bit System Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000 0000</td>
<td>0x0000 0000</td>
<td>0x00000000_00000000</td>
</tr>
<tr>
<td>0x0000 0000</td>
<td>0x0000 0004</td>
<td>0x00000000_00000004</td>
</tr>
<tr>
<td>0x0000 0000</td>
<td>0x0000 0008</td>
<td>0x00000000_00000008</td>
</tr>
<tr>
<td>0x0000 0000</td>
<td>0x0000 000C</td>
<td>0x00000000_0000000C</td>
</tr>
<tr>
<td>........</td>
<td>........</td>
<td>........</td>
</tr>
<tr>
<td>0xFFFF FFFF</td>
<td>0xFFFF FFFC</td>
<td>0xFFFFFFFF_FFFFFFFC</td>
</tr>
</tbody>
</table>

Note: The Host Driver must program the Descriptor Table on 32-bit boundaries and set the 32-bit boundary address to this register. ADMA2 ignores the lower two bits of this register, assuming it to be 00b.
8.5.6.4 Preset Value Registers

All Preset Value registers (SDHC_PRESET_0 to SDHC_PRESET_7) contain the same fields as described in the SDHC_PRESET_0 register. One of the Preset Value registers is automatically selected by the SDHC based on the selected bus-speed mode.

Table 8-91 shows a group of preset values per card or device. One of the Preset Value registers (SDHC_PRESET_1 – SDHC_PRESET_7) is selected by the SDHC hardware based on the Selected Bus Speed mode. Table 8-92 defines the conditions to select one of the Preset Value registers.

Table 8-91: Preset Value Register Example

<table>
<thead>
<tr>
<th>Offset</th>
<th>Preset Value Registers</th>
<th>Signal Voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0060]</td>
<td>Preset Value for Initialization</td>
<td>3.3V or 1.8V</td>
</tr>
<tr>
<td>[0x0062]</td>
<td>Preset Value for Default Speed</td>
<td>3.3V</td>
</tr>
<tr>
<td>[0x0064]</td>
<td>Preset Value for High Speed</td>
<td>3.3V</td>
</tr>
<tr>
<td>[0x0066]</td>
<td>Preset Value for SDR12</td>
<td>1.8V</td>
</tr>
<tr>
<td>[0x0068]</td>
<td>Preset Value for SDR25</td>
<td>1.8V</td>
</tr>
<tr>
<td>[0x006A]</td>
<td>Preset Value for SDR50</td>
<td>1.8V</td>
</tr>
<tr>
<td>[0x006C]</td>
<td>Preset Value for SDR104</td>
<td>1.8V</td>
</tr>
<tr>
<td>[0x006E]</td>
<td>Preset Value for DDR50</td>
<td>1.8V</td>
</tr>
</tbody>
</table>

Table 8-92: Preset Value Register Selection Conditions

<table>
<thead>
<tr>
<th>Selected Bus Speed Mode</th>
<th>1.8V Signaling Enable</th>
<th>High Speed Enable</th>
<th>UHS-I Mode Selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Speed</td>
<td>0</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>High Speed</td>
<td>0</td>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td>SDR12</td>
<td>1</td>
<td>N/A</td>
<td>0b000</td>
</tr>
<tr>
<td>SDR25</td>
<td>1</td>
<td>N/A</td>
<td>0b001</td>
</tr>
<tr>
<td>SDR50</td>
<td>1</td>
<td>N/A</td>
<td>0b010</td>
</tr>
<tr>
<td>SDR104</td>
<td>1</td>
<td>N/A</td>
<td>0b011</td>
</tr>
<tr>
<td>DDR50</td>
<td>1</td>
<td>N/A</td>
<td>0b100</td>
</tr>
<tr>
<td>Reserved</td>
<td>1</td>
<td>N/A</td>
<td>0b101 to 0b111</td>
</tr>
</tbody>
</table>
### Table 8-93: SDHC Preset Value 0 to Preset Value 7 Registers

<table>
<thead>
<tr>
<th>Preset Value</th>
<th>SDHC_PRESET</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 for Initialization</td>
<td>SDHC_PRESET_0</td>
<td>0x0060</td>
</tr>
<tr>
<td>1 for Initialization</td>
<td>SDHC_PRESET_1</td>
<td>0x0062</td>
</tr>
<tr>
<td>2 for Initialization</td>
<td>SDHC_PRESET_2</td>
<td>0x0064</td>
</tr>
<tr>
<td>3 for Initialization</td>
<td>SDHC_PRESET_3</td>
<td>0x0066</td>
</tr>
<tr>
<td>4 for Initialization</td>
<td>SDHC_PRESET_4</td>
<td>0x0068</td>
</tr>
<tr>
<td>5 for Initialization</td>
<td>SDHC_PRESET_5</td>
<td>0x006A</td>
</tr>
<tr>
<td>6 for Initialization</td>
<td>SDHC_PRESET_6</td>
<td>0x006C</td>
</tr>
<tr>
<td>7 for Initialization</td>
<td>SDHC_PRESET_7</td>
<td>0x006E</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:14</td>
<td>driver_strength</td>
<td>RO</td>
<td>1</td>
<td>Driver Strength Select Value &lt;br&gt;Driver strength is supported by 1.8V signaling bus speed modes. This field is not used for 3.3V signaling. &lt;br&gt;0b00: Driver Type B is selected &lt;br&gt;0b01: Driver Type A is selected &lt;br&gt;0b10: Driver Type C is selected &lt;br&gt;0b11: Driver Type D is selected</td>
</tr>
<tr>
<td>13:11</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use &lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>10:9</td>
<td>clk_gen</td>
<td>RO</td>
<td>0</td>
<td>Clock Generator Select Value &lt;br&gt;0: Programmable clock generator is not supported</td>
</tr>
<tr>
<td>9:0</td>
<td>sdclk_freq</td>
<td>RO</td>
<td>-</td>
<td>SDCLK Frequency Select Value &lt;br&gt;10-bit preset value to set the SDCLK Frequency Select field in the Clock Control register (SDHC_CLK_CN.upper_sdclk_freq_sel and SDHC_CLK_CN.sdclk_freq_sel)</td>
</tr>
</tbody>
</table>

### Table 8-94: SDHC Slot Interrupt Status Register

<table>
<thead>
<tr>
<th>Slot Interrupt Status Register</th>
<th>SDHC_SLOT_INT</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SDHC_SLOT_INT</td>
<td>0x00FC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:8</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use &lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>7:1</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use &lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>0</td>
<td>int_signals</td>
<td>RO</td>
<td>0</td>
<td>Interrupt Signals &lt;br&gt;Indicates the logical OR of Interrupt Signal and Wakeup Signal for the single slot. Only one slot is defined for the MAX32665—MAX32668, slot 0. Reset by POR and by software reset for all (SDHC_SW_RESET.reset_all).</td>
</tr>
</tbody>
</table>
### Table 8-95: SDHC Host Controller Version Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:8</td>
<td>vend_ver</td>
<td>RO</td>
<td>-</td>
<td><strong>Vendor Version</strong>&lt;br&gt;This status is reserved for the vendor version number. The Host Driver should not use this status.</td>
</tr>
<tr>
<td>7:0</td>
<td>spec_ver</td>
<td>RO</td>
<td>0x02</td>
<td><strong>Specification Version Number</strong>&lt;br&gt;This status indicates the Host Controller Specification Version.&lt;br&gt;0x02: SD Host Specification Version 3.00</td>
</tr>
</tbody>
</table>
9. **Standard DMA (DMAC)**

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

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

9.1 **Instances**

There are two instances of the DMAC, generically referred to as DMACm. Each instance provides 8 channels, generically referred to as DMACHn. Each instance of the DMAC has a set of interrupt registers common to all its channels, and a set of registers unique to each channel instance.
Table 9-1: MAX32665—MAX32668 DMAC and Channel Instances

<table>
<thead>
<tr>
<th>DMACm Instance</th>
<th>DMACHn Channel Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMAC0</td>
<td>DMACH0</td>
</tr>
<tr>
<td></td>
<td>DMACH1</td>
</tr>
<tr>
<td></td>
<td>DMACH2</td>
</tr>
<tr>
<td></td>
<td>DMACH3</td>
</tr>
<tr>
<td></td>
<td>DMACH4</td>
</tr>
<tr>
<td></td>
<td>DMACH5</td>
</tr>
<tr>
<td></td>
<td>DMACH6</td>
</tr>
<tr>
<td></td>
<td>DMACH7</td>
</tr>
<tr>
<td>DMAC1</td>
<td>DMACH0</td>
</tr>
<tr>
<td></td>
<td>DMACH1</td>
</tr>
<tr>
<td></td>
<td>DMACH2</td>
</tr>
<tr>
<td></td>
<td>DMACH3</td>
</tr>
<tr>
<td></td>
<td>DMACH4</td>
</tr>
<tr>
<td></td>
<td>DMACH5</td>
</tr>
<tr>
<td></td>
<td>DMACH6</td>
</tr>
<tr>
<td></td>
<td>DMACH7</td>
</tr>
</tbody>
</table>

9.2 DMA Channel Operation (DMACH)

9.2.1 DMA Channel Arbitration and DMA Bursts

DMAC 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 `DMACHn_CFG.priority` field determines the DMA channel priority.

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

A DMA channel is enabled using the `DMACHn_CFG.chen` bit.

When disabling a channel, poll the `DMACHn_ST.ch_st` bit to determine if the channel is truly disabled. In general, `DMACHn_ST.ch_st` follows the setting of the `DMACHn_CFG.chen` bit. However, the `DMACHn_ST.ch_st` bit is automatically cleared under the following conditions:

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

Whenever `DMACHn_ST.ch_st` transitions from 1 to 0, the corresponding `DMACHn_CFG.chen` bit is also cleared. If an active channel is disabled during an AHB read/write burst, the current burst will continuial until completed.

Only an error condition can interrupt an ongoing data transfer.
9.2.2 DMA Source and Destination Addressing

The source and destination for DMA transfers are dictated by the request select dedicated to the peripheral instance. The `DMACHn_CFG.reqsel` field dictates the source and destination for a channel’s DMA transfer as shown in Table 9-2. The `DMACHn_SRC` and `DMACHn_DST` registers hold the source and/or destination memory addresses, depending on the specific operation.

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

Table 9-2: MAX32665—MAX32668 DMAC Source and Destination by Peripheral

<table>
<thead>
<tr>
<th>DMACHn_CFG.reqs</th>
<th>Peripheral</th>
<th>DMA Source</th>
<th>DMA Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Memory-to-Memory</td>
<td><code>DMACHn_SRC</code></td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x01</td>
<td>SPI0</td>
<td>SPI0 Receive FIFO</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x02</td>
<td>SPI1</td>
<td>SPI1 Receive FIFO</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x03</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>UART0</td>
<td>UART0 Receive FIFO</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x05</td>
<td>UART1</td>
<td>UART1 Receive FIFO</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x06</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x07</td>
<td>I2C0</td>
<td>I2C0 Receive FIFO</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x08</td>
<td>I2C1</td>
<td>I2C1 Receive FIFO</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x09</td>
<td>ADC</td>
<td>ADC FIFO</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x0A</td>
<td>I2C2</td>
<td>I2C2 Receive FIFO</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x0B</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0D</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0E</td>
<td>UART2</td>
<td>UART2 Receive FIFO</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x0F</td>
<td>SPI2</td>
<td>SPI2 Receive FIFO</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x10</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x11</td>
<td>USB1</td>
<td>USB OUT Endpoint 1</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x12</td>
<td>USB2</td>
<td>USB OUT Endpoint 2</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x13</td>
<td>USB3</td>
<td>USB OUT Endpoint 3</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x14</td>
<td>USB4</td>
<td>USB OUT Endpoint 4</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x15</td>
<td>USB5</td>
<td>USB OUT Endpoint 5</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x16</td>
<td>USB6</td>
<td>USB OUT Endpoint 6</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x17</td>
<td>USB7</td>
<td>USB OUT Endpoint 7</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x18</td>
<td>USB8</td>
<td>USB OUT Endpoint 8</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x19</td>
<td>USB9</td>
<td>USB OUT Endpoint 9</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x1A</td>
<td>USB10</td>
<td>USB OUT Endpoint 10</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>0x1B</td>
<td>USB11</td>
<td>USB OUT Endpoint 11</td>
<td><code>DMACHn_DST</code></td>
</tr>
<tr>
<td>DMAChn_CFG.reqs</td>
<td>Peripheral</td>
<td>DMA Source</td>
<td>DMA Destination</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------</td>
<td>------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>0x1C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1D</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1E</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1F</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x20</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x21</td>
<td>SPI0</td>
<td>DMAChn_SRC</td>
<td>SPI0 Transmit FIFO</td>
</tr>
<tr>
<td>0x22</td>
<td>SPI1</td>
<td>DMAChn_SRC</td>
<td>SPI1 Transmit FIFO</td>
</tr>
<tr>
<td>0x23</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x24</td>
<td>UART0</td>
<td>DMAChn_SRC</td>
<td>UART0 Transmit FIFO</td>
</tr>
<tr>
<td>0x25</td>
<td>UART1</td>
<td>DMAChn_SRC</td>
<td>UART1 Transmit FIFO</td>
</tr>
<tr>
<td>0x26</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x27</td>
<td>I2C0</td>
<td>DMAChn_SRC</td>
<td>I2C0 Transmit FIFO</td>
</tr>
<tr>
<td>0x28</td>
<td>I2C1</td>
<td>DMAChn_SRC</td>
<td>I2C1 Transmit FIFO</td>
</tr>
<tr>
<td>0x29</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2A</td>
<td>I2C2</td>
<td>DMAChn_SRC</td>
<td>I2C2 Transmit FIFO</td>
</tr>
<tr>
<td>0x2B</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2D</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2E</td>
<td>UART2</td>
<td>DMAChn_SRC</td>
<td>UART2 Transmit FIFO</td>
</tr>
<tr>
<td>0x2F</td>
<td>SPI2</td>
<td>DMAChn_SRC</td>
<td>SPI2 Transmit FIFO</td>
</tr>
<tr>
<td>0x30</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x31</td>
<td>USB1</td>
<td>DMAChn_SRC</td>
<td>USB IN Endpoint 1</td>
</tr>
<tr>
<td>0x32</td>
<td>USB2</td>
<td>DMAChn_SRC</td>
<td>USB IN Endpoint 2</td>
</tr>
<tr>
<td>0x33</td>
<td>USB3</td>
<td>DMAChn_SRC</td>
<td>USB IN Endpoint 3</td>
</tr>
<tr>
<td>0x34</td>
<td>USB4</td>
<td>DMAChn_SRC</td>
<td>USB IN Endpoint 4</td>
</tr>
<tr>
<td>0x35</td>
<td>USB5</td>
<td>DMAChn_SRC</td>
<td>USB IN Endpoint 5</td>
</tr>
<tr>
<td>0x36</td>
<td>USB6</td>
<td>DMAChn_SRC</td>
<td>USB IN Endpoint 6</td>
</tr>
<tr>
<td>0x37</td>
<td>USB7</td>
<td>DMAChn_SRC</td>
<td>USB IN Endpoint 7</td>
</tr>
<tr>
<td>0x38</td>
<td>USB8</td>
<td>DMAChn_SRC</td>
<td>USB IN Endpoint 8</td>
</tr>
<tr>
<td>0x39</td>
<td>USB9</td>
<td>DMAChn_SRC</td>
<td>USB IN Endpoint 9</td>
</tr>
<tr>
<td>0x3A</td>
<td>USB10</td>
<td>DMAChn_SRC</td>
<td>USB IN Endpoint 10</td>
</tr>
<tr>
<td>0x3B</td>
<td>USB11</td>
<td>DMAChn_SRC</td>
<td>USB IN Endpoint 11</td>
</tr>
<tr>
<td>0x3C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3D</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3E</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**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>
<thead>
<tr>
<th>Register/Field</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMAChn_SRC</td>
<td>Source address</td>
<td>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.</td>
</tr>
<tr>
<td>DMAChn_CNT</td>
<td>Number of bytes to transfer before a CTZ condition occurs</td>
<td>This register is decremented on each read of the burst.</td>
</tr>
<tr>
<td>DMAChn_CFG.brst</td>
<td>Burst size (1-32)</td>
<td>This maximum number of bytes moved during the burst read.</td>
</tr>
<tr>
<td>DMAChn_CFG.srcw</td>
<td>Source width</td>
<td>This determines the maximum data width used during each read of the AHB burst (byte, two bytes, or four bytes). The actual AHB width might be less if DMAChn_CNT is not great enough to supply all the needed bytes.</td>
</tr>
<tr>
<td>DMAChn_CFG.srcinc</td>
<td>Source increment enable</td>
<td>Increments DMAChn_SRC. This field is ignored when the DMA source is a peripheral.</td>
</tr>
</tbody>
</table>

**9.2.3 Data Movement From the 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>
<thead>
<tr>
<th>Register/Field</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMAChn_DST</td>
<td>Destination address</td>
<td>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.</td>
</tr>
<tr>
<td>DMAChn_CFG.brst</td>
<td>Burst size (1-32)</td>
<td>The maximum number of bytes moved during a single AHB read/write burst.</td>
</tr>
<tr>
<td>DMAChn_CFG.dstwd</td>
<td>Destination width</td>
<td>This determines the maximum data width used during each write of the AHB burst (one byte, two bytes, or four bytes).</td>
</tr>
<tr>
<td>DMAChn_CFG.dstinc</td>
<td>Destination increment enable</td>
<td>Increments DMAChn_DST. This field is ignored when the DMA destination is a peripheral.</td>
</tr>
</tbody>
</table>
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 \texttt{DMACHn\_CFG.chen}, \texttt{DMACHn\_CFG.rlden} = 0, and \texttt{DMACHn\_ST.ctz\_st} = 0.
2. If using memory for the destination of the DMA transfer, configure \texttt{DMACHn\_DST} to the starting address of the destination in memory.
3. If using memory for the source of the DMA transfer, configure \texttt{DMACHn\_SRC} to the starting address of the source in memory.
4. Write the number of bytes to transfer to the \texttt{DMACHn\_CNT} register.
5. Configure the following \texttt{DMACHn\_CFG} register fields in one or more instructions. Do not set \texttt{DMACHn\_CFG.chen} to 1 or \texttt{DMA\_CNT_RLD.rlden} to 1 in this step:
   a. Configure \texttt{DMACHn\_CFG.req\_sel} to select the transfer operation associated with the DMA channel.
   b. Configure \texttt{DMACHn\_CFG.burst} for the desired burst size.
   c. Configure \texttt{DMACHn\_CFG.priority} to set the channel priority relative to other DMA channels.
   d. Configure \texttt{DMACHn\_CFG.dst\_width} to dictate the number of bytes written in each transaction.
   e. If desired, set \texttt{DMACHn\_CFG.dst\_inc} to 1 to enable automatic incrementing of the \texttt{DMACHn\_DST} register upon every AHB transaction.
   f. Configure \texttt{DMACHn\_CFG.src\_width} to dictate the number of bytes read in each transaction.
   g. If desired, set \texttt{DMACHn\_CFG.src\_inc} to 1 to enable automatic incrementing of the \texttt{DMACHn\_DST} register upon every AHB transaction.
   h. If desired, set \texttt{DMACHn\_CFG.chd\_ien} = 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 \texttt{DMACHn\_CFG.ctz\_ien} 1 to generate an interrupt when the \texttt{DMACHn\_CNT} register is decremented to zero.
6. If using the reload feature, configure the reload registers to set the destination, source, and count for the following DMA transaction.
   a. If desired, enable the channel timeout feature described in \textit{Channel Timeout Detect}. Clear \texttt{DMACHn\_CFG.to\_prescale} to 0x0 to disable the channel timeout feature.
7. Set \texttt{DMA\_CNT\_RLD.rlden} to 1 to enable the reload feature.
8. Set \texttt{DMACHn\_CFG.ch\_en} = 1 to immediately start the DMA transfer.
9. Wait for the status bit to become 0 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 \texttt{DMACHn\_CNT} is decremented to 0.
At this point, there are two possible responses depending on the value of the $\text{DMACHn\_CFG.\_rlden}$:

1. If $\text{DMACHn\_CFG.\_rlden} = 1$, then the $\text{DMACHn\_SRC}$, $\text{DMACHn\_DST}$, and $\text{DMACHn\_CNT}$ 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 $\text{DMACHn\_CFG.\_rlden} = 0$, then the channel is disabled, and $\text{DMACHn\_ST.ch_st}$ is cleared.

### 9.5 Chaining Buffers

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

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

- $\text{DMACHn\_CFG}$
- $\text{DMACHn\_SRC}$
- $\text{DMACHn\_DST}$
- $\text{DMACHn\_CNT}$
- $\text{DMAn\_SRC\_RLD}$
- $\text{DMAn\_DST\_RLD}$
- $\text{DMAn\_CNT\_RLD}$

Writing to any register while a channel is disabled is supported, but there are certain restrictions when a channel is enabled. The $\text{DMACHn\_ST.ch_st}$ 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 $\text{DMACHn\_SRC}$, $\text{DMACHn\_DST}$, or $\text{DMACHn\_CNT}$ registers while a channel is active ($\text{DMACHn\_ST.ch_st} = 1$). To disable any DMA channel, clear the $\text{DMACn\_CN.chien}$ bit. Then, poll the $\text{DMACHn\_ST.ch_st}$ bit to verify that the channel is disabled.
Figure 9-1: DMA Block-Chaining Flowchart

1. **RESET DMA**
   - \( rlden = 0 \)
   - \( chen = 0 \)
   - \( ctz_st = 1 \)

2. **CONFIGURE DMA TRANSFER PARAMETERS**
   - Set \( brst \), \( dstinc \), \( dstwd \), \( srcinc \), and \( srcwd \)

3. **CONFIGURE DMA TRANSFER PARAMETERS**
   - Set DMACHn_SRC, DMACHn_DST, and DMACHn_CNT

4. **CLEAR CTZ FLAG**
   - \( ctz_st = 1 \)

5. **WAIT FOR TRANSFER PAUSED**
   - \( ch_st = 0 \)
   - (Set \( ctzien = 1 \) to generate interrupt on channel stop.)

6. **PAUSE DMA TRANSFER**
   - \( chen = 0 \)

7. **CLEAR CTZ FLAG**
   - \( ctz_st = 1 \)

8. **WAIT FOR RELOAD TO OCCUR**
   - \( ctz_st = 1 \)
   - (Set \( ctzien = 1 \) to generate interrupt on reload.)

9. **START DMA TRANSFER**
   - \( rlden = 1 \)
   - \( chen = 1 \)

10. **START DMA TRANSFER**
    - \( rlden = 0 \)
    - \( chen = 1 \)

11. **WAIT FOR TRANSFER COMPLETE**
    - \( ch_st = 0 \)
    - (Set \( chdien = 1 \) to generate interrupt on completion.)

12. **DMA TRANSFER COMPLETE**

13. **CONFIRM DMA TRANSFER PARAMETERS**
    - DMACHn_CNT and DMA_CNT_RLD both Transferred?
    - CNT == 0?
    - Yes
    - No

14. **CONFIRM DMA TRANSFER PARAMETERS**
    - Need to send more data than specified in DMACHn_CNT?
    - Yes
    - No

15. **CONFIRM DMA TRANSFER PARAMETERS**
    - Need to send more data than specified in DMACHn_CNT?
    - Yes
    - No

16. **CONFIRM DMA TRANSFER PARAMETERS**
    - DMACHn_CNT and DMA_CNT_RLD both Transferred?
    - CNT == 0?
    - Yes
    - No

17. **CONFIRM DMA TRANSFER PARAMETERS**
    - WAIT FOR RELOAD TO OCCUR
    - \( ctz_st = 1 \)
    - (Set \( ctzien = 1 \) to generate interrupt on reload.)
9.6 DMA Interrupts

Enable interrupts for each channel by setting $\text{DMACn\_CN.chien}$. When an interrupt for a channel is pending, the corresponding $\text{DMACn\_INT.ipend} = 1$. Set the corresponding enable bit to cause an interrupt when the flag is set.

A channel interrupt ($\text{DMACHn\_ST.ipend} = 1$) is caused by:

- $\text{DMACHn\_CFG.ctzien} = 1$
  - If enabled all CTZ occurrences set the $\text{DMACHn\_ST.ipend}$ bit.
- $\text{DMACHn\_CFG.chdien} = 1$
  - If enabled, any clearing of the $\text{DMACHn\_ST.ch_st}$ bit sets the $\text{DMACHn\_ST.ipend}$ bit. Examine the $\text{DMACHn\_ST}$ register to determine which reasons caused the disable. The $\text{DMACHn\_CFG.chdien}$ bit also enables the $\text{DMACHn\_ST.to_st}$ bit. The $\text{DMACHn\_ST.to_st}$ bit does not clear the $\text{DMACHn\_ST.ch_st}$ bit.

To clear the channel interrupt, write 1 to the cause of the interrupt (the $\text{DMACHn\_ST.ctz_st}$, $\text{DMACHn\_ST.rld_st}$, $\text{DMACHn\_ST.bus_err}$, or $\text{DMACHn\_ST.to_st}$ bits).

When running in normal mode without buffer chaining ($\text{DMACHn\_CFG.rlden} = 0$), set the $\text{DMACHn\_CFG.chdien}$ bit only. An interrupt is generated upon DMA completion or an error condition (bus error or timeout error).

When running in buffer chaining mode ($\text{DMACHn\_CFG.rlden} = 1$), set both the $\text{DMACHn\_CFG.chdien}$ and $\text{DMACHn\_CFG.ctzien}$ bits. The CTZ interrupts occur on completion of each DMA (count reaches zero and reload occurs). The setting of $\text{DMACHn\_CFG.chdien}$ ensures that an error condition generates an interrupt. If $\text{DMACHn\_CFG.ctzien} = 0$, then the only interrupt occurs when the DMA completes and $\text{DMACHn\_CFG.rlden} = 0$ (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 of time. When the timeout start conditions are met, an internal 10-bit counter begins incrementing at a frequency determined by the AHB clock, $\text{DMACHn\_CFG.to_prescale}$, and $\text{DMACHn\_CFG.to_period}$ shown in Table 9-5: DMA Channel Timeout Configuration. A channel timeout event is generated if the timer is not reset by one of the events listed below before the timeout period expires.

<table>
<thead>
<tr>
<th>$\text{DMACHn_CFG.prescale}$</th>
<th>Timeout Period ($\mu$s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>$\text{Channel timeout disabled.}$</td>
</tr>
<tr>
<td>0x1</td>
<td>$2^8 \times [\text{Value from } \text{DMA_CFG.to_period}] / \text{HCLK}$</td>
</tr>
<tr>
<td>0x2</td>
<td>$2^{16} \times [\text{Value from } \text{DMA_CFG.to_period}] / \text{HCLK}$</td>
</tr>
<tr>
<td>0x3</td>
<td>$2^{24} \times [\text{Value from } \text{DMA_CFG.to_period}] / \text{HCLK}$</td>
</tr>
</tbody>
</table>
The start of the timeout period is controlled by \textit{DMACHn\_CFG\_reqwait}:

- If \textit{DMACHn\_CFG\_reqwait} = 0, the timer begins counting immediately after \textit{DMACHn\_CFG\_tose} is configured to a value other than 0x0.
- If \textit{DMACHn\_CFG\_reqwait} = 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 (\textit{DMACHn\_ST\_ch_st} = 0).

If the timeout timer period expires, hardware will set \textit{DMACHn\_ST\_to_st} = 1 to indicate a channel timeout event has occurred. A channel timeout will not disable the DMA channel.

### 9.8 Memory-to-Memory DMA

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

### 9.9 DMAC Registers

See \textit{Table 3-1: APB Peripheral Base Address Map} for this peripheral/module’s base address. If multiple instances are provided, each will have a unique base address. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific reset, if applicable.

\textit{Table 9-6: DMAC Register Summary}

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>DMACn_CN</td>
<td>DMA Control register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>DMACn_INT</td>
<td>DMA Interrupt Status register</td>
</tr>
</tbody>
</table>

### 9.10 DMAC Register Details

\textit{Table 9-7: DMACn Control Register}

<table>
<thead>
<tr>
<th>DMACn Control</th>
<th>DMACn_CN</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 31:0 | \textit{chien} | R/W | 0 | \textit{DMACn Channel Interrupt Enable} 
  
  Each bit in this field enables the corresponding channel interrupt m in DMACn\_INT. Register bits associated with unimplemented channels should not be changed from their default reset value.
  
  0: Disabled.
  
  1: Enabled. |
Table 9-8: DMACn Interrupt Register

<table>
<thead>
<tr>
<th>DMACn Interrupt</th>
<th>DMACn_INT</th>
<th>[0x0004]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>ipend</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

9.11 DMA Channel Registers

Table 9-9: Standard DMA Channel 0 to Channel 7 Register Summary

<table>
<thead>
<tr>
<th>Offset</th>
<th>DMA Channel</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0100]</td>
<td>DMACH0</td>
<td>DMACm Channel 0</td>
</tr>
<tr>
<td>[0x0120]</td>
<td>DMACH1</td>
<td>DMACm Channel 1</td>
</tr>
<tr>
<td>[0x0140]</td>
<td>DMACH2</td>
<td>DMACm Channel 2</td>
</tr>
<tr>
<td>[0x0160]</td>
<td>DMACH3</td>
<td>DMACm Channel 3</td>
</tr>
<tr>
<td>[0x0180]</td>
<td>DMACH4</td>
<td>DMACm Channel 4</td>
</tr>
<tr>
<td>[0x0200]</td>
<td>DMACH5</td>
<td>DMACm Channel 5</td>
</tr>
<tr>
<td>[0x0220]</td>
<td>DMACH6</td>
<td>DMACm Channel 6</td>
</tr>
<tr>
<td>[0x0240]</td>
<td>DMACH7</td>
<td>DMACm Channel 7</td>
</tr>
</tbody>
</table>

9.12 DMAC Channel Registers

See Table 3-1: APB Peripheral Base Address Map for the DMA Peripheral Base Address

Table 9-10: DMACH Channel Registers Summary

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>DMACHn_CFG</td>
<td>DMACHn Channel Configuration Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>DMACHn_ST</td>
<td>DMACHn Channel Status Register</td>
</tr>
<tr>
<td>[0x0008]</td>
<td>DMACHn_SRC</td>
<td>DMACHn Channel Source Register</td>
</tr>
<tr>
<td>[0x000C]</td>
<td>DMACHn_DST</td>
<td>DMACHn Channel Destination Register</td>
</tr>
<tr>
<td>[0x0010]</td>
<td>DMACHn_CNT</td>
<td>DMACHn Channel Count Register</td>
</tr>
<tr>
<td>[0x0014]</td>
<td>DMAn_SRC_RLD</td>
<td>DMACHn Channel Source Reload Register</td>
</tr>
<tr>
<td>[0x0018]</td>
<td>DMAn_DST_RLD</td>
<td>DMACHn Channel Destination Reload Register</td>
</tr>
<tr>
<td>[0x001C]</td>
<td>DMAn_CNT_RLD</td>
<td>DMACHn Channel Count Reload Register</td>
</tr>
</tbody>
</table>
### 9.13 DMA Channel Register Details

**Table 9-11: DMACHn Configuration Register**

<table>
<thead>
<tr>
<th>DMA Channel n Configuration</th>
<th>DMACHn_CFG</th>
<th>[0x0100]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31: ctzien</td>
<td>R/W</td>
<td>0</td>
<td>CTZ Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Enabled. DMACn_INT.ipend is set to 1 whenever a CTZ event occurs.</td>
</tr>
<tr>
<td>30: chdien</td>
<td>R/W</td>
<td>0</td>
<td>Channel Disable Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Enabled. DMACn_INT.ipend bit is set to 1 whenever DMACHn_ST.ch_st changes from 1 to 0.</td>
</tr>
<tr>
<td>29: Reserved</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>28:24 brst</td>
<td>R/W</td>
<td>0</td>
<td>Burst Size</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The number of bytes transferred into and out of the DMA FIFO in a single burst.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0b00000: 1 byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0b00001: 2 bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0b00010: 3 bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0b11111: 32 bytes</td>
</tr>
<tr>
<td>23: Reserved</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>22: distinc</td>
<td>R/W</td>
<td>0</td>
<td>Destination Increment Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This bit enables the automatic increment of the DMACHn_DST register upon every AHB transaction. This bit is ignored for a DMA transmit to peripherals.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>21:20 dstwd</td>
<td>R/W</td>
<td>0</td>
<td>Destination Width</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0: One byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x1: Two bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x2: Four bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x3: Reserved</td>
</tr>
<tr>
<td>19: Reserved</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>18: srinc</td>
<td>R/W</td>
<td>0</td>
<td>Source Increment on AHB Transaction Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This bit enables the automatic increment of the DMACHn_SRC register upon every AHB transaction. This bit is ignored for a DMA receive from peripherals.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>17:16 srcwd</td>
<td>R/W</td>
<td>0</td>
<td>Source Width</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Indicates the width of each AHB transaction from the source peripheral or memory. The actual width might be less than this if the DMACHn_CNT register indicates a smaller value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0: One byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x1: Two bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x2: Four bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x3: Reserved</td>
</tr>
</tbody>
</table>
### DMA Channel n Configuration

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 15:14 | pssel | R/W | 0 | **Timeout Timer Clock Pre-Scale Select**  
Selects the Pre-Scale divider for the timer clock input.  
0x0: Timer disabled.  
0x1: f\( \text{HCLK} \) / 2^8  
0x2: f\( \text{HCLK} \) / 2^16  
0x3: f\( \text{HCLK} \) / 2^24 |
| 13:11 | tosel | R/W | 0 | **Timeout Period Select**  
Selects the number of pre-scaled clocks seen by the channel timer before a timeout condition is generated. The value is approximate because of synchronization delays between timers  
0: 3-4  
1: 7-8  
2: 15-16  
3: 31-32  
4: 63-64  
5: 127-128  
6: 255-256  
7: 511-512 |
| 10 | reqwait | R/W | 0 | **Request DMA Timeout Timer Wait Enable**  
0: Start timer immediately when enabled.  
1: Delay timer start until after the first DMA transaction occurs. |
| 9:4 | reqsel | R/W | 0 | **Request Select**  
Selects the source and destination for the transfer as shown in Table 9-2: MAX32665—MAX32668 DMAC Source and Destination by Peripheral. |
| 3:2 | pri | R/W | 0 | **Channel Priority**  
Sets the priority of the channel relative to other channels of DMAm. Channels of the same priority are serviced in a round-robin fashion.  
0x0: Highest priority  
0x1: ...  
0x2: ...  
0x3: Lowest priority |
| 1 | rlden | R/W | 0 | **Reload Enable**  
Setting this bit to 1 allows reloading the DMACHn_SRC, DMACHn_DST, and DMACHn_CNT registers with their corresponding reload registers upon CTZ.  
Note: This bit is also writeable in the DMAn_CNT_RLD register. |
| 0 | chen | R/W | 0 | **Channel Enable**  
This bit is automatically cleared when DMACHn_ST.ch_st changes from 1 to 0.  
0: Disabled  
1: Enabled |

### Table 9-12: DMA Status Register

<table>
<thead>
<tr>
<th>DMA Channel n Status</th>
<th>DMACHn_ST</th>
<th>[0x0104]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 31:7 | - | RO | 0 | **Reserved for Future Use**  
Do not modify this field from its reset default value. |
### DMA Channel n Status

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>to_st</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>Timeout Status</strong>&lt;br&gt;Timeout status field. Write 1 to clear.&lt;br&gt;0: No time out. 1: A channel time out has occurred</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved</strong></td>
</tr>
<tr>
<td>4</td>
<td>bus_err</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>Bus Error</strong>&lt;br&gt;If this bit reads 1, an AHB abort occurred and the channel was disabled by hardware. Write 1 to clear.&lt;br&gt;0: No error found 1: An AHB bus error occurred</td>
</tr>
<tr>
<td>3</td>
<td>rld_st</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>Reload Status</strong>&lt;br&gt;Reload status field. Write 1 to clear.&lt;br&gt;0: Reload has not occurred. 1: Reload occurred.</td>
</tr>
<tr>
<td>2</td>
<td>ctz_st</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>CTZ Status</strong>&lt;br&gt;Write 1 to clear.&lt;br&gt;0: CTZ has not occurred. 1: CTZ has occurred.</td>
</tr>
<tr>
<td>1</td>
<td>ipend</td>
<td>RO</td>
<td>0</td>
<td><strong>Channel Interrupt Pending</strong>&lt;br&gt;0: No interrupt 1: Interrupt pending</td>
</tr>
<tr>
<td>0</td>
<td>ch_st</td>
<td>RO</td>
<td>0</td>
<td><strong>Channel Status</strong>&lt;br&gt;This bit indicates when it is safe to change the configuration, address, and count registers for the channel.&lt;br&gt;Whenever this bit is cleared by hardware, the DMACHn_CFG.chen bit is also cleared.&lt;br&gt;0: Disabled. 1: Enabled.</td>
</tr>
</tbody>
</table>

### DMA Channel n Source

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>src</td>
<td>R/W</td>
<td>0</td>
<td><strong>Source Device Address</strong>&lt;br&gt;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.&lt;br&gt;If DMACHn_CFG.srcinc = 1, then this register is incremented on each AHB transfer cycle by one, two, or four bytes depending on the data width.&lt;br&gt;If DMACHn_CFG.srcinc = 0, this register remains constant.&lt;br&gt;If a CTZ condition occurs while DMACHn_CFG.rlden = 1, then this register is reloaded with the contents of the DMAn_SRC_RLD register.</td>
</tr>
</tbody>
</table>
### Table 9-14: DMA Channel n Destination Register

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

### Table 9-15: DMA Channel n Count Register

<table>
<thead>
<tr>
<th>DMA Channel n Count</th>
<th>DMACHn_CNT</th>
<th>[0x0110]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:24</td>
<td>-</td>
<td>RO</td>
</tr>
</tbody>
</table>
| 23:0                | cnt        | R/W     | 0     | DMA Counter  
Load this register with the number of bytes to transfer. This field decreases on every AHB access to the DMA FIFO. The decrement is one, two, or four bytes depending on the data width. When the counter reaches 0, a CTZ condition is triggered.  
If a CTZ condition occurs while DMACHn_CFG.rlden = 1, then this register is reloaded with the contents of the DMAn_CNT_RLD register. |

### Table 9-16: DMA Channel n Source Reload Register

<table>
<thead>
<tr>
<th>DMA Source Reload</th>
<th>DMAn_SRC_RLD</th>
<th>[0x0114]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31</td>
<td>-</td>
<td>RO</td>
</tr>
</tbody>
</table>
| 30:0              | src_rld      | R/W     | 0     | Source Address Reload Value  
If DMACHn_CFG.rlden = 1, then the value of this register is loaded into DMACHn_SRC upon a CTZ condition. |

### Table 9-17: DMA Channel n Destination Reload Register

<table>
<thead>
<tr>
<th>DMA Destination Reload Register</th>
<th>DMAn_DST_RLD</th>
<th>[0x0118]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31</td>
<td>-</td>
<td>RO</td>
</tr>
</tbody>
</table>
| 30:0                            | dst_rld      | R/W     | 0     | Destination Address Reload Value  
If DMACHn_CFG.rlden = 1, then the value of this register is loaded into DMACHn_DST upon a CTZ condition. |
### Table 9-18: DMA Channel n Count Reload Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>rlden</td>
<td>R/W</td>
<td>0</td>
<td><strong>Reload Enable.</strong> Enables automatic loading of the DMACHn_SRC, DMACHn_DST, and DMACHn_CNT registers when a CTZ event occurs. Set this bit after the address reload registers are programmed. <strong>Note:</strong> This bit is automatically cleared to 0 when reload occurs. <strong>Note:</strong> This bit is also seen in the DMACHn_CFG register. 0: Reload disabled 1: Reload enabled</td>
</tr>
<tr>
<td>30:24</td>
<td></td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved</strong></td>
</tr>
<tr>
<td>23:0</td>
<td>cnt_rld</td>
<td>R/W</td>
<td>0</td>
<td><strong>Count Reload Value.</strong> If DMAn_CNT_RLD.rlden = 1, then the value of this register is loaded into DMACHn_CNT upon a CTZ condition.</td>
</tr>
</tbody>
</table>
10. Cyclic Redundancy Check Engine (CRC)

The Cyclic Redundancy Checks (CRCs) engine can perform CRC functions on data stored in SRAM. The CRC engine cannot be used to perform a CRC of data stored in flash memory.

An n-bit CRC can detect the following types of errors:

- Single-bit errors
- Two-bit errors for block lengths less than 2k where k is the order of the longest irreducible factor of the polynomial
- Odd numbers of errors for polynomials with the parity polynomial (x+1) as one of its factors (polynomials with an even number of terms)
- Burst errors less than n bits

Overall, all except 1 out of 2^n errors are detected:

- 99.998% for a 16-bit CRC
- 99.99999998% for a 32-bit CRC

The hardware accelerator calculates the CRC of a block of data. Data is written to the Crypto data register.

The starting initial CRC value is typically preset to all ones. If the starting initial value is preset to all zeros and an initial stream of all zeros is processed as the data, the CRC does not change.

Historically, CRCs were calculated on serial bit streams. Most serial bit streams were sent least significant bit (LSB) first. The CRC was calculated as each bit was transmitted or received. This resulted in the CRC being calculated on the LSB of the data first.

The CRC is typically appended to the end of the data. If the receiver calculates the CRC on both the data and received CRC, the result should be all zeros if the data and CRC were received error-free. Most implementations do not like to check against an all-zero checksum. Therefore, most implementations invert the CRC before transmitting it. By inverting the CRC on the transmitting end, the resulting CRC on the receiving end should be a constant. The specific constant is dependent upon the CRC polynomial. This works because the non-inverted CRC calculated at the end of the data XOR'd with the received inverted CRC is all ones (CRC ⊕ ~CRC = 1s). Shifting all ones through the polynomial results in the same constant for each message, and the constant is dependent upon the polynomial.

Because the receiving end calculates a new CRC on both the data and received CRC, you must send the received CRC in the correct order, so the highest-order term of the CRC is shifted through the generator first. Because data is typically shifted through the generator LSB first, this means the CRC is reversed bitwise, with the highest-order term of the remainder in the LSB position. Software CRC algorithms typically handle this by calculating everything backwards. They reverse the polynomial and do right shifts on the data. The resulting CRC ends up being bit swapped and in the correct format.

The CRYPTO_CRC register is preset to all ones if the crypto block is reset. The initial CRC state is written to any value. The final inversion must be done by software if required.

The CRC generator has a programmable polynomial up to 32 bits. The polynomial should be written to the CRC_POLY register. The largest term x^n defines the length of the CRC. 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 ... x^32).

CRC polynomials with good error-detection properties should be irreducible (the polynomial should not be factorable). Therefore, the constant term x^0 or 1 should always be present, otherwise, the polynomial would be factorable by x. If the constant term x^0 or 1 were not present, the resulting CRC would be cyclic with a subgroup smaller than x^n. The effective length of the CRC would be the difference between the highest- and lowest-order terms. Therefore, the highest- and lowest-order terms x^n and x^0 should always appear in the polynomial.
When found in literature, sometimes the LSB or MSB of the polynomial is omitted when the polynomial is written in binary. It is more common to see CRC polynomials with the MSB implied because that is the bit that is shifted off, XOR’d with the data, and tested to see if the result is set. Some literature assumes the reader knows that an \( n \)-bit CRC must have the \( x^n \) term set, or else it would be a smaller length CRC.

Some common CRC polynomials and their check constants are shown in Table 10-1. The polynomial register resets to the 32-bit CRC polynomial used by Ethernet, PPP, and file compression utilities such as zip or gzip.

By default, the CRC accelerator does right shifts and calculates the CRC on the LSB of the data first. The CRC can be calculated on the most significant bit (MSB) of the data first by setting the bit-swap control bit to 1 (\( CRC\_CTRL.msb = 1 \)). To calculate the CRC MSB first, you must left justify the polynomial in the \( CRC\_POLY \) register. The hardware implies the MSB of the polynomial just as it did when shifting the LSB first. The LSB of the polynomial should be set, this defines the length of the CRC. The initial state of the CRC should also be left justified. When the CRC calculation is complete, it is necessary to right shift the CRC to right justify it if the polynomial is less than 32 bits.

Table 10-1: Common CRC Polynomials

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Polynomial Expression</th>
<th>Order</th>
<th>Polynomal (( CRC_POLY ))</th>
<th>Check</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC-32 Ethernet</td>
<td>( x^{32} + x^{26} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + x^0 )</td>
<td>LSB</td>
<td>0xEDB8 8320</td>
<td>0xDEBB 20E3</td>
</tr>
<tr>
<td>CRC-CCITT</td>
<td>( x^{16} + x^{12} + x^5 + x^0 )</td>
<td>LSB</td>
<td>0x0000 8408</td>
<td>0x0000 F088</td>
</tr>
<tr>
<td>CRC-16</td>
<td>( x^{16} + x^{15} + x^2 + x^0 )</td>
<td>LSB</td>
<td>0x0000 A001</td>
<td>0x0000 8001</td>
</tr>
<tr>
<td>USB Data</td>
<td>( x^{16} + x^{15} + x^2 + x^0 )</td>
<td>MSB</td>
<td>0x8005 0000</td>
<td>0x800D 0000</td>
</tr>
<tr>
<td>Parity</td>
<td>( x^1 + x^0 )</td>
<td>LSB</td>
<td>0x0000 0001</td>
<td></td>
</tr>
</tbody>
</table>

10.1 Instances

There is one instance of this peripheral.

10.2 Linear Feedback Shift Register (LFSR)

Linear Feedback Shift Registers (LFSRs) are commonly used to implement Pseudo-Random Number Generators (PRNGs). An LFSR polynomial can be written to the \( CRC\_POLY \) register to generate pseudo-random data. The starting state or seed for the pseudo-random sequence should be written to the CRC register. The lockup state of all zeros is detected, and the LFSR substitutes the value 1 to prevent lockup.
Different polynomials generate different sequences of random data. Ideally, an n-bit polynomial generates a random sequence of $2^n - 1$ bits. Not all polynomials are maximal length. Some repeat before the theoretical maximum length of $2^n - 1$. There are thousands of different maximal length 32-bit LFSR polynomials. You can use any length of an LFSR polynomial up to 32 bits. Some tables of maximal length LFSR polynomials omit the MSB ($x^n$) term or the LSB ($x^0 = 1$) term. Fibonacci LFSRs feed back the XOR of all the taps to the constant term $x^0 = 1$. It is often implied when listing the taps but must be present when writing the polynomial to the CRC_POLY register.

The crypto accelerator automatically generates the next sequence of 32 bits whenever the CRYPTO_LFSR register is read. If the PRNG control bit is set, the incoming data is forced to zero. You can use the DMA to quickly fill a block of memory with pseudo-random data.

### 10.3 Registers

See Table 3-1: APB Peripheral Base Address Map for this peripheral/module's base address. If multiple instances are provided, each will have a unique base address. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific reset, if applicable.

#### Table 10-2: CRC Register Summary

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register Name</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>CRYPTO_CTRL</td>
<td>R/W</td>
<td>Crypto Control Register</td>
</tr>
<tr>
<td>0x000C</td>
<td>CRC_CTRL</td>
<td>R/W</td>
<td>CRC Control Register</td>
</tr>
<tr>
<td>0x0010</td>
<td>CRC_DMA_SRC</td>
<td>R/W</td>
<td>CRC DMA Source Address Register</td>
</tr>
<tr>
<td>0x0014</td>
<td>CRC_DMA_DST</td>
<td>R/W</td>
<td>CRC DMA Destination Address Register</td>
</tr>
<tr>
<td>0x0018</td>
<td>CRC_DMA_CNT</td>
<td>R/W</td>
<td>CRC DMA Byte Count Register</td>
</tr>
<tr>
<td>0x0020</td>
<td>CRC_DATA_IN0</td>
<td>R/W</td>
<td>CRC Data Input Register 0 (Bits 31:0)</td>
</tr>
<tr>
<td>0x0024</td>
<td>CRC_DATA_IN1</td>
<td>R/W</td>
<td>CRC Data Input Register 0 (Bits 63:32)</td>
</tr>
<tr>
<td>0x0028</td>
<td>CRC_DATA_IN2</td>
<td>R/W</td>
<td>CRC Data Input Register 0 (Bits 95:64)</td>
</tr>
<tr>
<td>0x002C</td>
<td>CRC_DATA_IN3</td>
<td>R/W</td>
<td>CRC Data Input Register 0 (Bits 127:96)</td>
</tr>
<tr>
<td>0x0030</td>
<td>CRC_DATA_OUT0</td>
<td>R/W</td>
<td>CRC Data Output Register 0 (Bits 31:0)</td>
</tr>
</tbody>
</table>
### Offset | Register Name | Access | Description
---|---|---|---
[0x0034] | CRC_DATA_OUT1 | R/W | CRC Data Output Register 0 (Bits 63:32)
[0x0038] | CRC_DATA_OUT2 | R/W | CRC Data Output Register 0 (Bits 95:64)
[0x003C] | CRC_DATA_OUT3 | R/W | CRC Data Output Register 0 (Bits 127:96)
[0x0040] | CRC_POLY | R/W | CRC Polynomial Register
[0x0044] | CRC_VAL | R/W | CRC Value Register
[0x0048] | CRC_PRNG | R/W | CRC Pseudo-Random Number Register

### 10.4 Register Details

#### Table 10-3. Flash Controller Address Pointer Register

<table>
<thead>
<tr>
<th>Crypto Control Register</th>
<th>CRYPTO_CTRL</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 31 | done | RO | 0 | **Done**  
Done bit indicator.  
0: CRC not done.  
1: CRC done. |
| 30 | rdy | RO | 1 | **Ready**  
CRC engine ready for operation.  
0: CRC engine busy.  
1: CRC engine ready for operation. |
| 29 | err | RO | 0 | **Error Flag**  
If this field reads 1 an error occurred.  
0: No error condition.  
1: Error occurred. |
| 28:25 | - | RO | 0 | **Reserved for Future Use**  
Do not modify this field. |
| 24 | dma_done | R/W | 0 | **DMA Complete Flag**  
This field is set to 1 when a DMA read/write operation is complete. Set this field to 0 prior to starting a DMA CRC operation. |
| 23:16 | - | R/O | 0 | **Reserved for Future Use**  
Do not modify this field. |
| 15 | dmadne_msk | R/W | 0 | **DMA Done Flag Mask for DMA Operation**  
This field sets the behavior of the CRYPTO_CTRL.dma_done flag. Setting this bit to 1 results in a CRYPTO_CTRL.dma_done complete to set the CRYPTO_CTRL.dma_done flag. The setting for this field does not affect the actual behavior of the CRYPTO_CTRL.dma_done flag.  
0: DMA Complete Flag is not used for setting CRYPTO_CTRL.dma_done field.  
1: DMA Complete condition sets the CRYPTO_CTRL.dma_done field. |
| 14 | flag_mode | R/W | 0 | **Done Flag Mode**  
This field configures the access behavior of the CRYPTO_CTRL.dma_done bit. When this field is set to 1, the CRYPTO_CTRL.dma_done bit is Write 1 to Clear. When this field is 0, the CRYPTO_CTRL.dma_done bit is unrestricted read/write.  
0: CRYPTO_CTRL.dma_done bit is unrestricted read/write access.  
1: CRYPTO_CTRL.dma_done bit is Write 1 to Clear.  
Note: This field is only reset on a Power-On Reset. |
<table>
<thead>
<tr>
<th>Crypto Control Register</th>
<th></th>
<th></th>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bits</strong></td>
<td><strong>Name</strong></td>
<td><strong>Access</strong></td>
<td><strong>Reset</strong></td>
<td><strong>CRYPTO_CTRL [0x0000]</strong></td>
</tr>
</tbody>
</table>
| 13:12 | - | R/W | 0 | Reserved for Future Use  
Do not modify this field. |
| 11:10 | rdsrc | R/W | 0 | Read FIFO Source Select  
This field selects the source of the read FIFO. The default is for the Read FIFO to not use DMA as its source buffer.  
0b00: DMA disabled.  
0b01: DMA or Direct via CRC_DATA_IN3:CRC_DATA_IN0 registers.  
0b10: CRC_PRNG  
0b11: Reserved for Future Use. |
| 9:8 | wrsrc | R/W | 0 | Write FIFO Source Select  
This field selects the source of the Write FIFO. The default is for the Write FIFO to not use DMA as its source buffer.  
0b00: DMA disabled.  
0b01: DMA or direct via CRC_DATA_OUT3:CRC_DATA_OUT0 registers.  
0b10: CRC_PRNG  
0b11: Reserved for Future Use. |
| 7:6 | - | R/W | 0 | Reserved for Future Use  
Do not modify this field. |
| 5 | bsi | R/W | 0 | Byte Swap Input  
Enables input word byte swapping if set.  
0: Input bytes are not byte swapped.  
1: Input bytes are byte swapped.  
*Note: Byte swap only occurs on full words.* |
| 4 | bso | R/W | 0 | Byte Swap Output  
Enable output word byte swapping if set.  
0: Output bytes are not byte swapped.  
1: Output bytes are byte swapped.  
*Note: Byte swap only occurs on full words.* |
| 3 | - | R/W | 0 | Reserved for Future Use  
Do not modify this field. |
| 2 | src | R/W | 0 | Source Select  
Selects the CRC generator input source.  
0: CRC uses input FIFO.  
1: CRC uses output FIFO. |
| 1 | int | R/W | 0 | Interrupt Enable  
Set this field to 1 to generate a CRC/Crypto IRQ when an interrupt flag is set (done or error).  
0: Interrupt disabled.  
1: Interrupt enabled. |
| 0 | rst | R/W | 0 | Reset  
Set this field to reset the CRC including the FIFOs. The **CRYPTO_CTRL** register is not reset.  
0: Reset not active.  
1: Reset the CRC Engine. |
Table 10-3: CRC Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:2</td>
<td></td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>1</td>
<td>msb</td>
<td>R/W</td>
<td>0</td>
<td>MSB Select</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set the order of calculating the CRC on the input data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: LSB data first.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: MSB data first.</td>
</tr>
<tr>
<td>0</td>
<td>crc_en</td>
<td>R/W</td>
<td>0</td>
<td>CRC Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Enable the CRC function.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: CRC function disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: CRC function enable.</td>
</tr>
</tbody>
</table>

Table 10-4: CRC DMA Source Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>src_addr</td>
<td>R/W</td>
<td>0</td>
<td>DMA Source Address</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set this field to the address of the DMA source input data.</td>
</tr>
</tbody>
</table>

Table 10-5: CRC DMA Destination Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>dst_addr</td>
<td>R/W</td>
<td>0</td>
<td>DMA Destination Address</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set this field to the address for the CRC DMA output data.</td>
</tr>
</tbody>
</table>

Table 10-6: CRC DMA Count Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>count</td>
<td>R/W</td>
<td>0</td>
<td>DMA Byte Count</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set this field to the number of bytes of input data to the CRC engine.</td>
</tr>
</tbody>
</table>
Table 10-7: CRC Data Input Registers

| CRC Data Input Register 0          | CRC_DATA_IN0       | [0x0020] |
| CRC Data Input Register 1          | CRC_DATA_IN1       | [0x0024] |
| CRC Data Input Register 2          | CRC_DATA_IN2       | [0x0028] |
| CRC Data Input Register 3          | CRC_DATA_IN3       | [0x002C] |

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0 | data   | R/W    | 0     | CRC Data Input
Writes to this register bypass the DMA interface and put data directly into the CRC FIFO. Data written to a register is placed into the FIFO in the order it is written. CRC_DATA_IN3:CRC_DATA_IN0 represent 128 bits with CRC_DATA_IN3 mapped to bits 127:96, CRC_DATA_IN2 mapped to bits 95:64, CRC_DATA_IN1 mapped to bits 63:32 and CRC_DATA_IN0 mapped to bits 31:0.
CRC_DATA_IN3:CRC_DATA_IN0 occupy four successive words to allow the use of multi-store instructions. Writes to any location are supported and writes are loaded into the CRC FIFO in the order they are written.
Note: The CRC_DATA_INn registers are affected by the input endian swap bit, CRYPTO_CTRL.bsi.

Table 10-8: CRC Data Output Registers

| CRC Data Output Register 0          | CRC_DATA_OUT0      | [0x0030] |
| CRC Data Output Register 1          | CRC_DATA_OUT1      | [0x0034] |
| CRC Data Output Register 2          | CRC_DATA_OUT2      | [0x0038] |
| CRC Data Output Register 3          | CRC_DATA_OUT3      | [0x003C] |

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0 | data   | R/W    | 0     | CRC Data Output
The output from the CRC is written to the CRC_DATA_OUT3:CRC_DATA_OUT0 registers representing a total of 128 bits. CRC_DATA_OUT3 maps to bits 127:96, CRC_DATA_OUT2 maps to bits 95:64, CRC_DATA_OUT1 maps to bits 63:31, and CRC_DATA_OUT0 represents output bits 31:0.
Note: The CRC_DATA_OUTn registers are affected by the output endian swap bit, CRYPTO_CTRL.bso.

Table 10-9: CRC Polynomial Register

| CRC Polynomial Register          | CRC_POLY          | [0x0040] |

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0 | src_addr | R/W    | 0     | CRC Polynomial
The polynomial used for Galois Field calculations (CRC or LFSR) is written to this register.
Note: This register is affected by the MSB control bit, CRC_CTRL.msb.
### Table 10-10: CRC Value Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>val</td>
<td>R/W</td>
<td>0</td>
<td>CRC Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This is the state for the Galois Field. Output of the CRC calculation or the current state of the LFSR.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: This register is affected by the MSB control bit, CRC_CTRL.msb.</td>
</tr>
</tbody>
</table>

### Table 10-11: CRC Pseudo-Random Number Generator Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>prng</td>
<td>R/W</td>
<td>0</td>
<td>CRC Pseudo-Random Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Output of the Galois Field shift register. This register holds the resulting pseudo-random number.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: This register is affected by the MSB control bit, CRC_CTRL.msb.</td>
</tr>
</tbody>
</table>
11. Analog to Digital Converter and Comparators (ADC)

The Analog to Digital Converter (ADC) is a 10-bit sigma-delta ADC with a single-ended input multiplexer and an integrated reference generator. The multiplexer selects an input channel from either of the 8 external analog input signals or the internal power supply inputs. The external analog input signals are defined as alternate functions on GPIO as shown in **Table 6-2: MAX32665—MAX32668 GPIO and Alternate Function Matrix, 140 WLP**. The 10-bit ADC conversions are stored as a 16-bit value selectable as most-significant bit (MSB) or least-significant bit (LSB) aligned. The 8 external analog inputs can be configured as 4 two-input comparators with interrupt capabilities.

11.1 Features

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

11.2 Instances

Table 6-1 lists the locations of the External ADC inputs for each package.

**Table 11-1: MAX32665—MAX32668 ADC Peripheral Pins**

<table>
<thead>
<tr>
<th>ALTERNATE FUNCTION</th>
<th>MAPPING</th>
<th>109 WLP</th>
<th>121 CTBGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIN0/AIN0N</td>
<td>AF1</td>
<td>P0.16</td>
<td>P0.16</td>
</tr>
<tr>
<td>AIN1/AIN0P</td>
<td>AF1</td>
<td>P0.17</td>
<td>P0.17</td>
</tr>
<tr>
<td>AIN2/AIN1N</td>
<td>AF1</td>
<td>P0.18</td>
<td>P0.18</td>
</tr>
<tr>
<td>AIN3/AIN1P</td>
<td>AF1</td>
<td>P0.19</td>
<td>P0.19</td>
</tr>
<tr>
<td>AIN4/AIN2N</td>
<td>AF1</td>
<td>P0.20</td>
<td>P0.20</td>
</tr>
<tr>
<td>AIN5/AIN2P</td>
<td>AF1</td>
<td>P0.21</td>
<td>P0.21</td>
</tr>
<tr>
<td>AIN6/AIN3N</td>
<td>AF1</td>
<td>P0.22</td>
<td>P0.22</td>
</tr>
<tr>
<td>AIN7/AIN3P</td>
<td>AF1</td>
<td>P0.23</td>
<td>P0.23</td>
</tr>
</tbody>
</table>
### 11.3 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 11-1*. Details of selecting the ADC clock frequency, $f_{adcclk}$, are covered in the *Clock Configuration* section.

*Equation 11-1: ADC 10-bit Word Sample Rate*

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

ADC offset and gain errors are factory trimmed and automatically loaded into the ADC controller during system power-up. Gain error is trimmed to null out the total errors of the ADC and internal reference.

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 $2\times$ to provide additional conversion resolution if the input signals are less than half the reference voltage.
Figure 11-1: Analog to Digital Converter Block Diagram
11.4 Clock Configuration

The ADC clock, `adcclk`, is controlled by the `GCR_PCLK_DIV.adcfrq` register field. Configure this field for the target ADC sample frequency. The maximum clock supported by the ADC is 8MHz. The divisor selection, `GCR_PCLK_DIV.adcfrq`, for the ADC depends on the peripheral clock. Equation 11-2 shows the calculation for the ADC clock frequency, where:

\[ f_{PCLK} = \frac{f_{SYSCLK}}{2} \]

Equation 11-2: ADC Clock Frequency

\[ f_{adcclk} = \frac{f_{PCLK}}{GCR_{PCKDIV}.adcfrq} \]

The `GCR_PCLK_DIV.adcfrq` register field setting must result in a value for `f_{adcclk} \leq 8NHZ` as shown in Table 11-2 with the System Clock set as the 96MHz high frequency oscillator.

**Table 11-2: ADC Clock Frequency and ADC Conversion Time (f_{SYSCLK} = 96MHz, f_{PCLK} = 48MHz)**

<table>
<thead>
<tr>
<th><code>GCR_{PCKDIV}.adcfrq[3:0]</code></th>
<th>ADC Clock Frequency (Hz) ( f_{adcclk} )</th>
<th>10-Bit Word Conversion Time (( \mu )s) ( t_{adc,sample} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0–0x7</td>
<td>Invalid</td>
<td>Invalid</td>
</tr>
<tr>
<td>0x8</td>
<td>6,000,000</td>
<td>171</td>
</tr>
<tr>
<td>0x9</td>
<td>5,333,333</td>
<td>192</td>
</tr>
<tr>
<td>0xA</td>
<td>4,800,000</td>
<td>214</td>
</tr>
<tr>
<td>0xB</td>
<td>4,363,636</td>
<td>235</td>
</tr>
<tr>
<td>0xC</td>
<td>4,000,000</td>
<td>256</td>
</tr>
<tr>
<td>0xD</td>
<td>3,692,308</td>
<td>278</td>
</tr>
<tr>
<td>0xE</td>
<td>3,428,571</td>
<td>299</td>
</tr>
<tr>
<td>0xF</td>
<td>3,200,000</td>
<td>320</td>
</tr>
</tbody>
</table>
11.5 Power-Up Sequence

Complete the following steps to configure the ADC:

1. Disable the ADC clock by setting `ADC_CTRL.clk_en` to 0.
2. Set the ADC clock (adcclk) using `GCR_PCLK_DIV.adcfreq`. See Clock Configuration.
3. Enable the ADC clock by setting `ADC_CTRL.clk_en` to 1.
4. Clear the ADC reference ready interrupt flag by writing a 1 to `ADC_INTR.ref_ready_if`.
5. Optionally enable the ADC reference ready interrupt (`ADC_INTR.ref_ready_ie = 1`), and enable the ADC interrupt vector (ADC IRQ).
6. Select one of the following ADC reference sources:
   a. Internal 1.22V bandgap reference (`ADC_CTRL.ref_sel = 0`).
   b. VDDA reference (`ADC_CTRL.ref_sel = 1`).
7. Complete the following steps to enable power:
   a. Set `ADC_CTRL.pwr` to 1 to turn on the ADC.
   b. Set `ADC_CTRL.refbuf_pwr` to 1 to turn on the internal reference buffer if using the internal bandgap reference.
   c. Set `ADC_CTRL.chargepump_pwr` to 1 to turn on the ADC charge pump. Note: The ADC charge pump takes approximately 10µs to fully charge and stabilize.
   d. Wait until hardware sets the `ADC_INTR.ref_ready_if` bit to 1 indicating the charge pump is fully stabilized.
   e. Clear the ADC reference ready interrupt flag by writing 1 to `ADC_INTR.ref_ready_if`.
   f. Optionally disable the ADC reference ready interrupt (`ADC_INTR.ref_ready_ie = 0`).

11.6 Conversion

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

1. Select the ADC input channel for the conversion by setting `ADC_CTRL.ch_sel` field. See ADC Channel Select for details.
2. Optionally set input and reference scaling. See Reference Scaling and Input Scaling for details on each input channel’s scale requirements.
3. Set the data alignment for the conversion output data using the `ADC_CTRL.data_align` field, 0 for LSB alignment or 1 for MSB alignment. See Table 11-4 for alignment details of the DATA register.
4. Clear the ADC done interrupt flag by writing 1 to the `ADC_INTR.done_if`.
5. Optionally enable the ADC done interrupt (`ADC_INTR.done_ie = 1`), and enable the ADC interrupt vector (ADC IRQ). See the Interrupt chapter for details.
6. Start the ADC conversion by setting `ADC_CTRL.start` to 1.
7. Poll the `ADC_INTR.done_if` flag until you read 1, or wait for the ADC interrupt to occur if enabled in step 5.
8. Read the data from the `ADC_DATA.data`, and clear the ADC done interrupt flag by writing 1 to `ADC_INTR.done_if`.

11.7 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.input_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.
11.7.1 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 \textit{ADC\_CTRL}.\textit{adc\_divsel}. The scale settings for the given input signal and reference must satisfy the following equation to be valid:

\textit{Equation 11-3: Input and Reference Scale Requirements Equation}

\[
\frac{\text{AIN}_n}{2^{\text{input\_scale}}} < \frac{\text{VREF}}{2^{\text{ref\_scale}}}
\]

11.7.2 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 \textit{Table 11-3}, below.

\textit{Table 11-3: Input and Reference Scale Support by ADC Input Channel}

<table>
<thead>
<tr>
<th>ADC Channel</th>
<th>ADC Input Signal</th>
<th>ADC_CTRL input_scale</th>
<th>ADC_CTRL ref_scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>\textit{V}\textit{COREA}</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>\textit{V}\textit{COREB}</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>\textit{V}\textit{RXOUT}</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>\textit{V}\textit{TXOUT}</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>\textit{V}\textit{DDA}</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>\textit{V}\textit{DDB}\textstyle/4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>\textit{V}\textit{DDIO}\textstyle/4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>\textit{V}\textit{DDIOH}\textstyle/4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>\textit{V}\textit{REG}/4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

11.7.3 Data Conversion Output Alignment

The ADC outputs a total of 10-bits per conversion and stores the data in the DATA register LSB justified by default. \textit{Table 11-4} shows the ADC data alignment based on the value of the \textit{ADC\_CTRL}.\textit{data\_align} bit.
Table 11-4: ADC Data Register Alignment Options

<table>
<thead>
<tr>
<th>ADC_CTRL.data_align = 0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MSB</td>
<td>LSB</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADC_CTRL.data_align = 1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MSB</td>
<td>LSB</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

11.7.4 Data Conversion Value Equations

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

Equation 11-4: ADC Data Calculation for Input Signal \( \text{ADC_CTRL.adc_ch_sel} = 0x00 \) thru \( 0x07 \) (AIN0 – AIN7)

\[
ADC_{\text{DATA}} = \text{round} \left\{ \left( \frac{\text{Input Signal}}{2^{\text{input\_scale}} \times (\text{adc\_divsel} + 1)} \right) \times \left( \frac{V_{\text{REF}}}{2^{\text{ref\_scale}}} \right) \right\} \times (2^{10} - 1)
\]

Note: Must satisfy Equation 11-3.

Equation 11-5: ADC Data Equation for Input Signal \( \text{ADC_CTRL.adc_ch_sel} = 0x08 \) thru \( 0x0C \) (VCOREA, VCOREB, V_RXOUT, V_TXOUT, V_{\text{DDA}})

\[
ADC_{\text{DATA}} = \text{round} \left\{ \left( \frac{\text{Input Signal}}{2^{\text{input\_scale}}} \right) \times \left( \frac{V_{\text{REF}}}{2^{\text{ref\_scale}}} \right) \right\} \times (2^{10} - 1)
\]

Note: See Table 11-3 for limitations.

Equation 11-6: ADC Data Calculation Input Signal \( \text{ADC_CTRL.adc_ch_sel} = 0x0D \) thru \( 0x10 \) (V_{DDB}, V_{D Dio}, V_{D DioH}, V_{REGI})

\[
ADC_{\text{DATA}} = \text{round} \left\{ \left( \frac{\text{Input Signal}}{2^{\text{input\_scale}}} \right) \times \left( \frac{V_{\text{REF}}}{2^{\text{ref\_scale}}} \right) \right\} \times (2^{10} - 1)
\]

Note: See Table 11-3 for limitations.
### 11.7.5 Data Limits and Out of Range Interrupts

Channel limits are implemented to minimize power consumption for power supply monitoring. The ADC includes four limit registers, ADC\_LIMIT0 to ADC\_LIMIT3, that you can use to set a high limit, low limit, and the ADC channel number to apply the limits against. A block diagram of the limit engine for each of the four limit registers is shown in Figure 11-2.

**Figure 11-2: 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 interrupts 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_LIMIT0.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.

### 11.7.6 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.chargepump_pwr` to 0, disabling the ADC charge pump.
4. Set `ADC_CTRL.clk_en` to 0, disabling the ADC internal clock.

### 11.8 Comparator Operation

Each comparator is individually enabled using `MCR_AINCOMP.aincompnd`. When the inputs to any comparator cross their bias potential, the corresponding flag bit in the `PWRSEQ_LPPWST` register will be set. Should the user desire, an interrupt can be generated by setting the corresponding bit in the `PWRSEQ_LPPWEN` register. The interrupts must be globally enabled by setting the `GCR_PMR.compwken` bit.
11.9 Registers

See Table 3-1: APB Peripheral Base Address Map for the base peripheral address of these registers. All fields are reset on peripheral, system, or power-on reset events unless otherwise specified.

**Table 11-5. ADC Registers Summary**

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

11.10 Register Details

**Table 11-6: ADC Control Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:21</td>
<td></td>
<td>RO</td>
<td>0x050</td>
<td>Reserved for Future Use&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>20</td>
<td>data_align</td>
<td>R/W</td>
<td>0</td>
<td><strong>ADC Data Alignment</strong>&lt;br&gt;Selects the alignment of the 16-bit data conversion stored in the DATA register.&lt;br&gt;0: Data is LSB justified in 16-bit DATA register. DATA[15:10] = 0.&lt;br&gt;1: Data is MSB justified in 16-bit DATA register. DATA[5:0] = 0.</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>18:17</td>
<td>adc_divsel</td>
<td>R/W</td>
<td>0</td>
<td><strong>External Input Scale</strong>&lt;br&gt;Scales the external inputs AIN0-AIN7. All eight of external inputs are scaled by the same value&lt;br&gt;0x0: No scaling,&lt;br&gt;0x1: Divide by 2&lt;br&gt;0x2: Divide by 3&lt;br&gt;0x3: Divide by 4</td>
</tr>
</tbody>
</table>
### ADC Control

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16:12</td>
<td>adc_ch_sel</td>
<td>R/W</td>
<td>0</td>
<td><strong>ADC Channel Select</strong>&lt;br&gt;Selects the active channel for the next ADC conversion.</td>
</tr>
<tr>
<td></td>
<td>adc_ch_sel</td>
<td>ADC Input Channel</td>
<td>Input</td>
<td></td>
</tr>
<tr>
<td>0x00</td>
<td>0</td>
<td>0</td>
<td>AIN0</td>
<td></td>
</tr>
<tr>
<td>0x01</td>
<td>1</td>
<td>1</td>
<td>AIN1</td>
<td></td>
</tr>
<tr>
<td>0x02</td>
<td>2</td>
<td>2</td>
<td>AIN2</td>
<td></td>
</tr>
<tr>
<td>0x03</td>
<td>3</td>
<td>3</td>
<td>AIN3</td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>4</td>
<td>4</td>
<td>AIN4</td>
<td></td>
</tr>
<tr>
<td>0x05</td>
<td>5</td>
<td>5</td>
<td>AIN5</td>
<td></td>
</tr>
<tr>
<td>0x06</td>
<td>6</td>
<td>6</td>
<td>AIN6</td>
<td></td>
</tr>
<tr>
<td>0x07</td>
<td>7</td>
<td>7</td>
<td>AIN7</td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>8</td>
<td>8</td>
<td>VDDRA</td>
<td></td>
</tr>
<tr>
<td>0x09</td>
<td>9</td>
<td>9</td>
<td>VDDRB</td>
<td></td>
</tr>
<tr>
<td>0x0A</td>
<td>10</td>
<td>10</td>
<td>VDDOUT</td>
<td></td>
</tr>
<tr>
<td>0x0B</td>
<td>11</td>
<td>11</td>
<td>VTXOUT</td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>12</td>
<td>12</td>
<td>VDDA</td>
<td></td>
</tr>
<tr>
<td>0x0D</td>
<td>13</td>
<td>13</td>
<td>VDDB  / 4</td>
<td></td>
</tr>
<tr>
<td>0x0E</td>
<td>14</td>
<td>14</td>
<td>VDDIO / 4</td>
<td></td>
</tr>
<tr>
<td>0x0F</td>
<td>15</td>
<td>15</td>
<td>VDDIOH / 4</td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>16</td>
<td>16</td>
<td>VDDH</td>
<td></td>
</tr>
<tr>
<td>0x11 – 0x1F</td>
<td>Reserved for Future Use</td>
<td>Reserved for Future Use</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>clk_en</td>
<td>R/W</td>
<td>0</td>
<td><strong>ADC Clock Enable</strong>&lt;br&gt;0: Disabled&lt;br&gt;1: Enabled</td>
</tr>
<tr>
<td>10</td>
<td>ref_sel</td>
<td>R/W</td>
<td>0</td>
<td><strong>ADC Reference Select</strong>&lt;br&gt;0: Internal bandgap reference.&lt;br&gt;1: VDDA</td>
</tr>
<tr>
<td>9</td>
<td>input_scale</td>
<td>R/W</td>
<td>0</td>
<td><strong>ADC Input Scalr</strong>&lt;br&gt;0: No scaling.&lt;br&gt;1: Input is scaled by ½.</td>
</tr>
<tr>
<td></td>
<td>ref_scale</td>
<td>R/W</td>
<td>0</td>
<td><strong>ADC Reference Scaler</strong>&lt;br&gt;0: No scaling.&lt;br&gt;1: Reference is scaled by ½.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><em>Note: See Reference Scaling and Input Scaling for valid settings for each ADC input</em></td>
</tr>
<tr>
<td>7:5</td>
<td></td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>4</td>
<td>chgpump_pwr</td>
<td>R/W</td>
<td>0</td>
<td><strong>ADC Charge Pump Enable</strong>&lt;br&gt;Setting this bit to 1 turns on the ADC charge pump required to perform ADC conversions.&lt;br&gt;0: Disabled&lt;br&gt;1: Enabled</td>
</tr>
<tr>
<td>3</td>
<td>refbuf_pwr</td>
<td>R/W</td>
<td>0</td>
<td><strong>Reference Buffer Power Enable</strong>&lt;br&gt;0: Disabled&lt;br&gt;1: Enabled</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>1</td>
<td>pwr</td>
<td>R/W</td>
<td>0</td>
<td><strong>ADC Power Enable</strong>&lt;br&gt;0: Disabled&lt;br&gt;1: Enabled</td>
</tr>
</tbody>
</table>
### Table 11-7: ADC Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:4</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td>overflow</td>
<td>RO</td>
<td>0</td>
<td>ADC Overflow Flag</td>
</tr>
<tr>
<td></td>
<td>pwr_up_active</td>
<td>RO</td>
<td>0</td>
<td>ADC Power-Up State</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td>0</td>
<td>active</td>
<td>RO</td>
<td>0</td>
<td>ADC Conversion in Progress</td>
</tr>
</tbody>
</table>

#### Description
- **Start ADC Conversion**: Write this bit to 1 to start an ADC conversion. When the conversion is complete, the hardware automatically sets this bit to 0 indicating the conversion is complete.
  - 0: ADC inactive or data conversion complete.
  - 1: Start ADC conversion and remains set until complete.

- **ADC Overflow Flag**: 0: No overflow on last conversion 1: Overflow on last conversion

- **ADC Power-Up State**: This field is set to 1 when the ADC charge pump is powering up.
  - 0: AFE is not in power-up delay.
  - 1: AFE is currently in the power-up delay state.

- **ADC Conversion in Progress**: 0: ADC is idle 1: ADC conversion is in progress

### Table 11-8: ADC Data Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:0</td>
<td>data</td>
<td>RO</td>
<td>0</td>
<td>ADC Data</td>
</tr>
</tbody>
</table>

#### Description
- **ADC Data**: This field contains the ADC conversion output data. See the [Data Conversion Output Alignment](#) for details.

### Table 11-9: ADC Interrupt Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:23</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td>22</td>
<td>pending</td>
<td>RO</td>
<td>0</td>
<td>ADC Interrupt Pending</td>
</tr>
<tr>
<td>21</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td>20</td>
<td>overflow_if</td>
<td>R/W1C</td>
<td>0</td>
<td>ADC Overflow Interrupt Flag</td>
</tr>
<tr>
<td>19</td>
<td>adc_lo_limit_if</td>
<td>R/W1C</td>
<td>0</td>
<td>ADC Low Limit Interrupt Flag</td>
</tr>
</tbody>
</table>

#### Description
- **Reserved for Future Use**: Do not modify this field.
- **ADC Interrupt Pending**: 0: No ADC interrupt pending. 1: At least one ADC interrupt is pending, and the corresponding interrupt enable bit is set.
- **Reserved for Future Use**: Do not modify this field.
- **ADC Overflow Interrupt Flag**: 1: The last conversion resulted in an overflow
- **ADC Low Limit Interrupt Flag**: 1: The last conversion resulted in a low-limit condition for one of the limit registers.
<table>
<thead>
<tr>
<th>ADC Interrupt Control</th>
<th>ADC_INTR</th>
<th>[0x000C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>adc_hi_limit_if</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>ref_ready_if</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>done_if</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:5</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>overflow_ie</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>adc_lo_limit_ie</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>adc_hi_limit_ie</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>ref_ready_ie</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>done_ie</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 11-10: ADC Limit 0 to 3 Registers

<table>
<thead>
<tr>
<th>ADC Limit 0</th>
<th>ADC_LIMIT0</th>
<th>[0x00010]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC Limit 1</td>
<td>ADC_LIMIT1</td>
<td>[0x00014]</td>
</tr>
<tr>
<td>ADC Limit 2</td>
<td>ADC_LIMIT2</td>
<td>[0x00018]</td>
</tr>
<tr>
<td>ADC Limit 3</td>
<td>ADC_LIMIT3</td>
<td>[0x0001C]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td>Reserved for Future Use</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Do not modify this field.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>ch_hi_limit_en</td>
<td>R/W</td>
<td>0</td>
<td>High Limit Monitoring Enable</td>
</tr>
<tr>
<td></td>
<td>If set, then an ADC conversion that results in a value greater than the ch_high_limit field generates an ADC interrupt if the ADC high-limit interrupt is enabled.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(ADC_INTR.hi_limit_ie = 1).</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: The high-limit comparison for the ch_sel channel is active.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: The high-limit comparison is not enabled.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>ch_lo_limit_en</td>
<td>R/W</td>
<td>0</td>
<td>Low Limit Monitoring Enable</td>
</tr>
<tr>
<td></td>
<td>If set, then an ADC conversion that results in a value less than the ch_high_limit field generates an ADC interrupt if the ADC low-limit interrupt is enabled (ADC_INTR.lo_limit_ie = 1).</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: The low-limit comparison for the ch_sel channel is active.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: The low-limit comparison is not enabled.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27:24</td>
<td>ch_sel</td>
<td>R/W</td>
<td>0</td>
<td>ADC Channel for Limit Monitoring</td>
</tr>
<tr>
<td></td>
<td>Sets the ADC input channel for high- and low-limit thresholds. See ADC_CTRL.ch_sel for valid values for this field.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# ADC Limit 0

<table>
<thead>
<tr>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC_LIMIT0</td>
<td>[0x0010]</td>
<td>0</td>
<td>Reserved for Future Use. Do not modify this field.</td>
</tr>
</tbody>
</table>

# ADC Limit 1

<table>
<thead>
<tr>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC_LIMIT1</td>
<td>[0x0014]</td>
<td>0</td>
<td>Reserved for Future Use. Do not modify this field.</td>
</tr>
</tbody>
</table>

# ADC Limit 2

<table>
<thead>
<tr>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC_LIMIT2</td>
<td>[0x0018]</td>
<td>0</td>
<td>Reserved for Future Use. Do not modify this field.</td>
</tr>
</tbody>
</table>

# ADC Limit 3

<table>
<thead>
<tr>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC_LIMIT3</td>
<td>[0x001C]</td>
<td>0</td>
<td>Reserved for Future Use. Do not modify this field.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>23:22</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use. Do not modify this field.</td>
</tr>
</tbody>
</table>

| 21:12 | ch_hi_limit | R/W   | 0x3FF | High Limit Threshold<br>Sets the threshold for high-limit comparisons. This field is a 10-bit value compared against any ADC conversion on the channel set in the ch_sel field. ADC conversions greater than this field are over threshold and can result in interrupt assertion if the ch_hi_limit_en field is set. Valid values for this field are 0x000 to 0x3FF. |

| 11:10 | -       | RO     | 0     | Reserved for Future Use<br>Do not modify this field. |

| 9:0  | ch_lo_limit | R/W   | 0     | Low Limit Threshold<br>Sets the threshold for low-limit comparisons. This field is a 10-bit value compared against any ADC conversion on the channel set in the ch_sel field. ADC conversions less than this field are under threshold and can result in interrupt assertion if the ch_lo_limit_en field is set. Valid values for this field are 0x000 to 0x3FF. |
12. UART

The industry-standard UART ports communicate with external devices using standard serial communications protocols. The UARTs are full-duplex Universal Asynchronous Receiver/Transmitter (UART) serial ports. Each UART instance supports identical functionality and registers unless specifically noted otherwise.

The following features are provided:

- Flexible baud rate generation up to 4 Mbps with ±2% accuracy
- Programmable character size of 5-bits to 8-bits
- Stop bit settings of 1, 1.5, or 2-bits
- Parity settings of even, odd, mark (always 1), space (always 0), and no parity
- Automatic parity error detection with selectable parity bias
- Automatic framing error detection
- Separate 32-bytes deep transmit and receive FIFOs
- Flexible interrupt conditions
- Hardware flow control for RTS and CTS
- Null modem support
- Break generation and detection
- Wakeup from DEEPSLEEP on UART edge with no character loss
- Receive timeout detection
- DMA capable

12.1 Instances

Three instances of the UART are provided: UART0, UART1, and UART2.

Each, supports identical functionality and registers unless specifically noted otherwise. For simplicity, the UARTs are referenced in the documentation as UARTn where n = 0, 1, or 2.

12.2 UART Frame

*Figure 12-1: UART Frame Diagram*

Character sizes of 5 to 8 bits are supported. The field UARTn_CTRL0.size is used to select the character size.

Stop bit support includes 1, 1.5, and 2 stop bits selected with the register field UARTn_CTRL0.stop. Character sizes of 5 can have 1.5 stop bits. All other character sizes can have 1 or 2 stop bits.

Parity support includes even, odd, mark, space or none. For no parity, set field UARTn_CTRL0.parity_en to 0. For all other parity options, select one of the four parity options using the UARTn_CTRL0.parity_mode field and enable parity.
(UARTn_CTRL0.parity_en=1). Parity can be based on the number of logic high bits or logic low bits in the receive characters as set in the register bit UARTn_CTRL0.parity_lvl.

Break frames are transmitted by setting the field UARTn_CTRL0.break to 1. A break sets all bits in the frame to 0.

When a break frame is received, two interrupts are available, UARTn_INT_FL.break is set to 1 when the break frame character is detected, and UARTn_INT_FL.break_end is set when the end of the break character is detected.

Note: A break condition does not set the frame error flag because breaks are not valid UART characters.

### 12.3 UART Interrupts

Interrupts can be generated for the for the conditions in the following table:

<table>
<thead>
<tr>
<th>Interrupt</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX FIFO Level</td>
<td>The transmit FIFO level transitions from being greater than to being equal to the set transmit threshold.</td>
</tr>
<tr>
<td>RX FIFO Level</td>
<td>The receive FIFO level less than the set receive threshold.</td>
</tr>
<tr>
<td>RX FIFO Overrun</td>
<td>The receive FIFO is full but is still receiving data.</td>
</tr>
<tr>
<td>CTS State Change</td>
<td>CTS is deasserted, which tells the UART to pause transmitting data.</td>
</tr>
<tr>
<td></td>
<td>CTS is asserted, which tells the UART to resume transmitting data.</td>
</tr>
<tr>
<td>RX Parity error</td>
<td></td>
</tr>
<tr>
<td>RX Frame</td>
<td>START or STOP bits were not detected.</td>
</tr>
<tr>
<td>RX Timeout</td>
<td>no characters were received within the set timeout period.</td>
</tr>
<tr>
<td>Break</td>
<td>Beginning and end of break.</td>
</tr>
</tbody>
</table>

### 12.4 UART Baud Rate Clock Source

The device can use either the peripheral clock or a fixed 7.3728MHz clock as its source for baud rate generation. The fixed 7.3728MHz clock should be used for the UART bit rate clock generator if the selected system clock (fSYS_CLK) does not meet the bit rate requirements of the application. In practice, the 7.3728MHz clock is ideal for use during low power mode where the Peripheral Clock is turned off for power conservation. The 7.3728MHz clock can be enabled during low power modes enabling the microcontroller to send and receive data while in low power mode.

The UART bit rate clock is set using the UART_CTRL0.clksel field. The UART defaults to the peripheral clock for the bit rate generator clock source. Setting UART_CTRL0.clksel to 1 selects the 7.3728MHz clock for the bit rate clock source.

The UART always uses the peripheral clock for register access and logic operation.

### 12.5 UART Baud Rate Calculation

The UART peripheral clock, fPCLK, is used as the input clock to the UART bit rate generator. The following fields are used to set the target bit rate for the UART instance.

- UARTn_BAUD0.clkdiv: Selects the bit rate clock divisor.
- UARTn_BAUD0.ibaud: Sets the integer portion of the bit rate divisor.
- UARTn_BAUD1.dbaud: Sets the decimal portion of the bit rate divisor.

The equations below are used to determine the values for each of the bit rate fields required to achieve a target bit rate for the UART instance.
Equation 12-1: UART Bit Rate Divisor Equation

\[
\text{DIV} = \frac{f_{\text{UART \_BIT \_RATE \_CLK}}}{(2^{(7-\text{UARTn \_BAUD0 \_.clkdiv})\times \text{Target \ Baud \ Rate})}}
\]

where,

Target Baud Rate is the desired UART interface speed

\(f_{\text{UART \_BIT \_RATE \_CLK}}\) is the UART interface time base frequency. This frequency is either \(f_{\text{PCLK}}\) or the 7.3728MHz clock.

Note: \(\text{UARTn \_BAUD0 \_.clkdiv}\) should be set to the lowest value that results in \(\lceil \text{DIV} \rceil \geq 1\) to achieve the highest accuracy for the target bit rate. \([x]\) is a function that takes as input a real number \(x\) and gives as output the greatest integer less than or equal to \(x\).

Equation 12-2: Bit Rate Integer Calculation

\(\text{UARTn \_BAUD0 \_.ibaud} = \lceil \text{DIV} \rceil\)

Equation 12-3: Bit Rate Remainder Calculation

\[y = \lceil (\text{DIV} - \text{UARTn \_BAUD0 \_.ibaud}) \times 128 \rceil\]

if \(y > 3\)

\(\text{UARTn \_BAUD1 \_.dbaud} = y - 3\)

else

\(\text{UARTn \_BAUD1 \_.dbaud} = y + 3\)

Example Baud Rate Calculation:

Target Bit Rate = 1,843,200 bits per second (1.8 Mbps)

\(f_{\text{UART \_BIT \_RATE \_CLK}} = f_{\text{PCLK}} = 48 \text{ MHz}\)

\[
\text{DIV} = \frac{48,000,000}{(2^{(7-\text{UARTn \_BAUD0 \_.clkdiv})} \times 1,843,200)}
\]

Table 12-2: Example Baud Rate Calculation Results, Target Bit Rate = 1.8Mbps

<table>
<thead>
<tr>
<th>UARTn_BAUD0.clkdiv</th>
<th>DIV</th>
<th>UARTn_BAUD0.ibaud</th>
<th>UARTn_BAUD1.dbaud</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3.26</td>
<td>3</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>1.63</td>
<td>1</td>
<td>77</td>
</tr>
<tr>
<td>2</td>
<td>0.81</td>
<td>0</td>
<td>(Must be 1 or greater. The value selected for UARTn_BAUD0.clkdiv is not valid)</td>
</tr>
<tr>
<td>1</td>
<td>0.41</td>
<td>0</td>
<td>(Must be 1 or greater. The value selected for UARTn_BAUD0.clkdiv is not valid)</td>
</tr>
<tr>
<td>0</td>
<td>0.20</td>
<td>0</td>
<td>(Must be 1 or greater. The value selected for UARTn_BAUD0.clkdiv is not valid)</td>
</tr>
</tbody>
</table>

Table 12-2, above, shows the resulting DIV for each of the UARTn_BAUD0.clkdiv field settings. With UARTn_BAUD0.clkdiv set to 4 or 3, the resulting DIV value is greater than 1. Setting UARTn_BAUD0.clkdiv to 3 will generate the most accurate
target bit rate because it is the smallest value that results in \( \text{DIV} \geq 1 \). Using 3 for \( \text{UARTn\_BAUD0} . \text{clkdiv} \), \( \text{UARTn\_BAUD0} . \text{ibaud} \) is 1, which is the integer portion of the 1.63 \( \text{DIV} \) calculation. The \( \text{UARTn\_BAUD1} . \text{dbaud} \) field calculation based on \( \text{UARTn\_BAUD0} . \text{clkdiv} = 3 \), \( \text{UARTn\_BAUD0} . \text{ibaud} = 1 \) and \( \text{DIV} = 1.63 \) is:

\[
\text{UARTn\_BAUD1} . \text{dbaud} = \lfloor (1.63 - 1) \times 128 \rfloor - 3
\]

The resulting field settings for the example 1,843,200 bps rate are:

- \( \text{UARTn\_BAUD0} . \text{clkdiv} = 3 \)
- \( \text{UARTn\_BAUD0} . \text{ibaud} = 1 \)
- \( \text{UARTn\_BAUD1} . \text{dbaud} = 77 \)

### 12.6 UART Configuration and Operation

To configure the UART, perform the following steps:

1. Set \( \text{UARTn\_CTRL0} \) and \( \text{UARTn\_CTRL1} \) registers for desired parity, character size, stop bits, flow control, and polarity.
2. Configure the desired baud rate by setting \( \text{UARTn\_BAUD0} . \text{clkdiv} \), \( \text{UARTn\_BAUD0} . \text{ibaud} \), and \( \text{UARTn\_BAUD1} . \text{dbaud} \) and described in UART Baud Rate Calculation.
3. Flush the Rx and Tx FIFO by setting \( \text{UARTn\_CTRL0} . \text{rxflush} \) and \( \text{UARTn\_CTRL0} . \text{txflush} \)
4. Clear any interrupts by writing 1 to \( \text{UARTn\_INT\_FL} .[9:0] \).
5. Enable interrupts as desired by setting individual fields in \( \text{UARTn\_INT\_EN} \).
6. Enable the UART by setting \( \text{UARTn\_CTRL0} . \text{enable} = 1 \).

### 12.7 Wakeup Time

Wakeup is configured by setting the UART I/O pins to GPIO with interrupt capability. Once a ATRAT bit is received, it generates a GPIO interrupt. The firmware interrupt handler must re-configure the GPIO pins to UART functionality to receive the first bit of the receive character. The time for the operation to configure GPIO pins and return for interrupt service to do so is highly dependent upon the value of \( f_{\text{SYS_CLK}} \).

### 12.8 Hardware Flow Control

When hardware flow control is enabled, the CTS (Clear-to-send) and RTS (Request-to-Send) external signals are directly managed by hardware without CPU intervention. RTS and CTS are active when flow control is enabled by setting the register bit \( \text{UARTn\_CTRL0} . \text{flowctl} = 1 \). The polarity of the CTS/RTS signals are configured with register bit \( \text{UARTn\_CTRL0} . \text{flowpol} \) and can be active low or active high.

In operation, the UART that wants to transmit data waits for its CTS input pin to be asserted. If CTS is asserted, then the UART begins transmitting data to the slave UART. If during the transmission the UART notices CTS is deasserted, the UART finishes transmitting the current character and then pauses to wait for CTS to return to an asserted level before transmitting more data.

If this UART is receiving data, and the Receive FIFO reaches the level set in \( \text{UARTn\_CTRL1} . \text{rts\_fifo\_lvl} \), then the RTS signal of this UART is deasserted, informing the transmitting UART to stop sending data to this UART to prevent data overflow. Transmission resumes when the level of the Receive FIFO drops below \( \text{UARTn\_CTRL1} . \text{rts\_fifo\_lvl} \), which automatically asserts RTS.

### 12.9 Registers

See Table 3-1: APB Peripheral Base Address Map for this peripheral/module's base address. If multiple instances are provided, each will have a unique base address. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific reset, if applicable.
### Table 12-3: UART Register Summary

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>UARTn_CTRL0</td>
<td>UARTn Control 0 Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>UARTn_CTRL1</td>
<td>UARTn Control 1 Register</td>
</tr>
<tr>
<td>[0x0008]</td>
<td>UARTn_STAT</td>
<td>UARTn Status Register</td>
</tr>
<tr>
<td>[0x000C]</td>
<td>UARTn_INT_EN</td>
<td>UARTn Interrupt Enable Register</td>
</tr>
<tr>
<td>[0x0010]</td>
<td>UARTn_INT_FL</td>
<td>UARTn Interrupt Flag Register</td>
</tr>
<tr>
<td>[0x0014]</td>
<td>UARTn_BAUD0</td>
<td>UARTn Baud Rate Integer Register</td>
</tr>
<tr>
<td>[0x0018]</td>
<td>UARTn_BAUD1</td>
<td>UARTn Baud Rate Decimal Register</td>
</tr>
<tr>
<td>[0x001C]</td>
<td>UARTn_FIFO</td>
<td>UARTn FIFO Read/Write Register</td>
</tr>
<tr>
<td>[0x0020]</td>
<td>UARTn_DMA</td>
<td>UARTn DMA Configuration Register</td>
</tr>
<tr>
<td>[0x0024]</td>
<td>UARTn_TXFIFO</td>
<td>UARTn Transmit FIFO Register</td>
</tr>
</tbody>
</table>

### 12.10 Register Details

#### Table 12-4: UART Control 0 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>23:16</td>
<td>to_cnt</td>
<td>R/W</td>
<td>0</td>
<td>Receive Timeout Frame Count</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Represents the number of frames to wait for a character. If the Receive FIFO contains data, a Receive Timeout condition occurs if the number of frames in this register passes without the FIFO receiving any new data. If a timeout occurs, the hardware sets the receive timeout flag to 1 (UARTn_INT_FL.rx_to = 1).</td>
</tr>
<tr>
<td>15</td>
<td>clk_sel</td>
<td>R/W</td>
<td>0</td>
<td>Bit Rate Clock Source Select</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: f_{UART_BIT_RATE_CLK} = f_{PCLK}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: f_{UART_BIT_RATE_CLK} = 7.3728MHz</td>
</tr>
<tr>
<td>14</td>
<td>break</td>
<td>R/W</td>
<td>0</td>
<td>Transmit BREAK</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set this field to 1 to set the TX line low during a character transmission. A character must be transmitting for this feature to operate. The Tx line remains low until this field is set to zero.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Normal UART operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Transmit zero during a character transfer.</td>
</tr>
<tr>
<td>13</td>
<td>nullmod</td>
<td>R/W</td>
<td>0</td>
<td>Null Modem Support</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Normal operation for RTS/CTS and TX/RX</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Null Modem Mode: RTS/CTS swapped, TX/RX swapped</td>
</tr>
<tr>
<td>12</td>
<td>flowpol</td>
<td>R/W</td>
<td>0</td>
<td>RTS/CTS Polarity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This field controls the polarity used for the RTS/CTS signals. Setting this field to 0 indicates active low assertion for the signals. Setting this field to 1 uses an active high assertion.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: RTS/CTS asserted is 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: RTS/CTS asserted is 1</td>
</tr>
<tr>
<td>11</td>
<td>flowctl</td>
<td>R/W</td>
<td>0</td>
<td>Hardware Flow Control Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Hardware flow control disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Hardware RTS/CTS flow control enabled.</td>
</tr>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
<td>Reset</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>--------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>10</td>
<td>stop</td>
<td>R/W</td>
<td>0</td>
<td><strong>Stop Bit Mode Select</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Select the number of stop bits per character. The default is 1 stop bit, stop = 0. Setting this field to 1 enables 1.5 stop bits for a 5-bit character size and 2 stop bits for all other character sizes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: 1 stop bit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: 1.5 stop bits for 5-bit character size or 2 STOP bits for all other character sizes</td>
</tr>
<tr>
<td>9:8</td>
<td>size</td>
<td>R/W</td>
<td>0</td>
<td><strong>Character Size</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set the number of bits per character.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0: 5 data bits</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1: 6 data bits</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x2: 7 data bits</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x3: 8 data bits</td>
</tr>
<tr>
<td>7</td>
<td>bitacc</td>
<td>R/W</td>
<td>0</td>
<td><strong>Frame or Bit Accuracy Select</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This field selects between either Frame Accuracy or Bit Accuracy for transmitting data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Frame Accuracy: Individual character bit durations may be varied by hardware to meet the target frame period.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bit accuracy: Character bit width is prioritized by the hardware. The overall frame accuracy may vary.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>For clock rates that cannot achieve the +/- 2% baud rate accuracy, this bit allows the firmware to select whether bit times are varied to approximate the desired baud rate.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Frame accuracy is prioritized.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Bit accuracy is prioritized.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: A frame includes the start, stop, all data bits, and parity bit/bits for the character being transmitted.</td>
</tr>
<tr>
<td>6</td>
<td>rxflush</td>
<td>R/W1O</td>
<td>0</td>
<td><strong>Receive FIFO Flush</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to flush the receive FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Cleared to 0 by hardware when flush is completed</td>
</tr>
<tr>
<td>5</td>
<td>txflush</td>
<td>R/W1O</td>
<td>0</td>
<td><strong>Transmit FIFO Flush</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to flush the Transmit FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Cleared to 0 by hardware when flush is completed</td>
</tr>
<tr>
<td>4</td>
<td>parity_lvl</td>
<td>R/W</td>
<td>1</td>
<td><strong>Parity Level Select</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Parity is based on number of 0 bits in the character.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Parity is based on number of 1 bits in the character.</td>
</tr>
<tr>
<td>3:2</td>
<td>parity_mode</td>
<td>R/W</td>
<td>0</td>
<td><strong>Parity Mode Select</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set this field to the type of parity for the UART data. Mark parity always sets the parity bit to 1 and Space parity always sets the parity bit to 0. For even parity, the parity bit is set to 1 if the number of 1 bits in the character is odd. Odd parity sets the parity bit to 1 if the number of 1 bits in the character is even.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0: Even parity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1: Odd Parity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x2: Mark parity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x3: Space parity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: For even and odd parity, the default uses the number of 1 bits in the character to calculate the parity bit. Set the UARTn_CTRL0.parity_lvl to 0 to base the parity on the number of 0 bits in the character.</td>
</tr>
</tbody>
</table>
### UART Control 0

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>parity_en</td>
<td>R/W</td>
<td>0</td>
<td><strong>Parity Enable</strong>&lt;br&gt; If parity is enabled, parity is generated and verified based on the <code>UARTn_CTRL0.parity_mode</code> field.&lt;br&gt;0: No parity checking or generation.&lt;br&gt;1: Parity generation and checking is enabled.</td>
</tr>
<tr>
<td>0</td>
<td>enable</td>
<td>R/W</td>
<td>0</td>
<td><strong>UART Enable</strong>&lt;br&gt;Enabling the UART activates the bit rate generator. Setting this field to 0 disables the UART, flushes the Transmit FIFO and the Receive FIFO and disables the bit rate generator.&lt;br&gt;0: UART disabled. The Receive FIFO and the Transmit FIFO are flushed, and the bit rate generator is off.&lt;br&gt;1: UART enabled and bit rate generator is enabled.</td>
</tr>
</tbody>
</table>

---

**Table 12-5: UART Control 1 Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:22</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>21:16</td>
<td>rts_fifo_lvl</td>
<td>R/W</td>
<td>0</td>
<td><strong>RTS Receive FIFO Threshold Level</strong>&lt;br&gt;When the Receive FIFO level is equal to or greater than this value, de-assert RTS output signal to inform the transmitting UART to stop sending data. Valid values are 1 to 32.</td>
</tr>
<tr>
<td>15:14</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>13:8</td>
<td>tx_fifo_lvl</td>
<td>R/W</td>
<td>0</td>
<td><strong>Transmit FIFO Threshold Level</strong>&lt;br&gt;When the Transmit FIFO level is less than or equal to this value, set <code>UARTn_INT_FL.tx_fifo_lvl</code> interrupt flag. Valid values are 1 to 32. Set this field greater than 1 to avoid a stall condition when transmitting UART data.&lt;br&gt;Note: See Table 12-1: UART Interrupt Conditions for description.</td>
</tr>
<tr>
<td>7:6</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>5:0</td>
<td>rx_fifo_lvl</td>
<td>R/W</td>
<td>0</td>
<td><strong>Receive FIFO Threshold Level</strong>&lt;br&gt;When the Receive FIFO level is equal to or greater than this value, the hardware sets the <code>UARTn_INT_FL.rx_fifo_lvl</code> interrupt flag is set. Valid values are 1 to 32. Set this field to less than 32 to avoid a Receive FIFO overrun condition.&lt;br&gt;Note: See Table 12-1: UART Interrupt Conditions for description.</td>
</tr>
</tbody>
</table>

---

**Table 12-6: UART Status Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:22</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>21:16</td>
<td>tx_num</td>
<td>RO</td>
<td>0</td>
<td><strong>Number of characters in the Transmit FIFO</strong>&lt;br&gt;Read this field to determine the number of characters in the transmit FIFO.</td>
</tr>
<tr>
<td>15:14</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
</tbody>
</table>
### UART Status

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>13:8</td>
<td>rx_num</td>
<td>RO</td>
<td>0</td>
<td><strong>Number of characters in the Receive FIFO</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Read this field to determine the number of characters in the receive FIFO.</td>
</tr>
<tr>
<td>7</td>
<td>tx_full</td>
<td>RO</td>
<td>0</td>
<td><strong>Transmit FIFO Full Status</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: FIFO is not full.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: FIFO is full.</td>
</tr>
<tr>
<td>6</td>
<td>tx_empty</td>
<td>RO</td>
<td>1</td>
<td><strong>Transmit FIFO Empty Status</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: FIFO is not empty.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: FIFO is empty.</td>
</tr>
<tr>
<td>5</td>
<td>rx_full</td>
<td>RO</td>
<td>0</td>
<td><strong>Receive FIFO Full Status</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: FIFO is not full.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: FIFO is full.</td>
</tr>
<tr>
<td>4</td>
<td>rx_empty</td>
<td>RO</td>
<td>1</td>
<td><strong>Receive FIFO Empty Status</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: FIFO is not empty.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: FIFO is empty.</td>
</tr>
<tr>
<td>3</td>
<td>break</td>
<td>RO</td>
<td>0</td>
<td><strong>Break Status</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set while a break condition exists.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: A break has not been received.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: A break condition exists.</td>
</tr>
<tr>
<td>2</td>
<td>parity</td>
<td>RO</td>
<td>0</td>
<td><strong>Parity Bit State</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This field returns the state of the parity bit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Parity bit is 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Parity bit is 1.</td>
</tr>
<tr>
<td>1</td>
<td>rx_busy</td>
<td>RO</td>
<td>0</td>
<td><strong>Receive Busy</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This field reads 1 when the UART is receiving data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: UART is not actively receiving data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: UART is actively receiving data.</td>
</tr>
<tr>
<td>0</td>
<td>tx_busy</td>
<td>RO</td>
<td>0</td>
<td><strong>Transmit Busy</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This field reads 1 when the UART is transmitting data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: UART is not actively transmitting data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: UART is transmitting data.</td>
</tr>
</tbody>
</table>

### Table 12-7: UART Interrupt Enable Register

<table>
<thead>
<tr>
<th>UART Interrupt Enable</th>
<th>UARTn_INT_EN</th>
<th>[0x000C]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:10</td>
<td>-</td>
<td>RO</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>9</td>
<td>break_end</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Enables the BREAK End Detection interrupt.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>8</td>
<td>rx_to</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>7</td>
<td>break</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Enables the BREAK Detection interrupt.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>UART Interrupt Enable</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| **6** tx_fifo_lvl | Transmit FIFO Threshold Level Interrupt Enable (UARTn_INT_EN[0x000C])
| R/W | 0 | Enables the Transmit FIFO threshold level interrupt. This interrupt occurs when the number of entries in the Transmit FIFO is equal or less than the value set in UARTn_CTRL1.tx_fifo_lvl.
| | | 0: Disabled.
| | | 1: Enabled. |

| **5** tx_fifo#ae | Transmit FIFO Almost Empty Interrupt Enable
| R/W | 0 | This interrupt occurs when there is one byte remaining in the Transmit FIFO.
| | | 0: Disabled.
| | | 1: Enabled. |

| **4** rx_fifo_lvl | Receive FIFO Threshold Level Interrupt Enable
| R/W | 0 | Note: See Table 12-1: UART Interrupt Conditions for description. |

| **3** rx_overrun | Receive FIFO Overrun Interrupt Enable
| R/W | 0 | Note: See Table 12-1: UART Interrupt Conditions for description. |

| **2** cts | CTS State Change Interrupt Enable |
| R/W | 0 | Enable the CTS level change interrupt event. This is often referred to as Modem Status Interrupt. |
| | | 0: Disabled.
| | | 1: Enabled. |

| **1** rx_parity_error | Receive Parity Error Interrupt Enable |
| R/W | 0 | 0: Disabled.
| | | 1: Enabled. |

| **0** rx_frame_error | Receive Frame Error Interrupt Enable |
| R/W | 0 | 0: Disabled.
| | | 1: Enabled. |

### Table 12-8: UART Interrupt Flags Register

<table>
<thead>
<tr>
<th>UART Interrupt Flags</th>
<th>Description</th>
</tr>
</thead>
</table>
| **31:10** - | Reserved for Future Use
| RO | Do not modify this field. |

| **9** break_end | BREAK End Interrupt Flag |
| R/W1C | 0 | When the UART receives a series of BREAK frames, this flag is set when the last BREAK frame is received. Write 1 to clear this field.
| | | 0: Last BREAK condition has not occurred.
| | | 1: Last BREAK condition has occurred.
| | | Note: See Table 12-1: UART Interrupt Conditions for description. |

| **8** rx_to | Receive Frame Timeout Interrupt Flag |
| R/W1C | 0 | This field is set when a receive frame timeout occurs. Write 1 to clear this field.
<p>| | | 0: A receive frame timeout was detected by the UART. |</p>
<table>
<thead>
<tr>
<th>UART Interrupt Flags</th>
<th>UARTn_INT_FL</th>
<th>[0x0010]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bits</strong></td>
<td><strong>Field</strong></td>
<td><strong>Access</strong></td>
</tr>
<tr>
<td>7</td>
<td>break</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>tx_fifo_lvl</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>tx_fifo_ae</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>rx_fifo_lvl</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>rx_ovr</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>cts</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>parity</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>frame</td>
<td>R/W1C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 12-9: UART Rate Integer Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:19</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>18:16</td>
<td>clkdiv</td>
<td>R/W</td>
<td>0</td>
<td>Bit Rate Clock Divisor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This field is used to divide the bit rate clock by the selected Clock Divider value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x0: 128</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x1: 64</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x2: 32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x3: 16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x4: 8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x5: Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x6: Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0x7: Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: See the UART Baud Rate Calculation section for details of determining this field's value for a given UART bit rate.</td>
</tr>
<tr>
<td>15:12</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>11:0</td>
<td>ibaud</td>
<td>R/W</td>
<td>0</td>
<td>Integer Portion of Baud Rate Divisor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This field contains the integer value of the bit rate divisor. See the UART Baud Rate Calculation section to calculate this field's value for a given UART bit rate.</td>
</tr>
</tbody>
</table>

### Table 12-10: UART Baud Rate Decimal Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:12</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>11:0</td>
<td>dbaud</td>
<td>R/W</td>
<td>0</td>
<td>Decimal Portion of Baud Rate Divisor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This field contains the remainder portion of the bit rate divisor. See the UART Baud Rate Calculation section to calculate this field's value for a given UART bit rate.</td>
</tr>
</tbody>
</table>

### Table 12-11: UART FIFO Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>7:0</td>
<td>fifo</td>
<td>R/W</td>
<td>N/A</td>
<td>UART FIFO Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reading this field reads data from the Receive FIFO and writes to this field write to the Transmit FIFO.</td>
</tr>
</tbody>
</table>

### Table 12-12: UART DMA Configuration Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:22</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
</tbody>
</table>
### UART DMA Configuration Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>21:16</td>
<td>rxdma_lvl</td>
<td>R/W</td>
<td>0</td>
<td>Receive FIFO Level DMA Trigger</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If the Receive FIFO level is greater than this value, the DMA channel transfers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>data from the Receive FIFO. DMA transfers continue until the Receive FIFO is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>empty. To avoid an Receive FIFO overrun, do not set this value to 32.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Values above 32 are reserved for future use.</td>
</tr>
<tr>
<td>15:14</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>13:8</td>
<td>txdma_lvl</td>
<td>R/W</td>
<td>0</td>
<td>Transmit FIFO Level DMA Trigger</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If the Transmit FIFO level is less than this value, the DMA channel transfers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>data into the Transmit FIFO. DMA transfers continue until the Transmit FIFO is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>full. To avoid stalling a UART transmission, do not set this value to 1 or 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Values above 32 are Reserved for Future Use.</td>
</tr>
<tr>
<td>7:6</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>5</td>
<td>rxdma_auto_to</td>
<td>R/W</td>
<td>0</td>
<td>Receive DMA Timeout Start</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If UARTn_CTRL0.to_cnt causes an RX timeout and UARTn_STAT.rx_num is greater</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>than zero, the DMA transfer will start. Auto-clear after set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Start not initiated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Start DMA transfer.</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>3</td>
<td>rxdma_start</td>
<td>R/W1</td>
<td>0</td>
<td>Receive DMA Start</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Regardless of the setting of UARTn_DMA.rxdma_lvl, start the DMA transfer if</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UARTn_STAT.rx_num is greater than zero. Auto-clear after set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Start not initiated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Start DMA transfer.</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>1</td>
<td>rxdma_en</td>
<td>R/W</td>
<td>0</td>
<td>Receive FIFO DMA Channel Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>0</td>
<td>txdma_en</td>
<td>R/W</td>
<td>0</td>
<td>Transmit FIFO DMA Channel Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
</tbody>
</table>

### UART Transmit FIFO Data Output Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>7:0</td>
<td>data</td>
<td>RO</td>
<td>0</td>
<td>Transmit FIFO Peek Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reads from this register return the next character available for transmission</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>at the end of the Transmit FIFO. If no data is available, reads of this field</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>return 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reads from this register do not affect the Transmit FIFO state.</td>
</tr>
</tbody>
</table>

---

**Table 12-13: UART Transmit FIFO Data Output Register**

**Table 12-14: UART DMA Configuration Register**

---

MAX32665-MAX32668 User Guide

Maxim Integrated
13. **I²C Master/Slave Serial Communications Peripheral (I2C)**

The I2C peripherals can be configured as either an I²C master or I²C slave at standard data rates. For simplicity, I2Cn is used throughout this section to refer to any of the I2C peripherals.

The MAX32665—MAX32668 has the ability to access the I2Cn peripherals via two different busses (clock sources). Bus 0 is the traditional I²C implementation where the peripheral timing is based on the synchronous APB clock (fSYS_CLOCK divided by 2). Obviously, as the system clock is changed, the timing for the I2Cn peripherals change as well. Managing the timing for I²C can become complex if the system clock is ever changed, especially real-time, for example when trying to maximize battery life. Bus 1, on the other hand, provides access to the I2Cn peripherals via an asynchronous APB bus running at a fixed frequency, 7.3728MHz, and simplifies the management of the I2Cn peripherals being that their timing is not dependent on a clock which may change real-time.

The user selects the desired bus by using either the I2Cn registers designated BUS 0 or designated BUS 1. The registers for each of the bus instances are identical except for the base address. Likewise, the registers for each of the I2Cn peripherals are identical except for the base addresses. The base addresses are illustrated in Figure 12-1: UART Frame Diagram.

For detailed information on I²C bus operation, refer to Maxim Application Note 4024 “SPI/I²C Bus Lines Control Multiple Peripherals” https://www.maximintegrated.com/en/app-notes/index.mvp/id/4024
Figure 13-1: I2C Block Diagram
13.1 I²C Master/Slave Features

Each I²C Master/Slave is compliant with the I²C Bus Specification and include the following features:

- Communicates via a serial data bus (SDA) and a serial clock line (SCL)
- Operates as either a master or slave device as a transmitter or receiver
- Supports I²C Standard Mode, Fast Mode, Fast Mode Plus and High Speed (Hs) mode
- Transfers data at rates up to:
  - 100kbps in Standard Mode
  - 400kbps in Fast Mode
  - 1Mbps in Fast Mode Plus
  - 3.4Mbps in Hs Mode
- Supports multi-master systems, including support for arbitration and clock synchronization for Standard, Fast and Fast Plus modes
- Supports 7- and 10-bit addressing
- Supports RESTART condition
- Supports clock stretching
- Provides transfer status interrupts and flags
- Provides DMA data transfer support
- Supports I²C timing parameters fully controllable via firmware
- Provides glitch filter and Schmitt trigger hysteresis on SDA and SCL
- Provides control, status, and interrupt events for maximum flexibility
- Provides independent 8-byte RX FIFO and 8-byte TX FIFO
- Provides TX FIFO preloading
- Provides programmable interrupt threshold levels for the TX and RX FIFO

13.2 Instances

The three instances of the peripheral are shown in Figure 12-1: UART Frame Diagram. Table 13-1: MAX32665 – MAX32668 I²C Peripheral Pins lists the locations of the SDA and SCL signals for each of the I2Cn peripherals per package.

<table>
<thead>
<tr>
<th>I2Cn</th>
<th>ALTERNATE FUNCTION</th>
<th>ALT FUNCTION #</th>
<th>109 WLP</th>
<th>121 CTBGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2C0</td>
<td>I2C0_SCL</td>
<td>AF1</td>
<td>P0.6</td>
<td>P0.6</td>
</tr>
<tr>
<td></td>
<td>I2C0_SDA</td>
<td>AF1</td>
<td>P0.7</td>
<td>P0.7</td>
</tr>
<tr>
<td>I2C1</td>
<td>I2C1_SCL</td>
<td>AF1</td>
<td>P0.14</td>
<td>P0.14</td>
</tr>
<tr>
<td></td>
<td>I2C1_SDA</td>
<td>AF1</td>
<td>P0.15</td>
<td>P0.15</td>
</tr>
<tr>
<td>I2C2</td>
<td>I2C2_SCL</td>
<td>AF1</td>
<td>P1.14</td>
<td>P1.14</td>
</tr>
<tr>
<td></td>
<td>I2C2_SDA</td>
<td>AF1</td>
<td>P1.15</td>
<td>P1.15</td>
</tr>
</tbody>
</table>

Note: The pins apply to both BUS 0 and BUS 1.
13.3  I^2C Overview

13.3.1  I^2C Bus Terminology

Table 13-2, below, contains terms and definitions used in this chapter for the I^2C Bus Terminology.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmitter</td>
<td>The device that sends data to the bus.</td>
</tr>
<tr>
<td>Receiver</td>
<td>The device that receives data from the bus.</td>
</tr>
<tr>
<td>Master</td>
<td>The device that initiates a transfer, generates clock signals and terminates a transfer.</td>
</tr>
<tr>
<td>Slave</td>
<td>The device addressed by a master.</td>
</tr>
<tr>
<td>Multi-master</td>
<td>More than one master can attempt to control the bus at the same time without corrupting the message.</td>
</tr>
<tr>
<td>Arbitration</td>
<td>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.</td>
</tr>
<tr>
<td>Synchronization</td>
<td>Procedure to synchronize the clock signals of two or more devices.</td>
</tr>
<tr>
<td>Clock Stretching</td>
<td>When a slave device holds SCL low to pause a transfer until it is ready. This feature is optional according to the I^2C 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.</td>
</tr>
</tbody>
</table>

13.3.2  I^2C Transfer Protocol Operation

The I^2C protocol operates over a two-wire bus: a clock circuit (SCL) and a data circuit (SDA). I^2C 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^2C 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^2C transmit and receive data transfer operations occur through the I2Cn_FIFO register. Writes to the register load the TX FIFO and reads of the register return data from the RX FIFO. If a slave sends a NACK in response to a write operation, the I2C master generates an interrupt. The I2C controller can be configured to issue a STOP condition to free the bus.

The receive FIFO contains the received data. If the receive FIFO is full or the transmit FIFO is empty, the I2C master stops the clock to allow time to read bytes from the receive FIFO or load bytes into the transmit FIFO.

13.3.5  Acknowledge and Not Acknowledge

An acknowledge bit (ACK) is generated by the receiver, whether I^2C 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^2C master can then either generate a STOP
condition to abort the transfer, or it can generate a repeated START condition (that is, send a START condition without an intervening STOP condition) to start a new transfer.

A receiver can generate a NACK after a byte transfer if any of the following conditions occur:

- No receiver is present on the bus with the transmitted address. In that case, no device will respond with an acknowledge signal.
- The receiver is unable to receive or transmit because it is busy and is not ready to start communication with the master.
- During the transfer, the receiver receives data or commands it does not understand.
- During the transfer, the receiver is unable to receive any more data.
- If an I²C master has requested data from a slave, it signals the slave to stop transmitting by sending a NACK following the last byte it requires.

### 13.3.6 Bit Transfer Process

Both SDA and SCL circuits are open-drain, bidirectional circuits. Each requires an external pullup resistor that ensures each circuit is high when idle. The I²C specification states that during data transfer, the SDA line can change state only when SCL is low, and that SDA is stable and able to be read when SCL is high as shown in Figure 13-2, below.

**Figure 13-2: I²C Write Data Transfer**

![I²C Write Data Transfer Diagram]

An example of an I²C data transfer is as follows:

1. A bus master indicates a data transfer to a slave with a START condition.
2. The master then transmits one byte with a 7-bit slave address and a single read-write bit: a zero for a write or a one for a read.
3. During the next SCL clock following the read-write bit, the master releases SDA. During this clock period, the addressed slave responds with an ACK by pulling SDA low.
4. The master senses the ACK condition and begins transferring data. If reading from the slave, it floats SDA and allows the slave to drive SDA to send data. After each byte, the master drives SDA low to acknowledge the byte. If writing to the slave, the master drives data on the SDA circuit for each of the eight bits of the byte, and then floats SDA during the ninth bit to allow the slave to reply with the ACK indication.
5. After the last byte is transferred, the master indicates the transfer is complete by generating a STOP condition. A STOP condition is generated when the master pulls SDA from a low to high while SCL is high.
13.4  \(\text{I}^2\text{C}\) Configuration and Usage

13.4.1  \(\text{SCL and SDA Bus Drivers}\)

\(\text{SCL}\) and \(\text{SDA}\) are open-drain signals. In this device, once the \(\text{I}^2\text{C}\) peripheral is enabled and the proper GPIO alternate function is selected, the corresponding pad circuits are automatically configured as open-drain outputs. However, \(\text{SCL}\) 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 \(\text{I}^2\text{C}\) slave device can hold \(\text{SCL}\) low, such as for clock stretching. Push-pull operation is enabled by setting \(\text{i2Cn_CTRL0.sclppm}\) to 1. \(\text{SDA}\), on the other hand, always operates in open-drain mode.

13.4.2  \(\text{SCL Clock Configurations}\)

The \(\text{SCL}\) frequency is dependent upon the values of \(\text{I}^2\text{C}\) peripheral clock and the values of the external pullup resistor and trace capacitance on the \(\text{SCL}\) clock line.

\textit{Note: An external RC load on the SCL line will affect the target SCL frequency calculation.}

13.4.3  \(\text{SCL Clock Generation for Standard, Fast and Fast-Plus Modes}\)

The master generates the \(\text{I}^2\text{C}\) clock on the \(\text{SCL}\) line. When operating as a master, application code must configure the \(\text{i2Cn_CLK_HI}\) and \(\text{i2Cn_CLK_LO}\) registers for the desired \(\text{I}^2\text{C}\) operating frequency.

The \text{MAX32665—MAX32668} has the ability to select the source for the \(\text{I}^2\text{C}\) peripheral clock. Application code can select between the system peripheral clock, \(f_{\text{PCLK}}\) (accessed via \text{i2Cn Bus 0 registers}), or the 7.3728MHz oscillator (accessed via \text{i2Cn Bus 1 registers}). The frequency \(f_{\text{PCLK}}\) is \(f_{\text{SYS_CLK}}\) divided by 2. All three \text{i2Cn} peripherals default to Bus 0. Switching one or more of the peripherals to Bus 1 consists of first selecting the 7.3728MHz bus (GCR_APB_ASYNC.apbasynci2Cn), followed by accessing it via the corresponding \text{i2Cn Bus 1 registers}. The base address are shown in the block diagram shown in Figure 12-1: UART Frame Diagram.

The \(\text{SCL}\) high time is configured in the \(\text{I}^2\text{C}\) Clock High Time register field \(\text{i2Cn_CLK_HI.scl_hi}\) using Equation 13-1. The \(\text{SCL}\) low time is configured in the \(\text{I}^2\text{C}\) Clock Low Time register field \(\text{i2Cn_CLK_LO.scl_lo}\) using Equation 13-2. Each of these fields is 8-bits. The value \(\frac{1}{t_{\text{I2C_CLK}}}\) is either \(f_{\text{PCLK}}\) or 7.3728MHz.

\textit{Equation 13-1: I\textsuperscript{2}C Clock High Time Calculation}

\[ t_{\text{SCL_hi}} = t_{\text{I2C_CLK}} \times (\text{i2Cn_CLK_HI.scl_hi} + 1) \]

\textit{Equation 13-2: I\textsuperscript{2}C Clock Low Time Calculation}

\[ t_{\text{SCL_lo}} = t_{\text{I2C_CLK}} \times (\text{i2Cn_CLK_LO.scl_lo} + 1) \]

\textit{Figure 13-3} shows the association between the \(\text{SCL}\) clock low and high times for Standard, Fast and Fast Plus \(\text{I}^2\text{C}\) frequencies.
During synchronization, external masters or external slaves may be driving SCL simultaneously. This affects the SCL duty cycle. By monitoring SCL, the controller can determine whether an external master or slave is holding SCL low. In either case, the controller waits until SCL is high before starting to count the number of SCL high cycles. Similarly, if an external master pulls SCL low before the controller has finished counting SCL high cycles, then the controller starts counting SCL low cycles and releases SCL once the time period, I2Cn_CLK_LO.scl_lo, has expired.

Because the controller does not start counting the high/low time until the input buffer detects the new value, the actual clock behavior is based on many factors. These include bus loading, other devices on the bus holding SCL low, and the filter delay time of this device.

### 13.4.4 SCL Clock Generation for Hs-mode

To operate the I2C interface in Hs-mode at its maximum speed (~3.4MHz), values to be programmed into the I2Cn_HS_CLK.hs_clk_lo register and I2Cn_HS_CLK.hs_clk_hi register must be determined. Since the Hs-mode operation is entered by first using one of the lower speed modes for pre-ambles, a relevant lower speed mode must also be configured. See **SCL Clock Generation for Standard, Fast and Fast-Plus Modes** for information regarding configuration of lower speed modes.

#### 13.4.4.1 Hs-Mode Timing

With I2C bus capacitances less than 100pf, the following specifications are extracted from the I2C-bus Specification User Manual Rev. 6 April 2014 https://www.nxp.com/docs/en/user-guide/UM10204.pdf

- LOW_MIN, the minimum low time for the I2C bus clock, = 160ns
- HIGH_MIN, the minimum high time for the I2C bus clock, = 60ns
- tR_H_MAX, the maximum rise time of the I2C bus clock, = 40ns
- tF_H_MAX, the maximum fall time of the I2C bus clock, = 40ns

#### 13.4.4.2 Hs-Mode Clock Configuration

The maximum Hs-mode bus clock frequency can now be determined. The system clock frequency, fSYS_CLK, must be known. Hs-mode timing information from **Hs-Mode Timing** must be used.

**Equation 13-3: I2C Target SCL Frequency**

This is the desired target for the maximum I2C clock speed.

\[
\text{Target I2C Clock Frequency, } f_{\text{SCL}} = \frac{1}{f_{\text{SYS_CLK}}}
\]
**Equation 13-4: \( I^2C \) Peripheral Source Clock Period**

\[
I_{I2C,CLK} = \left\{ \begin{array}{l}
\frac{2}{f_{SYS,CLK}}, \quad GCR_{APB,ASYNC}.apbasyncI2Cn = 0 \\
7.3728MHz, \quad GCR_{APB,ASYNC}.apbasyncI2Cn = 1
\end{array} \right.
\]

In HS-mode, the analog glitch filter within the device adds a minimum delay of \( t_{AF,MIN} = 10ns \).

**Equation 13-5: Determining the \( I2Cn_HS_CLK .hs_clk_lo \) Register Value**

\[
I2Cn_HS_CLK .hs_clk_lo = \text{MAX}\left( \left\lfloor \frac{t_{LOW,MIN} + t_{FCL,MAX} + t_{I2C,CLK} - t_{AF,MIN}}{t_{I2C,CLK}} \right\rfloor - 1, \quad \frac{t_{SCL}}{t_{I2C,CLK}} - 1 \right)
\]

**Equation 13-6: Determining the \( I2Cn_HS_CLK .hs_clk_hi \) Register Value**

\[
I2Cn_HS_CLK .hs_clk_hi = \left\lfloor \frac{t_{HIGH,MIN} + t_{rCL,MAX} + t_{I2C,CLK} - t_{AF,MIN}}{t_{I2C,CLK}} \right\rfloor - 1
\]

**Equation 13-7: The Calculated Frequency of the \( I^2C \) Bus Clock Using the Results of Equation 13-5 and Equation 13-6**

Calculated Frequency = \( \left( (I2Cn_HS_CLK .hs_clk_hi + 1) + (I2Cn_HS_CLK .hs_clk_lo + 1) \right) \times t_{I2C,CLK} \)

*Table 13-3: Calculated \( I^2C \) Bus Clock Frequencies* shows the \( I^2C \) bus clock calculated frequencies given different \( f_{SYS,CLK} \) frequencies.

<table>
<thead>
<tr>
<th>( f_{SYS,CLK} ) (MHz)</th>
<th>( I2Cn_HS_CLK .hs_clk_hi )</th>
<th>( I2Cn_HS_CLK .hs_clk_lo )</th>
<th>Calculated Frequency (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>96</td>
<td>4</td>
<td>9</td>
<td>3.2</td>
</tr>
<tr>
<td>48</td>
<td>2</td>
<td>4</td>
<td>3.0</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>2</td>
<td>2.4</td>
</tr>
</tbody>
</table>

### 13.4.5 \( I^2C \) Addressing

After a START condition, an \( I^2C \) slave address byte is transmitted. The \( I^2C \) slave address is composed of a slave address followed by a read/write bit.

*Table 13-4: \( I^2C \) Slave Address Format*

<table>
<thead>
<tr>
<th>Slave Address Bits</th>
<th>R/W Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>000</td>
<td>General Call Address</td>
</tr>
<tr>
<td>0000</td>
<td>001</td>
<td>CBUS Address</td>
</tr>
<tr>
<td>0000</td>
<td>010</td>
<td>Reserved for different bus format</td>
</tr>
<tr>
<td>0000</td>
<td>011</td>
<td>Reserved for future purposes</td>
</tr>
<tr>
<td>0000</td>
<td>1XX</td>
<td>HS-mode master code</td>
</tr>
<tr>
<td>1111</td>
<td>1XX</td>
<td>Reserved for future purposes</td>
</tr>
<tr>
<td>1111</td>
<td>0XX</td>
<td>10-bit slave addressing</td>
</tr>
</tbody>
</table>
In 7-bit addressing mode, the master sends one address byte. To address a 7-bit address slave, first clear $I2Cn\_MSTR\_MODE.sea=0$, then write the address to the TX FIFO formatted as follows where $A_n$ is address $A_6:A_0$.

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 ($I2Cn\_MSTR\_MODE.sea=1$), the first byte the master sends is the 10-bit Slave Addressing byte which includes the first two bits of the 10-bit address, followed by a 0 for the R/W bit. That is followed by a second byte representing the remainder of the 10-bit address. If the operation is a write, this is followed by data bytes to be written to the slave. If the operation is a read, it is followed by a repeated START. Firmware then writes the 10-bit address again with a 1 for the R/W bit. This I²C then starts receiving data from the slave device.

13.4.6 I²C Master Mode Operation

The peripheral operates in master mode when Master Mode Enable $I2Cn\_CTRL0.mst=1$. To initiate a transfer, the master generates a START condition by setting $I2Cn\_MSTR\_MODE.start=1$. 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 $I2Cn\_MSTR\_MODE.restart=1$. If a transaction is in progress, the peripheral finishes the transaction before generating a RESTART. The peripheral then transmits the slave address stored in the TX FIFO. The $I2Cn\_MSTR\_MODE.restart$ bit is automatically cleared to 0 as soon as the master begins a RESTART condition.

$I2Cn\_MSTR\_MODE.start$ 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 $I2Cn\_MSTR\_MODE.stop=1$.

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 $I2Cn\_MSTR\_MODE.stop$ bit is cleared and ignored.

A slave cannot generate START, RESTART, or STOP conditions. Therefore, when Master Mode is disabled, the $I2Cn\_MSTR\_MODE.start$, $I2Cn\_MSTR\_MODE.restart$, and $I2Cn\_MSTR\_MODE.stop$ bits are all cleared to 0.
For master mode operation, the following registers should only be configured when either 1) the I²C peripheral disabled, or 2) the I²C bus is guaranteed to be idle/free. If this peripheral is the only master on the bus, then changing the registers outside of a transaction (I2Cn_MSTR_MODE.start = 0) will satisfy this requirement:

- I2Cn_CTRL0.mst
- I2Cn_CTRL0.irxm
- I2Cn_CTRL0.scl_ppm
- I2Cn_CTRL0.hsmode
- I2Cn_RX_CTRL1.rxcnt
- I2Cn_MSTR_MODE.sea
- I2Cn_MSTR_MODE.mcode
- I2Cn_CLK_LO.scl_lo
- I2Cn_CLK_HI.scl_hi
- I2Cn_HS_CLK.hs_clk_lo
- I2Cn_HS_CLK.hs_clk_hi

In contrast to the above set of registers, these registers below can be safely (re)programmed at any time:

- All interrupt flags and interrupt enables
- I2Cn_TX_CTRL0.txth
- I2Cn_RX_CTRL0.rxth
- I2Cn_TIMEOUT.to
- I2Cn_DMA.txen
- I2Cn_DMA.data
- I2Cn_MSTR_MODE.start
- I2Cn_MSTR_MODE.restart
- I2Cn_MSTR_MODE.stop

**13.4.6.1 I²C Master Mode Receiver Operation**

When in Master Mode, initiating a Master Receiver operation begins with the following sequence:

1. Write the number of data bytes to receive to the I²C Receive Count field (I2Cn_RX_CTRL1.rxcnt).
2. Write the I²C Slave Address Byte to the I2Cn_FIFO register with the R/W bit set to 1
3. Send a START condition by setting I2Cn_MSTR_MODE.start = 1
4. The slave address is transmitted by the controller from the I2Cn_FIFO register.
5. The I²C controller receives an ACK from the slave and the controller sets the address ACK interrupt flag (I2Cn_INT_FLO.adracki = 1).
6. The I²C controller receives data from the slave and automatically ACKs each byte. Firmware must retrieve this data by reading the I2Cn_FIFO register.
7. Once I2Cn_RX_CTRL1.rxcnt data bytes have been received, the I²C controller sends a NACK to the slave and sets the Transfer Done Interrupt Status Flag (I2Cn_INT_FLO.donei = 1).
8. If I2Cn_MSTR_MODE.restart or I2Cn_MSTR_MODE.stop is set, then the I²C controller sends a repeated START or STOP, respectively.
13.4.6.2 \( ^{\text{I}^2\text{C}} \) Master Mode Transmitter Operation

When in Master Mode, initiating a Master Transmitter operation begins with the following sequence:

1. Write the I\(^2\)C Slave Address Byte to the \( ^{\text{I}^2\text{C}}\text{FIFO} \) register with the R/W bit set to 0
2. Write the desired data bytes to the \( ^{\text{I}^2\text{C}}\text{FIFO} \) register, up to the size of the TX FIFO. (e.g. If the TX FIFO size is 8 bytes, firmware may write one address byte and seven data bytes prior to starting the transaction.)
3. Send a START condition by setting \( ^{\text{I}^2\text{C}}\text{MSTR\_MODE}.\text{start} = 1 \)
4. The controller transmits the slave address byte written to the \( ^{\text{I}^2\text{C}}\text{FIFO} \) register.
5. The I\(^2\)C controller receives an ACK from the slave and the controller sets the address ACK interrupt flag (\( ^{\text{I}^2\text{C}}\text{INT\_FLO}.\text{adracki} = 1 \)).
6. The \( ^{\text{I}^2\text{C}}\text{FIFO} \) register data bytes are transmitted on the SDA line.
   a. The I\(^2\)C controller receives an ACK from the slave after each data byte
   b. As the transfer proceeds, firmware should refill the TX FIFO by writing to the \( ^{\text{I}^2\text{C}}\text{FIFO} \) register as needed.
   c. If the TX FIFO goes empty during this process, the controller will pause at the beginning of the byte and wait for firmware to either write more data or instruct the controller to send a RESTART or STOP condition.
7. Once firmware has written all the desired bytes to the \( ^{\text{I}^2\text{C}}\text{FIFO} \) register, firmware should set either \( ^{\text{I}^2\text{C}}\text{MSTR\_MODE}.\text{restart} \) or \( ^{\text{I}^2\text{C}}\text{MSTR\_MODE}.\text{stop} \).
8. Once the controller sends all the remaining bytes and empties the TX FIFO, it will set \( ^{\text{I}^2\text{C}}\text{INT\_FLO}.\text{donei} \) and proceed to send out either a RESTART condition, if \( ^{\text{I}^2\text{C}}\text{MSTR\_MODE}.\text{restart} \) was set, or a STOP condition, if \( ^{\text{I}^2\text{C}}\text{MSTR\_MODE}.\text{stop} \) was set.

13.4.6.3 \( ^{\text{I}^2\text{C}} \) Multi-Master Operation

The I\(^2\)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 and/or address different slaves, only one master can remain in master mode and complete its transaction. The other master must back off transmission and wait until the bus is idle. This process by which the winning master is determined is called bus arbitration.

To determine which master wins the arbitration, for each address or data bit, the master compares the data being transmitted on SDA to the value observed on SDA. If a master attempts to transmit a 1 on SDA (that is, the master lets SDA float) but senses a 0 instead, then that master loses arbitration, and the other master that sent a zero continues with the transaction. The losing master cedes the bus by switching off its SDA and SCL drivers.

*Note that this arbitration scheme works with any number of bus masters: if more than two masters begin transmitting simultaneously, the arbitration continues as each master cedes the bus until only one master remains transmitting. Data is not corrupted because as soon as each master realizes it has lost arbitration, it stops transmitting on SDA, leaving the following data bits sent on SDA intact.*

If the I\(^2\)C master peripheral detects it has lost arbitration, it stops generating SCL; sets \( ^{\text{I}^2\text{C}}\text{INT\_FLO}.\text{areri} \); sets \( ^{\text{I}^2\text{C}}\text{INT\_FLO}.\text{txloi} \), flushing any remaining data in the TX FIFO; and clears \( ^{\text{I}^2\text{C}}\text{MSTR\_MODE}.\text{start} \), \( ^{\text{I}^2\text{C}}\text{MSTR\_MODE}.\text{restart} \), and \( ^{\text{I}^2\text{C}}\text{MSTR\_MODE}.\text{stop} \) to 0. So long as the peripheral is not itself addressed by the winning master, the I\(^2\)C peripheral stays in master mode (\( ^{\text{I}^2\text{C}}\text{CTRL0}.\text{mst} = 1 \)). If at any time another master addresses this peripheral using the address programmed in \( ^{\text{I}^2\text{C}}\text{SLV\_ADDR}.\text{sla} \), then the I\(^2\)C peripheral clears \( ^{\text{I}^2\text{C}}\text{CTRL0}.\text{mst} \) to 0 and begins responding as a slave. This can even occur during the same address transmission during which the peripheral lost arbitration.

*Note: Arbitration loss is considered an error condition, and like the other error conditions will set txloi. Therefore, after an arbitration loss, firmware will need to clear \( ^{\text{I}^2\text{C}}\text{INT\_FLO}.\text{txloi} \) and reload the TX FIFO.*
Also, in a multi-master environment, application firmware does not need to wait for the bus to become free before attempting to start a transaction (writing 1 to $I2Cn_{\text{MSTR\_MODE}}$.start). If the bus is free when $I2Cn_{\text{MSTR\_MODE}}$.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 $t_{\text{BUF}} = t_{\text{SCL\_LO}}$ (see Equation 13-2), and then
3. Send a START condition and begin transmitting the slave address byte(s) in the TX FIFO, followed by the rest of the transfer.

The I$^2$C master peripheral is compliant with all bus arbitration and clock synchronization requirements of the I$^2$C specification; this operation is automatic, and no additional programming is required.

### 13.4.7 I$^2$C Slave Mode Operation

When in slave mode, the I2Cn peripheral operates as a slave device on the I$^2$C bus and responds to an external master’s requests to transmit or receive data. To configure the I2Cn peripheral as a slave, write the $I2Cn_{\text{CTRL0}}$.mst bit to zero. The I2Cn clock is driven by the master on the bus, so the SCL device pin will be driven by the external master and $I2Cn_{\text{STAT}}$.ckmd remains a zero. The desired slave address must be set by writing to the $I2Cn_{\text{SLV\_ADDR}}$.sla register.
For slave mode operation, the following registers should be configured with the I2C peripheral disabled:

- **I2Cn_CTRL0.mst** – 0 for Slave operation.
- **I2Cn_CTRL0.gcen**
- **I2Cn_CTRL0.irxm** – The recommended value for this field is 0. Also, note that a setting of 1 is incompatible with slave mode operation with clock stretching disabled (**I2Cn_CTRL0.scl_strd**=1).
- **I2Cn_CTRL0.scl_strd**
- **I2Cn_CTRL0.hsmode**
- **I2Cn_RX_CTRL0.dnr** – SMBus/PMBus applications should set this to 0, while other applications should set this to 1.
- **I2Cn_TX_CTRL0.macktxafdis**
- **I2Cn_TX_CTRL0.samrtxafdis**
- **I2Cn_TX_CTRL0.samwtxafdis**
- **I2Cn_TX_CTRL0.gcamtxafdis**
- **I2Cn_TX_CTRL0.txpreld** – Recommended value is 0 for applications that can tolerate slave clock stretching (scl_strd = 0), and 1 for applications that do not allow slave clock stretching (scl_strd = 1).
- **I2Cn_CLK_HI.scl_hi** – Applies to Slave Mode when clock stretching is enabled (scl_strd=0) - This will be used to satisfy tSU:DAT after clock stretching; program it so that the value defined by Equation 1-1 is >= tSU:DAT(min)
- **I2Cn_HS_CLK.hs_clk_hi** – Applies to Slave Mode when clock stretching is enabled (scl_strd=0) - This will be used to satisfy tSU:DAT after clock stretching during Hs-Mode operation; program it so that the value defined by Equation 1-1 is >= tSU:DAT(min) for Hs-Mode
- **I2Cn_SLV_ADDR.sla**
- **I2Cn_SLV_ADDR.ea**

In contrast to the above registers, these registers can be safely (re)programmed at any time:

- All Interrupt Flags and Interrupt Enables
- **I2Cn_TX_CTRL0.txth** and **I2Cn_RX_CTRL0.rxth** – TX and RX FIFO Threshold Levels
- **I2Cn_TX_CTRL1.tx_rdy** – Transmit Ready (Can only be cleared by hardware)
- **I2Cn_TIMEOUT.to** – Time Out Control
- **I2Cn_DMA.rxen/I2Cn_DMA.txen** – TX and RX DMA Enables
- **I2Cn_FIFO.data** – FIFO access register

### 13.4.7.1 Slave Transmitter

The device will operate as a slave transmitter when the received address matches the device slave address with the R/W bit set to 1. The master is then reading from the device slave. There are two main modes of slave transmitter operation: just-in-time mode and preload mode.

In just-in-time mode, firmware waits to write the transmit data to the TX FIFO until after the master addresses it for a READ transaction, “just in time” for the data to be sent to the master. This allows firmware to defer the determination of what data should be sent until the time of the address match. As an example, the transmit data could be based off an immediately preceding I²C WRITE transaction that requests a certain block of data to be sent, or the data could represent the latest, most up-to-date value of a sensor reading. Clock stretching must be enabled (**I2Cn_CTRL0.scl_strd** = 0) for just-in-time mode operation.
Program flow for transmit operation in just-in-time mode is as follows:

1. With $I2Cn\_CTRL0.i2cen = 0$, initialize all relevant registers, including specifically for this mode $I2Cn\_CTRL0.scl\_strd = 0$, $I2Cn\_TX\_CTRL0[5:2] = 0x8$ and $I2Cn\_TX\_CTRL0.txpreld=0$. Don’t forget to program $I2Cn\_CLK\_HI.scl\_hi$ and $I2Cn\_HS\_CLK.hs\_clk\_hi$ with appropriate values satisfying tSU;DAT (and HS tSU;DAT).

2. SW sets $I2Cn\_CTRL0.i2cen = 1$.
   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, HW will set $I2Cn\_INT\_FL0.ami$ and $I2Cn\_INT\_FL0.txloi$.

3. SW waits for $I2Cn\_INT\_FL0.ami = 1$, either via polling the interrupt flag or setting $I2Cn\_INT\_EN0.amie$ to interrupt the CPU.

4. After reading $I2Cn\_INT\_FL0.ami = 1$, SW reads $I2Cn\_CTRL0.read$ to determine whether the transaction is a transmit (read=1) or receive (read=0) operation. In this case we assume read=1, indicating transmit.
   a. At this point, HW will hold SCL low until SW clears $I2Cn\_INT\_FL0.txloi$ and loads data into the FIFO.

5. SW clears $I2Cn\_INT\_FL0.ami$ and $I2Cn\_INT\_FL0.txloi$. Now that $I2Cn\_INT\_FL0.txloi$ is 0, SW can begin loading the transmit data into $I2Cn\_FIFO$.

6. As soon as there is data in the FIFO, HW will release SCL (after counting out $I2Cn\_CLK\_HI.scl\_hi$) and send out the data on the bus.

7. While the master keeps requesting data and sending ACKs, $I2Cn\_INT\_FL0.donei$ will remain 0 and SW should continue to monitor the TX FIFO and refill it as needed.
   a. The FIFO level can be monitored synchronously via the TX FIFO status/interrupt flags, or asynchronously by setting $I2Cn\_TX\_CTRL0.txth$ and setting $I2Cn\_INT\_EN0.txthie$ interrupt.
   b. If the TX FIFO ever empties during the transaction, the HW will start clock stretching and wait for it to be refilled.

8. The master ends the transaction by sending a NACK. Once this happens the $I2Cn\_INT\_FL0.donei$ interrupt flag is set, and SW can stop monitoring the TX FIFO.

9. The transaction is complete, SW should “clean up”, including clearing $I2Cn\_INT\_FL0.donei$ and clearing $I2Cn\_INT\_EN0.txthie$ interrupt. We return to step 3, waiting on an address match.

10. If SW needs to know how many data bytes were transmitted to the master, it should check the TX FIFO level as soon as SW sees $I2Cn\_INT\_FL0.donei = 1$ and use that to determine how many data bytes were successfully sent.
    a. Please note that any data remaining in the TX FIFO will be discarded prior to the next transmit operation; it is NOT necessary for SW to manually flush the TX FIFO for this to occur.

The other mode of operation for slave transmit is preload mode. In this mode, it is assumed that the application firmware knows prior to the transmit operation what data it should send to the master. This data is then “preloaded” into the TX FIFO. Once the address match occurs, this data can be sent out without any software intervention. Preload mode can be used with clock stretching either enabled or disabled, but it is the only option if clock stretching must be disabled.
To use slave transmit preload mode:

1. With \textit{I2Cn_CTRL0.i2cen} = 0, initialize all relevant registers, including specifically for this mode \textit{I2Cn_CTRL0.scl_strd} = 1, \textit{I2Cn_TX_CTRL0}[5:2] = 0xF and \textit{I2Cn_TX_CTRL0.txpreld}=1.
2. SW sets \textit{I2Cn_CTRL0.i2cen} = 1.
   a. Even though the controller is enabled, at this point it will not ACK an address match with R/W=1 until SW sets \textit{I2Cn_TX_CTRL1.txrdy}=1.
3. SW prepares for the transmit operation by loading data into the transmit FIFO, enabling DMA, setting \textit{I2Cn_TX_CTRL0.txth} and setting \textit{I2Cn_INT_EN0.txthie} interrupt, etc.
   a. If clock stretching is disabled, then an empty TX FIFO during the transmit operation will cause a TX underrun error. Therefore, firmware should take any necessary steps to avoid a underrun prior to setting \textit{I2Cn_TX_CTRL1.txrdy}=1.
   b. If clock stretching is enabled, then an empty TX FIFO will not cause a TX underrun error. However, it is recommended to follow the same preparation steps in order to minimize the amount of time spent clock stretching, which will let the transaction complete as quickly as possible.
4. Once SW has prepared for the transmit operation, it sets \textit{I2Cn_TX_CTRL1.txrdy}=1.
   a. The controller is now fully enabled and will respond with an ACK to an address match.
   b. HW will set \textit{I2Cn_INT_FL0.ami} once an address match has occurred. \textit{I2Cn_INT_FL0.txloi} will NOT be set and will remain 0.
5. SW waits for \textit{I2Cn_INT_FL0.ami}=1, either via polling the interrupt flag or setting \textit{I2Cn_INT_EN0.amie} to interrupt the CPU.
6. After seeing \textit{I2Cn_INT_FL0.ami}=1, SW reads \textit{I2Cn_CTRL0.read} to determine whether the transaction is a transmit (read=1) or receive (read=0) operation. In this case we assume \textit{I2Cn_CTRL0.read}, indicating transmit.
   a. At this point, HW will begin sending out the data that was preloaded into the TX FIFO.
   b. Once the first data byte is sent, HW will also automatically clear \textit{I2Cn_TX_CTRL1.txrdy} to 0.
7. While the master keeps requesting data and sending ACKs, \textit{I2Cn_INT_FL0.donei} will remain 0 and SW should continue to monitor the TX FIFO and refill it as needed.
   a. The FIFO level can be monitored synchronously via the TX FIFO status/interrupt flags, or asynchronously by setting \textit{I2Cn_TX_CTRL0.txth} and setting \textit{I2Cn_INT_EN0.txthie} interrupt.
   b. If clock stretching is disabled and the TX FIFO ever empties during the transaction, the HW will set \textit{I2Cn_INT_FL1.txufi}=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 \textit{I2Cn_INT_FL0.donei} interrupt flag is set.
   a. If the TX FIFO goes empty at the same time that the master indicates the transaction is complete by sending a NACK, this is not considered an underrun event, and the \textit{I2Cn_INT_FL1.txufi} flag will remain 0.
9. The transaction is complete, SW should "clean up", which should include clearing \textit{I2Cn_INT_FL0.donei}. We return to step 3 and prepare for the next transaction.
   a. If SW needs to know how many data bytes were transmitted to the master, it should check the TX FIFO level as soon as SW sees \textit{I2Cn_INT_FL0.donei} = 1 and use that to determine how many data bytes were successfully sent.
   b. By default, any data remaining in the TX FIFO will NOT be discarded, and instead will be reused for the next transmit operation.
   c. If this is not desired, SW can flush the TX FIFO. The safest way to do this is by clearing and then re-setting i2cen. This will flush both the TX and RX FIFOs.

Once a slave starts transmitting out of its \textit{I2Cn_FIFO}, detection of out of sequence STOP, START or RESTART condition will terminate the current transaction. When a transaction is terminated in such manner, \textit{I2Cn_INT_FL0.strteri} or \textit{I2Cn_INT_FL0.stoperi} will be set to 1.
If the TX FIFO is not ready (I2Cn_TX_CTRL1.txrdy = 0) and the I2C controller receives a data read request from the master, the hardware automatically sends a NACK at the end of the first address byte. The setting of the Do Not Respond field is ignored by the hardware in this case because the only opportunity to send a NACK for an I2C read transaction is after the address byte.

13.4.7.2 Slave Receiver

The device will operate as a slave receiver when the received address matches the device slave address with the R/W bit set to 0. The external master is writing to the slave.

Program flow for a receive operation is as follows:

1. With I2Cn_CTRL0.i2cen = 0, initialize all relevant registers.
2. Set I2Cn_CTRL0.i2cen = 1.
   a. If an address match with R/W=0 occurs, and the RX FIFO is empty, the peripheral will respond with an ACK and the I2Cn_INT_FL0.ami flag will be set.
   b. If the RX FIFO is not empty, then depending on the value of I2Cn_RX_CTRL0.dnr, the peripheral will NACK either the address byte (I2Cn_RX_CTRL0.dnr = 1) or the first data byte (I2Cn_RX_CTRL0.dnr = 0).
3. Wait for I2Cn_INT_FL0.ami = 1, either by polling or by enabling the I2Cn_INT_FL0.wrami interrupt to the CPU. Once a successful address match occurs, hardware will set I2Cn_INT_FL0.ami = 1.
4. Read I2Cn_CTRL0.read to determine whether the transaction is a transmit (I2Cn_CTRL0.read = 1) or receive (I2Cn_CTRL0.read = 0) operation. In this case we assume I2Cn_CTRL0.read = 0, indicating receive. At this point, the device will begin receiving data into the RX FIFO.
5. Clear I2Cn_INT_FL0.amic, and while the master keeps sending data, I2Cn_INT_FL0.donei will remain 0 and software should continue to monitor the RX FIFO and empty it as needed.
   a. The FIFO level can be monitored synchronously via the RX FIFO status/interrupt flags, or asynchronously by setting I2Cn_RX_CTRL0.rxth and enabling the I2Cn_INT_FL0.rxthi interrupt.
   b. If the RX FIFO ever fills up during the transaction, then hardware will set I2Cn_INT_FL1.rxofi and then either:
      i. if I2Cn_CTRL0.scl_strd = 0, start clock stretching and wait for software to read from the RX FIFO, or,
      ii. if I2Cn_CTRL0.scl_strd = 1, respond to the master with a NACK and the last byte is discarded.
6. The master ends the transaction by sending a RESTART or STOP. Once this happens the I2Cn_INT_FL0.donei interrupt flag is set, and software can stop monitoring the RX FIFO.
7. Once a slave starts receiving into its RX_FIFO, detection of out of sequence STOP, START or RESTART condition will release the I2C bus to the Idle state and hardware will set I2Cn_INT_FL0.strteri or I2Cn_INT_FL0.stoperi to 1.

If software has not emptied the data in the RX FIFO from the previous transaction by the time that a master addresses it for another write (i.e. receive) transaction, then the controller will not participate in the transaction, and no additional data will be written into the FIFO. Although a NACK will be sent to the master, software can control whether the NACK is sent with the initial address match, or if instead it is sent at the end of the first data byte. Setting I2Cn_RX_CTRL0.dnr to 1 chooses the former, while setting I2Cn_RX_CTRL0.dnr to 0 chooses the latter.
13.4.8 I²C Interrupt Sources

The I²C controller has a very flexible interrupt generator that generates an interrupt signal to the Interrupt Controller on any of several events. On recognizing the I²C interrupt, firmware determines the cause of the interrupt by reading the I²C Interrupt Flags registers I2Cn_INT_FL0 and I2Cn_INT_FL1. Interrupts can be generated for the following events:

- Transaction Complete (Master/Slave)
- Address NACK received from slave (Master)
- Data NACK received from slave (Master)
- Lost arbitration (Master)
- Transaction timeout (Master/Slave)
- FIFO is empty, not empty, full to configurable threshold level (Master/Slave)
- TX FIFO locked out because it is being flushed (Master/Slave)
- Out of sequence START and STOP conditions (Master/Slave)
- Sent a NACK to an external master because the TX or RX FIFO was not ready (Slave)
- Address ACK or NACK received (Master)
- Incoming address match (Slave)
- TX Underflow or RX Overflow (Slave)

Interrupts for each event can be enabled or disabled by setting or clearing the corresponding bit in the I2Cn_INT_EN0 or I2Cn_INT_EN1 interrupt enable register.

Note: Disabling the interrupt does not prevent the corresponding flag from being set by hardware but does prevent an IRQ when the interrupt flag is set.

Note: Prior to enabling an interrupt, the status of the corresponding interrupt flag should be checked and, if necessary, serviced or cleared. This prevents a previous interrupt event from interfering with a new I²C communications session.

13.4.9 TX FIFO and RX FIFO

There are separate transmit and receive FIFOs, TX FIFO and RX FIFO. Both are accessed using the FIFO Data register I2Cn_FIFO. Writes to this register enqueue data into the TX FIFO. Writes to a full TX FIFO have no effect. Reads from I2Cn_FIFO dequeue data from the RX FIFO. Writes to a full TS FIFO have no effect, and reads from an empty RX FIFO return 0xFF.

The TX and RX FIFO will only read or write one byte at a time. Transactions larger than 8 bits can still be performed, however. A 16- or 32-bit write to the TX FIFO stores just the lowest 8 bits of the write data. A 16- or 32-bit read from the RX FIFO will have the valid data in the lowest 8 bits and 0's in the upper bits. In any case, the TX and RX FIFOs will only accept 8 bits at a time for either reads or a writes.

To offload work from the CPU, the DMA can read and write to each FIFO. See section I²C DMA Control for more information on configuring the DMA.
During a receive transaction (which during master operation is a READ, and during slave operation is a WRITE), received bytes are automatically written to the RX FIFO. Software should monitor the RX FIFO level and unload data from it as needed by reading \texttt{I2Cn\_FIFO}. If the receive FIFO becomes full during a master mode transaction, then the controller sets the \texttt{I2Cn\_INT\_FL1.rxofi} bit and one of two things will happen depending on the value of \texttt{I2Cn\_CTRL0.scl\_strd}:

- If clock stretching is enabled (\texttt{I2Cn\_CTRL0.scl\_strd} = 0), then the controller stretches the clock until software makes space available in the RX FIFO by reading from \texttt{I2Cn\_FIFO}. Once space is available, the peripheral moves the data byte from the shift register into the RX FIFO, the SCL device pin is released, and the master is free to continue the transaction.
- If clock stretching is disabled (\texttt{I2Cn\_CTRL0.scl\_strd} = 1), then the controller responds to the master with a NACK and the data byte is lost. The master can return the bus to idle with a STOP condition or start a new transaction with a RESTART condition.

During a transmit transaction (which during master operation is a WRITE, and during slave operation is a READ), either user software or the DMA can provide data to be transmitted by writing to the TX FIFO. Once the peripheral finishes transmitting each byte, it removes it from the TX FIFO and, if available, begins transmitting the next byte.

Interrupts can be generated for the following FIFO status:

- TX FIFO level less than or equal to threshold
- RX FIFO level greater than or equal to threshold
- TX FIFO underflow
- RX FIFO overflow
- TX FIFO locked for writing

Both the RX FIFO and TX FIFO are flushed when the I2C port is disabled by clearing \texttt{I2Cn\_CTRL0.i2cen}=0. While the peripheral is disabled, writes to the TX FIFO have no effect and reads from the RX FIFO return 0xFF.

The TX FIFO and RX FIFO can be flushed by setting the Transmit FIFO Flush bit (\texttt{I2Cn\_TX\_CTRL0.txfsh=1}) or the Receive FIFO Flush bit (\texttt{I2Cn\_RX\_CTRL0.rxfsh=1}), respectively. In addition, under certain conditions the TX FIFO is automatically locked by hardware and flushed so stale data is not unintentionally transmitted. The TX FIFO is automatically flushed, and writes locked out from software under the following conditions:

- General Call Address Match. Automatic flushing and lockout can be disabled by setting \texttt{I2Cn\_TX\_CTRL0.gcamtxafdis}.
- Slave Address Match Write. Automatic flushing and lockout can be disabled by setting \texttt{I2Cn\_TX\_CTRL0.samwtxafdis}.
- Slave Address Match Read. Automatic flushing and lockout can be disabled by setting \texttt{I2Cn\_TX\_CTRL0.samrtxfdis}.
- During operation as a slave transmitter, a NACK is received. Automatic flushing and lockout can be disabled by setting \texttt{I2Cn\_TX\_CTRL0.rnacktxafdis}.
- Any of the following interrupts: Arbitration Error, Timeout Error, Master Mode Address NACK Error, Master Mode Data NACK Error, Start Error, and STOP Error. Automatic flushing cannot be disabled for these conditions.

When the above conditions occur, the TX FIFO is flushed so that data intended for a previous transaction is not unintentionally transmitted for a new transaction. In addition to flushing the the TX FIFO, the Transmit Lockout Flag is set (\texttt{I2Cn\_INT\_FL0.txloi=1}) and writes to the TX FIFO are ignored until firmware acknowledges the external event by clearing \texttt{I2Cn\_INT\_FL0.txloi}.

### 13.4.10 TX FIFO Preloading

There may be situations during slave mode operation where software wants to preload the TX FIFO prior to a transmission, such as when clock stretching is disabled. In this scenario, rather than responding to an external master requesting data with an ACK and clock stretching while software writes the data to the TX FIFO, the controller will instead respond with a NACK until software has preloaded the requested data into the TX FIFO.
When TX FIFO Preloading is enabled, the software controls ACKs to the external master using the TX Ready (\texttt{I2Cn_TX_CTRL1.txrdy}) bit. When \texttt{I2Cn_TX_CTRL1.txrdy} is set to 0, hardware automatically NACKs all read transactions from the Master. Setting \texttt{I2Cn_TX_CTRL1.txrdy} to 1 sends an ACK to the Master on the next read transaction and transmits the data in the TX FIFO. Preloading the TX FIFO should be complete prior to setting the \texttt{I2Cn_TX_CTRL1.txrdy} field to 1.

The required steps for implementing TX FIFO Preloading in an application are as follow:

1. Enable TX FIFO Preloading by setting \texttt{I2Cn_TXCTRL1.txpreld} to 1.
   a. This will automatically clear \texttt{I2Cn_TX_CTRL1.txrdy} to 0
2. If the TX FIFO Lockout Flag (\texttt{I2Cn_INT_FL0.txloi}) is set to 1, write 1 to clear the flag and enable writes to the TX FIFO.
3. Enable DMA or Interrupts if required.
4. Load the TX FIFO with the data to send when the Master sends the next read request.
5. Set \texttt{I2Cn_TX_CTRL1.txrdy} to 1 to automatically let the hardware send the preloaded FIFO on the next read from a Master.
6. \texttt{I2Cn_TX_CTRL1.txrdy} is cleared by hardware once it finishes transmitting the first byte, and data is transmitted from the TX FIFO. Once cleared, the application firmware may repeat the Preloading process or disable TX FIFO Preloading.

Note: To prevent the preloaded data from being cleared when the master tries to read it, firmware must at least set \texttt{I2Cn_TX_CTRL0.samwtxfadis} to 1, disabling autoflush on READ address match. The software will determine whether the other autoflush disable bits should be set. For example, if a master uses I\(^2\)C WRITE transactions to determine what data the slave should send in the following READ transactions, then software can clear \texttt{I2Cn_TX_CTRL0.samwtxfadis} to 0. Then when a WRITE occurs, the TX FIFO is flushed, giving firmware time to load the new data. For the READ transaction, the external master can poll the slave address until the new data has been loaded and \texttt{I2Cn_TX_CTRL1.txrdy} is set, at which point the peripheral responds with an ACK.

### 13.4.11 Interactive Receive Mode (IRXM)

In some situations, the I\(^2\)C might want to inspect and respond to each byte of received data. In this case, Interactive Receive Mode (IRXM) can be used. Interactive Receive Mode is enabled by setting \texttt{I2Cn_CTRL0.irxm}=1. If Interactive Receive Mode is enabled, it must occur before any I\(^2\)C transfer is initiated.

When Interactive Receive Mode (IRXM) is enabled, after every data byte received the I\(^2\)C peripheral automatically holds SCL low before the ACK bit. Additionally, after the 8th SCL falling edge, the I\(^2\)C peripheral sets the IRXM Interrupt Status Flag (\texttt{I2Cn_INT_FLO.irxmi} = 1). Application firmware must read the data and generate a response (ACK or NACK) by setting the IRXM Acknowledge (\texttt{I2Cn_CTRL0.ack}) bit accordingly. Send an ACK by clearing the \texttt{I2Cn_CTRL0.ack} bit to 0. Send a NACK by setting the \texttt{I2Cn_CTRL0.ack} bit to 1.

After setting the \texttt{I2Cn_CTRL0.ack} bit, clear the IRXM interrupt flag. Write 1 to \texttt{I2Cn_INT_FLO.irxmi} to clear the interrupt flag. When the IRXM interrupt flag is cleared, the I\(^2\)C peripheral hardware releases the SCL line and sends the \texttt{I2Cn_CTRL0.ack} on the SDA line.

While the I\(^2\)C peripheral is waiting for the application firmware to clear the \texttt{I2Cn_INT_FLO.irxmi} flag, firmware can disable Interactive Receive Mode and, if operating as a master, load the remaining number of bytes to be received for the transaction. This allows firmware to examine the initial bytes of a transaction, which might be a command, and then disable Interactive Receive Mode to receive the remaining bytes in normal operation.

During IRXM, received data is not placed in the RX FIFO. Instead, the \texttt{I2Cn_FIFO} address is repurposed to directly read the receive shift register, bypassing the RX FIFO. Therefore, before disabling Interactive Receive Mode, firmware must first read the data byte from \texttt{I2Cn_FIFO.data}. If the IRXM byte is not read, the byte is lost and the next read from the RX FIFO returns 0xFF.

Note: Interactive Receive Mode does not apply to address bytes, only to data bytes.
Note: Interactive Receive Mode does not apply to general call address responses or START byte responses.

Note: When enabling Interactive Receive Mode and operating as a slave, clock stretching must remain enabled ($I2Cn_{-}CTRL0.scl\_strd = 0$).

13.4.12 Clock Stretching

When the I2C peripheral requires some response or intervention from the application firmware in order to continue with a transaction, it will hold SCL low, preventing the transfer from continuing. This is called ‘clock stretching’ or ‘stretching the clock’. While the I2C Bus Specification defines the term ‘clock stretching’ to only apply to a slave device holding the SCL line low, this section describes situations where the I2C peripheral holds the SCL line low in either slave or master mode and refers to both as clock stretching.

When the I2C peripheral stretches the clock, it typically does so in response to either a full RX FIFO during a receive operation, or an empty TX FIFO during a transmit operation. Necessarily, this occurs before the next data byte begins, either between the ACK bit and the first data bit or, if at the beginning of a transaction, immediately after a START or RESTART condition. However, when operating in Interactive Receive Mode ($I2Cn_{-}CTRL0.irxm = 1$), the peripheral can also clock stretch before the ACK bit, allowing firmware to decide whether to send an ACK or NACK.

For a transmit operation (as either master or slave), when the TX FIFO is empty, SCL is automatically held low after the ACK bit and before the next data byte begins. To stop clock stretching and continue the transaction, firmware must write data to $I2Cn_{-}FIFO.data$. If operating in master mode, however, instead of sending more data, firmware may also set either $I2Cn_{-}MSTR\_MODE.stop$ or $I2Cn_{-}MSTR\_MODE.restart$ to send a STOP or RESTART condition, respectively.

For a receive operation (as either master or slave), when both the RX FIFO and the receive shift register are full, SCL is automatically held low until at least one data byte is read from the RX FIFO. To stop clock stretching and continue the transaction, firmware must read data from $I2Cn_{-}FIFO.data$. If operating in master mode and this is the final byte of the transaction, as determined by $I2Cn_{-}RX\_CTRL1.rxcnt$, then application firmware must also set either $I2Cn_{-}MSTR\_MODE.stop$ or $I2Cn_{-}MSTR\_MODE.restart$ to send a STOP or RESTART condition, respectively. This must be done in addition to reading from the RX FIFO, since the peripheral cannot start sending the STOP or RESTART until the last data byte has been moved from the RX shift register into the RX FIFO. (This will occur automatically once there is space in the RX FIFO.)

Note: Since some masters do not support other devices stretching the clock, it is possible to completely disable all clock stretching during slave mode by setting $I2Cn_{-}CTRL0.scl\_strd$ to 1 and clearing $I2Cn_{-}CTRL0.irxm$ to 0. In this case, instead of clock stretching the I2C peripheral sends a NACK if receiving data, or sends 0xFF if transmitting data.

Note: The clock synchronization required to support other I2C master or slave devices stretching the clock is built into the peripheral and requires no intervention from software to operate correctly.

13.4.13 I2C Bus Timeout

The Timeout register, $I2Cn_{-}TIMEOUT.to$, is used to detect bus errors. Equation 13-8 and Equation 13-9 show equations for calculating the maximum and minimum timeout values based on the value loaded into the $I2Cn_{-}TIMEOUT.to$ field.

Equation 13-8: I2C Timeout Maximum

\[
t_{\text{TIMEOUT}} \leq \left( \frac{1}{f_{I2C\_CLK}} \right) \times (I2Cn_{-}TIMEOUT.to \times 32 + 3)
\]

Due to clock synchronization, the timeout is guaranteed to meet the following minimum time calculation shown in Equation 13-9.

Equation 13-9: I2C Timeout Minimum

\[
t_{\text{TIMEOUT}} \leq \left( \frac{1}{f_{I2C\_CLK}} \right) \times (I2Cn_{-}TIMEOUT.to \times 32 + 2)
\]
The timeout feature is disabled when $I2Cn\_TIMEOUT$ = 0 and is enabled for any non-zero value. When the timeout is enabled, the timeout timer starts counting when the I2C peripheral hardware drives SCL low and is reset by the I2C peripheral hardware when the SCL line is released.

The timeout counter only monitors if the I2C peripheral hardware is driving the SCL line low. It does not monitor if an external I2C 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 I2C peripheral hardware releases the SCL and SDA lines and sets the timeout error interrupt flag to 1 ($I2Cn\_INT\_FL0\_toeri = 1$).

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 ($I2Cn\_TIMEOUT = 0$).

### 13.4.14 I2C DMA Control

There are independent DMA channels for each TX FIFO and each RX FIFO. DMA activity is triggered by the TX FIFO ($I2Cn\_TX\_CTRL0\_txth$) and RX FIFO ($I2Cn\_RX\_CTRL0\_rxth$) threshold levels.

When the TX FIFO byte count ($I2Cn\_TX\_CTRL1\_txfifo$) is less than or equal to the TX FIFO Threshold Level $I2Cn\_TX\_CTRL0\_txth$, then the DMA transfers data into the TX FIFO according to the DMA configuration. To ensure the DMA does not overflow the TX FIFO, the DMA burst size should be set as follows:

**Equation 13-10: DMA Burst Size Calculation for I2C Transmit**

$$\text{DMA Burst Size} \leq \text{TX FIFO Depth} - I2Cn\_TXCTRL0\_txth = 8 - I2Cn\_TXCTRL0\_txth$$

where $0 \leq I2Cn\_TXCTRL0\_txth \leq 7$

Applications trying to avoid transmit underflow and/or clock stretching should use a smaller burst size and higher $I2Cn\_TX\_CTRL0\_txth$ setting. This fills up the FIFO more frequently but increases internal bus traffic.

When the RX FIFO count ($I2Cn\_RX\_CTRL1\_rfifo$) is greater than or equal to the RX FIFO Threshold Level $I2Cn\_RX\_CTRL0\_rxth$, the DMA transfers data out of the RX FIFO according to the DMA configuration. To ensure the DMA does not underflow the RX FIFO, the DMA burst size should be set as follows:

**Equation 13-11: DMA Burst Size Calculation for I2C Receive**

$$\text{DMA Burst Size} \leq I2Cn\_RX\_CTRL0\_rxth$$

where $1 \leq I2Cn\_RX\_CTRL0\_rxth \leq 8$

Applications trying to avoid receive overflow and/or clock stretching should use a smaller burst size and lower $I2Cn\_RX\_CTRL0\_rxth$. This results in reading from the Receive FIFO more frequently but increases internal bus traffic.

Note for receive operations, the length of the DMA transaction (in bytes) must be an integer multiple of $I2Cn\_RX\_CTRL0\_rxth$. Otherwise, the receive transaction will end with some data still in the RX FIFO, but not enough to trigger an interrupt to the DMA, leaving the DMA transaction incomplete. One easy way to ensure this for all transaction lengths is to set burst size to 1 ($I2Cn\_RX\_CTRL0\_rxth = 1$).

To enable DMA transfers, enable the TX DMA channel ($I2Cn\_DMA\_txen$) and/or the RX DMA channel ($I2Cn\_DMA\_rxen$).

### 13.5 Registers

See Table 3-1: APB Peripheral Base Address Map for the I2C0, I2C1, and I2C2 Register Peripheral Base Addresses for BUS 0 and BUS 1.

**Table 13-5: I2C Registers**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>$I2Cn_CTRL0$</td>
<td>I2C Control 0 Register</td>
</tr>
<tr>
<td>Offset</td>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>I2Cn_STAT</td>
<td>I2C Status Register</td>
</tr>
<tr>
<td>[0x0008]</td>
<td>I2Cn_INT_FL0</td>
<td>I2C Interrupt Flags 0 Register</td>
</tr>
<tr>
<td>[0x000C]</td>
<td>I2Cn_INT_EN0</td>
<td>I2C Interrupt Enable 0 Register</td>
</tr>
<tr>
<td>[0x0010]</td>
<td>I2Cn_INT_FL1</td>
<td>I2C Interrupt Flags 1 Register</td>
</tr>
<tr>
<td>[0x0014]</td>
<td>I2Cn_INT_EN1</td>
<td>I2C Interrupt Enable 1 Register</td>
</tr>
<tr>
<td>[0x0018]</td>
<td>I2Cn_FIFO_LEN</td>
<td>I2C FIFO Length Register</td>
</tr>
<tr>
<td>[0x001C]</td>
<td>I2Cn_RX_CTRL0</td>
<td>I2C Receive Control 0 Register</td>
</tr>
<tr>
<td>[0x0020]</td>
<td>I2Cn_RX_CTRL1</td>
<td>I2C Receive Control 1 Register</td>
</tr>
<tr>
<td>[0x0024]</td>
<td>I2Cn_TX_CTRL0</td>
<td>I2C Transmit Control 0 Register</td>
</tr>
<tr>
<td>[0x0028]</td>
<td>I2Cn_TX_CTRL1</td>
<td>I2C Transmit Control 1 Register</td>
</tr>
<tr>
<td>[0x002C]</td>
<td>I2Cn_FIFO</td>
<td>I2C Transmit and Receive FIFO Register</td>
</tr>
<tr>
<td>[0x0030]</td>
<td>I2Cn_MSTR_MODE</td>
<td>I2C Master Mode Register</td>
</tr>
<tr>
<td>[0x0034]</td>
<td>I2Cn_CLK_LO</td>
<td>I2C Clock Low Time Register</td>
</tr>
<tr>
<td>[0x0038]</td>
<td>I2Cn_CLK_HI</td>
<td>I2C Clock High Time Register</td>
</tr>
<tr>
<td>[0x003C]</td>
<td>I2Cn_HS_CLK</td>
<td>I2C Hs-Mode Clock Control Register</td>
</tr>
<tr>
<td>[0x0040]</td>
<td>I2Cn_TIMEOUT</td>
<td>I2C Timeout Register</td>
</tr>
<tr>
<td>[0x0048]</td>
<td>I2Cn_DMA</td>
<td>I2C DMA Enable Register</td>
</tr>
<tr>
<td>[0x004C]</td>
<td>I2Cn_SLV_ADDR</td>
<td>I2C Slave Address Register</td>
</tr>
</tbody>
</table>

### 13.6 Register Details

**Table 13-6: I2C Control 0 Register**

<table>
<thead>
<tr>
<th></th>
<th>I2C Control 0</th>
<th>I2Cn_CTRL0</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
<td>Reset</td>
</tr>
<tr>
<td>31:16</td>
<td>-</td>
<td>RO</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>hsmode</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>-</td>
<td>RO</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>scl_ppm</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>scl_strd</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>read</td>
<td>R</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>swoe</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### I2C Control 0

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>sda</td>
<td>R</td>
<td>-</td>
<td>SDA Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: SDA pin is logic low.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: SDA pin is logic high.</td>
</tr>
<tr>
<td>8</td>
<td>scl</td>
<td>R</td>
<td>-</td>
<td>SCL Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: SCL pin is logic low.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: SCL pin is logic high.</td>
</tr>
<tr>
<td>7</td>
<td>sdao</td>
<td>R/W</td>
<td>0</td>
<td>SDA Pin Output Control</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set the state of the SDA hardware pin (actively pull low or float).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Pull SDA Low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Release SDA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Only valid when I2Cn_CTRL0.swoe=1</td>
</tr>
<tr>
<td>6</td>
<td>sclo</td>
<td>R/W</td>
<td>0</td>
<td>SCL Pin Output Control</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set the state of the SCL hardware pin (actively pull low or float).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Pull SCL Low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Release SCL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Only valid when I2Cn_CTRL0.swoe=1</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>4</td>
<td>ack</td>
<td>R/W</td>
<td>0</td>
<td>Interactive Receive Mode (IRXM) Acknowledge</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If iRXM is enabled (I2Cn_CTRL0.irxm = 1), this field determines if the hardware sends an ACK or a NACK to an IRM transaction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Respond to IRXM with ACK</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Respond to IRXM with NACK</td>
</tr>
<tr>
<td>3</td>
<td>irxm</td>
<td>R/W</td>
<td>0</td>
<td>Interactive Receive Mode (IRXM)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When receiving data, allows for an Interactive Receive Mode (IRM) interrupt event after each received byte of data. The I2C peripheral hardware can be enabled to send either an ACK or NACK for IRXM. See Interactive Receive Mode section for detailed information.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Only set this field when the I2C bus is inactive.</td>
</tr>
<tr>
<td>2</td>
<td>gcen</td>
<td>R/W</td>
<td>0</td>
<td>General Call Address Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Ignore General Call Address</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Acknowledge General Call Address</td>
</tr>
<tr>
<td>1</td>
<td>mst</td>
<td>R/W</td>
<td>0</td>
<td>Master Mode Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Slave mode enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Master mode enabled.</td>
</tr>
<tr>
<td>0</td>
<td>i2cen</td>
<td>R/W</td>
<td>0</td>
<td>I2C Peripheral Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
</tbody>
</table>

### Table 13-7: I2C Status Register

<table>
<thead>
<tr>
<th>I2C Status</th>
<th>I2Cn_STAT</th>
<th>[0x0004]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:6</td>
<td>-</td>
<td>RO</td>
</tr>
</tbody>
</table>
### I2C Status

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 5    | ckm  | RO     | 0     | Master Mode \(^\text{I}^2\text{C}\) Bus Transaction Active  
The peripheral is operating in Master mode and a valid transaction beginning with a \(\text{START}\) command is in progress on the \(^\text{I}^2\text{C}\) bus. This bit will read 1 until the master ends the transaction with a \(\text{STOP}\) command. This bit will continue to read 1 while a slave performs clock stretching.  
0: Device not actively driving SCL clock cycles.  
1: Device operating as master and actively driving SCL clock cycles. |
| 4    | txf  | RO     | 0     | TX FIFO Full  
0: Not full  
1: Full |
| 3    | txe  | RO     | 1     | TX FIFO Empty  
0: Not empty  
1: Empty |
| 2    | rxf  | RO     | 0     | RX FIFO Full  
0: Not full  
1: Full |
| 1    | rxe  | RO     | 1     | RX FIFO Empty  
0: Not empty  
1: Empty |
| 0    | busy | RO     | 0     | Master or Slave Mode \(^\text{I}^2\text{C}\) Bus Transaction Active  
The peripheral is operating in Master or Slave mode and a valid transaction beginning with a \(\text{START}\) command is in progress on the \(^\text{I}^2\text{C}\) bus. This bit will read 1 until the peripheral acting as a master or an external master ends the transaction with a \(\text{STOP}\) command. This bit will continue to read 1 while a slave performs clock stretching.  
0: \(^\text{I}^2\text{C}\) bus is idle.  
1: \(^\text{I}^2\text{C}\) bus transaction in progress. |

### Table 13-8: \(^\text{I}^2\text{C}\) Interrupt Flag 0 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
| 23   | wrami | R/W1C  | 0     | Slave Write Address Match Interrupt Flag  
If set, the device has been accessed for a write (i.e. receive) transaction in slave mode and the address received matches the device slave address.  
0: No address match.  
1: Address match. |
| 22   | rdami | R/W1C  | 0     | Slave Read Address Match Interrupt Flag  
If set, the device has been accessed for a read (i.e. transmit) transaction in slave mode and the address received matches the device slave address.  
0: No address match.  
1: Address match. |
| 21:16 | -     | RO     | 0     | Reserved |
| 15   | txloi | R/W1C  | 0     | TX FIFO Locked Interrupt Flag  
If set, the TX FIFO is locked and writes to the TX FIFO are ignored. When set, the TX FIFO is automatically flushed. Writes to the TX FIFO are ignored until this flag is cleared. Write 1 to clear.  
0: TX FIFO not locked.  
1: TX FIFO is locked and all writes to the TX FIFO are ignored. |
<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>stoperi</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>Out of Sequence STOP Interrupt Flag</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This flag is set if a STOP condition occurs out of expected sequence. Write 1 to clear this field. Writing 0 has no effect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Error condition has not occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Out of sequence STOP condition occurred.</td>
</tr>
<tr>
<td>13</td>
<td>strteri</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>Out of Sequence START Interrupt Flag</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This flag is set if a START condition occurs out of expected sequence. Write 1 to clear this field. Writing 0 has no effect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Error condition has not occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Out of sequence START condition occurred.</td>
</tr>
<tr>
<td>12</td>
<td>dnreri</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>Slave Mode Do Not Respond Interrupt Flag</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This occurs if an address match is made, but the TX FIFO or RX FIFO is not ready. Write 1 to clear this field. Writing 0 has no effect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Error condition has not occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: I2C address match has occurred and either the TX or RX FIFO is not configured.</td>
</tr>
<tr>
<td>11</td>
<td>dateri</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>Master Mode Data NACK from External Slave Interrupt Flag</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Error condition has not occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Data NACK received from a slave.</td>
</tr>
<tr>
<td>10</td>
<td>adrer1</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>Master Mode Address NACK from Slave Error Flag</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This flag is set by hardware if an Address NACK is received from a slave bus. This flag is only valid if the I2Cn peripheral is configured for Master Mode operation. Write 1 to clear. Write 0 has no effect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Error condition has not occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Address NACK received from a slave.</td>
</tr>
<tr>
<td>9</td>
<td>toeri</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>Timeout Error Interrupt Flag</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This occurs when this device holds SCL low longer than the programmed timeout value. Applies to both Master and Slave Mode. Write 1 to clear. Write 0 has no effect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Timeout error has not occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Timeout error occurred.</td>
</tr>
<tr>
<td>8</td>
<td>arberi</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>Master Mode Arbitration Lost Interrupt Flag</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to clear. Write 0 has no effect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Condition has not occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Condition occurred.</td>
</tr>
<tr>
<td>7</td>
<td>adracki</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>Master Mode Address ACK from External Slave Interrupt Flag</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This field is set when a slave address ACK is received. Write 1 to clear. Write 0 has no effect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Condition has not occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: The slave device ACK for the address was received.</td>
</tr>
<tr>
<td>6</td>
<td>stopi</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>Slave Mode STOP Condition Interrupt Flag</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This flag is set by hardware when a STOP condition is detected. Write 1 to clear. Write 0 has no effect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Condition has not occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Condition occurred.</td>
</tr>
</tbody>
</table>
### Table 13-9: I2C Interrupt Flag 0 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>donei</td>
<td>R/W</td>
<td>0</td>
<td>Transfer Complete Interrupt Flag</td>
</tr>
<tr>
<td>0</td>
<td>irxmi</td>
<td>R/W</td>
<td>0</td>
<td>Interactive Receive Mode Interrupt Flag</td>
</tr>
<tr>
<td>0</td>
<td>ami</td>
<td>R/W</td>
<td>0</td>
<td>Slave Mode Incoming Address Match Status Interrupt Flag</td>
</tr>
<tr>
<td>0</td>
<td>gci</td>
<td>R/W</td>
<td>0</td>
<td>Slave Mode General Call Address Match Received Interrupt Flag</td>
</tr>
<tr>
<td>1</td>
<td>wramie</td>
<td>R/W</td>
<td>0</td>
<td>Slave Write Address Match Interrupt Enable</td>
</tr>
<tr>
<td>1</td>
<td>rdamie</td>
<td>R/W</td>
<td>0</td>
<td>Slave Read Address Match Interrupt Enable</td>
</tr>
</tbody>
</table>

### Table 13-9: I2C Interrupt Enable 0 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>23:21</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>15</td>
<td>txloie</td>
<td>R/W</td>
<td>0</td>
<td>TX FIFO Lock Out Interrupt Enable</td>
</tr>
<tr>
<td>22</td>
<td>rdamie</td>
<td>R/W</td>
<td>0</td>
<td>Slave Read Address Match Interrupt Enable</td>
</tr>
<tr>
<td>23</td>
<td>wramie</td>
<td>R/W</td>
<td>0</td>
<td>Slave Write Address Match Interrupt Enable</td>
</tr>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
<td>Reset</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>--------</td>
<td>-------</td>
<td>-------------------------------------------------------------------</td>
</tr>
<tr>
<td>14</td>
<td>stoperie</td>
<td>R/W</td>
<td>0</td>
<td>Out of Sequence STOP Condition Detected Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>13</td>
<td>strterie</td>
<td>R/W</td>
<td>0</td>
<td>Out of Sequence START Condition Detected Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>12</td>
<td>dnerrie</td>
<td>R/W</td>
<td>0</td>
<td>Slave Mode Do Not Respond Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set this field to enable interrupts in Slave Mode when the “Do Not Respond” condition occurs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled.</td>
</tr>
<tr>
<td>11</td>
<td>daterie</td>
<td>R/W</td>
<td>0</td>
<td>Master Mode Received Data NACK from Slave Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>10</td>
<td>adrierie</td>
<td>R/W</td>
<td>0</td>
<td>Master Mode Received Address NACK from Slave Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>9</td>
<td>toerie</td>
<td>R/W</td>
<td>0</td>
<td>Timeout Error Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>8</td>
<td>arberie</td>
<td>R/W</td>
<td>0</td>
<td>Master Mode Arbitration Lost Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>7</td>
<td>adrackie</td>
<td>R/W</td>
<td>0</td>
<td>Received Address ACK from Slave Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set this field to enable interrupts for Master Mode slave device address ACK events.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled.</td>
</tr>
<tr>
<td>6</td>
<td>stopie</td>
<td>R/W</td>
<td>0</td>
<td>STOP Condition Detected Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>5</td>
<td>txthie</td>
<td>R/W</td>
<td>0</td>
<td>TX FIFO Threshold Level Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>4</td>
<td>rxthie</td>
<td>R/W</td>
<td>0</td>
<td>RX FIFO Threshold Level Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>3</td>
<td>amie</td>
<td>R/W</td>
<td>0</td>
<td>Slave Mode Incoming Address Match Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>2</td>
<td>gcie</td>
<td>R/W</td>
<td>0</td>
<td>Slave Mode General Call Address Match Received Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>1</td>
<td>irxmie</td>
<td>R/W</td>
<td>0</td>
<td>Interactive Receive Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>0</td>
<td>doneie</td>
<td>R/W</td>
<td>0</td>
<td>Transfer Complete Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
</tbody>
</table>
### Table 13-10: I2C Interrupt Flag 1 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:3</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>starti</td>
<td>R/W1C</td>
<td>0</td>
<td>START Condition Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If set, a device START condition has been detected.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: START condition not detected.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: START condition detected.</td>
</tr>
<tr>
<td>1</td>
<td>txufi</td>
<td>R/W1C</td>
<td>0</td>
<td>Slave Mode: TX FIFO Underflow Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In Slave Mode operation, the hardware sets this flag automatically if the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TX FIFO is empty and the master requests more data by sending an ACK after</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the previous byte transferred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Slave mode TX FIFO underflow condition has not occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Slave mode TX FIFO underflow condition occurred.</td>
</tr>
<tr>
<td>0</td>
<td>rxofi</td>
<td>R/W1C</td>
<td>0</td>
<td>Slave Mode: RX FIFO Overflow Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In Slave Mode operation, the hardware sets this flag automatically when an</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RX FIFO overflow occurs. Write 1 to clear. Write 0 has no effect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Slave mode RX FIFO overflow event has not occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Slave mode RX FIFO overflow condition occurred (data lost).</td>
</tr>
</tbody>
</table>

### Table 13-11: I2C Interrupt Enable 1 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:3</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>startie</td>
<td>R/W</td>
<td>0</td>
<td>START Condition Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>1</td>
<td>txufie</td>
<td>R/W</td>
<td>0</td>
<td>Slave Mode TX FIFO Underflow Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>0</td>
<td>rxofie</td>
<td>R/W</td>
<td>0</td>
<td>Slave Mode RX FIFO Overflow Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
</tbody>
</table>

### Table 13-12: I2C FIFO Length Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>15:8</td>
<td>txlen</td>
<td>RO</td>
<td>8</td>
<td>TX FIFO Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Returns the length of the TX FIFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8: 8-byte TX FIFO.</td>
</tr>
<tr>
<td>7:0</td>
<td>rxlen</td>
<td>RO</td>
<td>8</td>
<td>RX FIFO Length</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Returns the length of the RX FIFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8: 8-byte RX FIFO.</td>
</tr>
</tbody>
</table>

### Table 13-13: I2C Receive Control 0 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:12</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
### I2C Receive Control 0

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11:8</td>
<td>rxth</td>
<td>R/W</td>
<td>0</td>
<td>RX FIFO Threshold Level</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set this field to the required number of bytes to trigger a RX FIFO threshold event. When the number of bytes in the RX FIFO is equal to or greater than this field, the hardware sets the `i2cn_INT_FLO.rxth` bit indicating an RX FIFO threshold level event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: 0 bytes or more in the RX FIFO causes a threshold event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: 1+ bytes in the RX FIFO triggers a receive threshold event (recommended minimum value).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8: RX FIFO threshold event only occurs when the RX FIFO is full.</td>
</tr>
<tr>
<td>7</td>
<td>rxfsh</td>
<td>R/W1O</td>
<td>0</td>
<td>Flush RX FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to this field to initiate a RX FIFO flush, clearing all data in the RX FIFO. This field is automatically cleared by hardware when the RX FIFO flush completes. Writing 0 has no effect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: RX FIFO flush complete or not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Flush the RX FIFO</td>
</tr>
<tr>
<td>6:1</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>0</td>
<td>dnr</td>
<td>R/W</td>
<td>0</td>
<td>Slave Mode Do Not Respond</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Slave mode operation only. If the device has been addressed for a write operation, and there is still data in the RX_FIFO then:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Always respond to an address match with an ACK but then always respond to data bytes with a NACK.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: NACK the address.</td>
</tr>
</tbody>
</table>

#### Table 13-14: I\(^2\)C Receive Control 1 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:12</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>11:8</td>
<td>rxfifo</td>
<td>R</td>
<td>0</td>
<td>RX FIFO Byte Count Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No data in the RX FIFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: 1 byte in the RX FIFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2: 2 bytes in the RX FIFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3: 3 bytes in the RX FIFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4: 4 bytes in the RX FIFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5: 5 bytes in the RX FIFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6: 6 bytes in the RX FIFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7: 7 bytes in the RX FIFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8: 8 bytes in the RX FIFO (max value).</td>
</tr>
<tr>
<td>7:0</td>
<td>rxcnt</td>
<td>R/W</td>
<td>1</td>
<td>RX FIFO Transaction Byte Count Configuration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When in Master Mode, write the number of bytes to be received in a transaction from 1 to 256. 0x00 represents 256.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: 256 byte receive transaction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: 1 byte receive transaction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2: 2 byte receive transaction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>255: 255 byte receive transaction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This field is ignored when `i2cn_CTRL0.irxm` = 1. To receive more than 256 bytes, use `i2cn_CTRL0.irxm` = 1.</td>
</tr>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
<td>Reset</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>----------------</td>
<td>--------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>31:12</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>11:8</td>
<td>txth</td>
<td>R/W</td>
<td>0</td>
<td>TX FIFO Threshold Level</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sets the level for a Transmit FIFO threshold event interrupt.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If the number of bytes remaining in the TX FIFO falls to this level or lower</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the interrupt flag i2Cn_INT_FL0.txthi is set indicating a TX FIFO Threshold</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Event occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: 0 bytes remaining in the TX FIFO triggers a TX FIFO threshold event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: 1 byte or fewer remaining in the TX FIFO triggers a TX FIFO threshold</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>event (recommended minimum value).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7: 7 or fewer bytes remaining in the TX FIFO triggers a TX FIFO threshold</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>event.</td>
</tr>
<tr>
<td>7</td>
<td>txfsh</td>
<td>R/W1O</td>
<td>0</td>
<td>TX FIFO Flush</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A TX FIFO flush clears all remaining data from the transmit FIFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: TX FIFO flush is complete or not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Flush the TX FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Hardware automatically clears this bit to 0 after it is written to 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>when the flush is completed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If i2Cn_INT_FL0.txloi = 1, then i2Cn_TX_CTRL0.txfsh = 1.</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>5</td>
<td>nacktxafdis</td>
<td>R/W</td>
<td>0</td>
<td>TX FIFO received NACK Auto Flush Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Various situations or conditions are described in this user guide that</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>lead to the Transmit FIFO being flushed and locked out (i2Cn_INT_FL0.txloi</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 1).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Received NACK at end of Slave Transmit operation enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Received NACK at end of Slave Transmit operation disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: upon entering TX Preload Mode, hardware will automatically set this</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit to 0 after it is written to 1 when the flush is completed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Software can subsequently set to any value desired (ie HW does not</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>continuously force the bitfield to this value).</td>
</tr>
<tr>
<td>4</td>
<td>samrtxafdis</td>
<td>R/W</td>
<td>0</td>
<td>TX FIFO Slave Address Match Read Auto Flush Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Various situations or conditions are described in this user guide that</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>lead to the Transmit FIFO being flushed and locked out (i2Cn_INT_FL0.txloi</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 1).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: upon entering TX Preload Mode, hardware will automatically set this</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit to 1 after it is written to 1 when the flush is completed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Software can subsequently set to any value desired (ie HW does not</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>continuously force the bitfield to this value).</td>
</tr>
<tr>
<td>3</td>
<td>samwtxafdis</td>
<td>R/W</td>
<td>0</td>
<td>TX FIFO Slave Address Match Write Auto Flush Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Various situations or conditions are described in this user guide that</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>lead to the Transmit FIFO being flushed and locked out (i2Cn_INT_FL0.txloi</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>= 1).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: upon entering TX Preload Mode, hardware will automatically set this</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bit to 1 after it is written to 1 when the flush is completed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Software can subsequently set to any value desired (ie HW does not</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>continuously force the bitfield to this value).</td>
</tr>
</tbody>
</table>
### Table 13-16: I2C Transmit Control 1 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:12</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>11:8</td>
<td>txfifo</td>
<td>R</td>
<td>0</td>
<td>Transmit FIFO Byte Count Status</td>
</tr>
<tr>
<td>7:1</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>0</td>
<td>txrdy</td>
<td>R/1</td>
<td>1</td>
<td>Transmit FIFO Preload Ready Status</td>
</tr>
</tbody>
</table>

### Table 13-17: I2C Data Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
### Table 13-18: I²C Master Mode Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:11</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>10:8</td>
<td>mcode</td>
<td>R/W</td>
<td>0</td>
<td>MCODE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>These bits set the master code used in Hs-mode operation</td>
</tr>
<tr>
<td>7</td>
<td>sea</td>
<td>R/W</td>
<td>0</td>
<td>Slave Extended Addressing Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Send a 7-bit address to the slave</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Send a 10-bit address to the slave</td>
</tr>
<tr>
<td>6:3</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>stop</td>
<td>R/W1O</td>
<td>0</td>
<td>Send STOP Condition</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Send a STOP Condition at the end of the current transaction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: This bit is automatically cleared by hardware when the STOP condition begins.</td>
</tr>
<tr>
<td>1</td>
<td>restart</td>
<td>R/W1O</td>
<td>0</td>
<td>Send Repeated START Condition</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>After sending data to a slave, the master may send another START to retain</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>control of the bus.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Send a Repeated START condition to Slave instead of sending a STOP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>condition at the end of the current transaction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: This bit is automatically cleared by hardware when the repeated START</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>condition begins.</td>
</tr>
<tr>
<td>0</td>
<td>start</td>
<td>R/W1O</td>
<td>0</td>
<td>Start Master Mode Transfer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Start Master Mode Transfer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: This bit is automatically cleared by hardware when the transfer is completed or aborted.</td>
</tr>
</tbody>
</table>

### Table 13-19: I²C SCL Low Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:9</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>8:0</td>
<td>scl_lo</td>
<td>R/W</td>
<td>0x001</td>
<td>Clock Low Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In Master Mode, this configures the SCL low time.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[t_{\text{SCL,LO}} = f_{\text{I2C,CLK}} \times (\text{scl_lo} + 1)]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: 0 is not a valid setting for this field.</td>
</tr>
</tbody>
</table>

### Table 13-20: I²C SCL High Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:9</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
### I²C Clock High Control

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:0</td>
<td>scl_hi</td>
<td>R/W</td>
<td>0x001</td>
<td>Clock High Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In Master Mode, this configures the SCL high time.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[ t_{\text{SCL,HI}} = \frac{1}{I_{\text{I2C_CLK}}} \times (\text{scl_hi} + 1) ]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In both Master and Slave Mode, this also configures the time SCL is held low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>after new data is loaded from the TX FIFO or after firmware clears irxmi.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>during Interactive Receive Mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: 0 is not a valid setting for this field.</td>
</tr>
</tbody>
</table>

### I²C Hs-Mode Clock Control Register

<table>
<thead>
<tr>
<th>I²C Hs-Mode Clock Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Table 13-21</strong></td>
<td></td>
</tr>
<tr>
<td><strong>I²C Hs-Mode Clock Control</strong></td>
<td></td>
</tr>
<tr>
<td>Bits</td>
<td>Access</td>
</tr>
<tr>
<td>31:16</td>
<td>-</td>
</tr>
<tr>
<td>15:8</td>
<td>hs_clk_hi</td>
</tr>
<tr>
<td>7:0</td>
<td>hs_clk_lo</td>
</tr>
</tbody>
</table>

### I²C Timeout Register

<table>
<thead>
<tr>
<th>I²C Timeout</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Table 13-22</strong></td>
<td></td>
</tr>
<tr>
<td><strong>I²C Timeout</strong></td>
<td></td>
</tr>
<tr>
<td>Bits</td>
<td>Access</td>
</tr>
<tr>
<td>31:16</td>
<td>-</td>
</tr>
<tr>
<td>15:0</td>
<td>to</td>
</tr>
</tbody>
</table>

### I²C DMA Register

<table>
<thead>
<tr>
<th>I²C DMA</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Table 13-23</strong></td>
<td></td>
</tr>
<tr>
<td><strong>I²C DMA</strong></td>
<td></td>
</tr>
<tr>
<td>Bits</td>
<td>Access</td>
</tr>
<tr>
<td>31:2</td>
<td>-</td>
</tr>
</tbody>
</table>
### Table 13-24: I²C Slave Address Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>15</td>
<td>ea</td>
<td>R/W</td>
<td>0</td>
<td>Slave Mode Extended Address Length Select</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: 7-bit addressing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: 10-bit addressing.</td>
</tr>
<tr>
<td>14:10</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>9:0</td>
<td>sla</td>
<td>R/W</td>
<td>0</td>
<td>Slave Mode Slave Address</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>In Slave Mode Operation, (I2Cn_CTRL0.mstr=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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> (I2Cn_SLV_ADDR.ea) controls if this field is a 7-bit or 10-bit address.</td>
</tr>
</tbody>
</table>
14. Quad Serial Peripheral Interface (SPI)

The Quad Serial Peripheral Interface (QSPI) 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, single, dual or quad data lines, and one or more slave select lines for communication with external SPI devices. An SPI network uses a single master and one or more slaves for any given transaction.

The provided QSPI ports support full-duplex, bi-direction I/O and each QSPI includes a Bit Rate Generator (BRG) for generating the clock signal when operating in master mode. Each QSPI 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 1 to 16-bit characters
- 3-wire and 4-wire SPI operation for single-bit communication
- Single, Dual and Quad I/O
- 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 14-1: QSPI Block Diagram* shows the structure of the peripheral. See *Table 14-1: MAX32665—MAX32668 SPI Instances* for the peripheral-specific peripheral bus assignment and bit rate generator clock source.
Figure 14-1: QSPI Block Diagram

![QSPI Block Diagram](image)

* The number of slave select signals varies can vary for each instance of the peripheral.
** The bus interface (APB or AHB) can vary for each instance of the peripheral.

### 14.1 Instances

The following instances of the peripheral are provided. For a specific instance replace n in register names with either 0, 1, or 2 depending on the instance of the peripheral.

**Table 14-1: MAX32665—MAX32668 SPI Instances**

<table>
<thead>
<tr>
<th>Name</th>
<th>Formats</th>
<th>Bus Assignment</th>
<th>Bit Rate Generator</th>
<th>Slave Select Signals</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3-Wire</td>
<td>4-Wire</td>
<td>Dual</td>
<td>Quad data</td>
</tr>
<tr>
<td>SPI0</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SPI1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SPI2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Table 14-2: MAX32665—MAX32668 QSPI Signal Mapping shows the mapping of the QSPI alternate functions.

### Table 14-2: MAX32665—MAX32668 QSPI Signal Mapping

<table>
<thead>
<tr>
<th>QSPI Port to Pin Mapping</th>
<th>AF1</th>
<th>AF2</th>
<th>AF2</th>
<th>AF2</th>
</tr>
</thead>
<tbody>
<tr>
<td>QSPI0</td>
<td>P1.11</td>
<td>P0.2</td>
<td>P0.19</td>
<td>P0.27</td>
</tr>
<tr>
<td>QSPI1</td>
<td>P1.1</td>
<td>P0.0</td>
<td>P0.17</td>
<td>P0.25</td>
</tr>
<tr>
<td>QSPI2</td>
<td>P1.10</td>
<td>P0.12</td>
<td>P0.20</td>
<td>P0.28</td>
</tr>
</tbody>
</table>

#### 14.2 SPI Formats

##### 14.2.1 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 14-3, below.

### Table 14-3: Four-Wire Format Signals

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCK</td>
<td>Serial Clock</td>
<td>The master generates the Serial Clock signal, which is an output from the master and an input to the slave.</td>
</tr>
<tr>
<td>MOSI</td>
<td>Master Output Slave Input</td>
<td>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.</td>
</tr>
<tr>
<td>MISO</td>
<td>Master Input Slave Output</td>
<td>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.</td>
</tr>
<tr>
<td>SS</td>
<td>Slave Select</td>
<td>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 slaves. In slave mode QSPIn_SS0 is a dedicated input which indicates an external master is going to start communication. Other slave select signals into the peripheral are ignored in slave mode.</td>
</tr>
</tbody>
</table>

The MAX32665—MAX32668 supports up to three slave select lines for each instance, QSPIn_SS0, QSPIn_SS1 and QSPIn_SS2.

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

The signals in three-wire SPI operation are shown in Table 14-4: Three-Wire Format Signals. The MOSI signal is used as a bi-directional, half-duplex I/O referred to as Slave Input Slave Output (SISO). Three-wire SPI also uses a serial clock signal generated by the master and a slave select pin controlled by the master.

**Table 14-4: Three-Wire Format Signals**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCK</td>
<td>Serial Clock</td>
<td>The master generates the serial clock signal, which is an output from the master and an input to the slave.</td>
</tr>
<tr>
<td>SISO</td>
<td>Slave Input Slave Output</td>
<td>This is a half-duplex, bidirectional I/O pin used for communication between the SPI master and. This signal is used to transmit data from the master to the slave and to receive data from the slave by the master.</td>
</tr>
<tr>
<td>SS</td>
<td>Slave Select</td>
<td>In master mode, this signal is an output used to select a slave device prior to communication. In slave mode QSPIn_SS0 is a dedicated input which indicates an external master is going to start communication. Other slave select signals into the peripheral are ignored in slave mode.</td>
</tr>
</tbody>
</table>

A three-wire SPI network is shown in Figure 14-3, below. 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, usually on the second edge of a clock cycle. 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.
14.3 Pin Configuration

Before configuring the QSPIn peripheral, first disable any SPI activity for the port by setting the QSPIn_CTRL0.enable field to 0.

14.3.1 QSPIn Alternate Function Mapping

Pin selection and configuration is required to use the QSPIn port. The following information applies to SPI master and slave operation as well as three-wire, four-wire, dual and quad mode communications. Table 14-2 maps standard SPI signal names to the Alternate Function name. Determine the pins required for the SPI type and mode in the application and configure the required GPIO as described in the following sections.

The Alternate Function Name column in Table 14-2 maps the standard SPI signal name to the Alternate Function name. Refer to the data sheet for pin availability for a specific package.

When the QSPIn port is disabled, QSPIn_CTRL0.enable = 0, the GPIO pins enabled for QSPI alternate function are placed in high-impedance input mode.

14.3.2 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. The following steps set up QSPI1 for four wire SPI with three Slave Select lines.

Note: QSPI0 provides SPI pins on Alternate Function 1 and Alternate Function 2. 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.

14.3.3 Three-Wire Format Configuration

Three wire SPI uses SCK, MOSI (SOSI) and one or more slave select pins for an SPI transaction. Three-wire SPI configuration is identical to the four wire configuration except QSPIn_MISO does not need set up for the QSPI alternate function. The direction of communication in three-wire SPI mode is controlled by the QSPI 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.
14.3.4 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 14-4, below. The configuration of the GPIO pins for Dual mode SPI is identical to four wire SPI and the mode is controlled by setting QSPIn_CTRL2.data_width to 1 indicating to the QSPIn hardware to use SDIO0 and SDIO1 for half-duplex communication rather than full-duplex communication.

Figure 14-4: Dual Mode SPI Connection Diagram

14.3.5 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 QSPIn_CTRL2.data_width to 2.

14.4 QSPI Clock Configuration

14.4.1 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 QSPIn is set to master mode, the QSPIn bit rate generator creates the serial clock and outputs it on the configured QSPIn_SCK pin. When QSPIn is configured for slave operation the QSPIn_SCK pin is an input from the external master and the QSPIn hardware synchronizes communications using the SCK input. Operating as a slave, if a QSPIn slave select input is not asserted, the QSPIn ignores any signals on the serial clock and serial data lines.

When QSPIn is configured for slave operation, the maximum SCK input frequency supported is \( f_{\text{SCK}} = \frac{f_{\text{PCLK}}}{8} \). For example, if \( f_{\text{PCLK}} = 48\text{MHz} \), the maximum SPI clock frequency supported in slave mode for QSPIn is 6MHz.

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, QSPIn_CTRL2.phase. The SCK clock polarity field, QSPIn_CTRL2.clkpol, controls if the SCK signal is active high or active low.

The QSPIn peripheral supports four combinations of SCK phase and polarity referred to as SPI Modes 0, 1, 2, and 3. Clock Polarity (QSPIn_CTRL2.clkpol) selects an active low/high clock and has no effect on the transfer format. Clock Phase (QSPIn_CTRL2.phase) selects one of two different transfer formats.

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

The System Peripheral Clock, PCLK, drives the QSPIn peripheral clock. The SPI0 provides an internal clock, SPI0_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 SPI0 internal clock using the field QSPIn_CLK_CFG.scale as shown in Equation 14-1. Valid settings for QSPIn_CLK_CFG.scale are 0 to 8, allowing a divisor of 1 to 256.

**Equation 14-1: SPI Peripheral Clock**

\[
\frac{f_{SPI_CLK}}{2^{\text{scale}}}
\]

14.4.3 **Master Mode Serial Clock Generation**

In master and multi-master mode the SCK clock is generated by the master. The SPI0 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 QSPIn peripheral clock as a base value and the high and low values are a count of the number of fSPI_CLK clocks. Figure 14-5, below, visually represents the use of the QSPIn_CLK_CFG.hi and QSPIn_CLK_CFG.lo fields for a non-50% duty cycle serial clock generation. See Equation 14-2 and Equation 14-3 for calculating the SCK high and low time from the QSPIn_CLK_CFG.hi and QSPIn_CLK_CFG.lo field values.

**Figure 14-5: SCK Clock Rate Control**

![Diagram of SCK Clock Rate Control]

**Equation 14-2: SCK High Time**

\[
t_{SCK,\text{HI}} = t_{QSPICLK} \times \text{QSPIn_CLK_CFG}.\text{hi}
\]

**Equation 14-3: SCK Low Time**

\[
t_{SCK,\text{LOW}} = t_{QSPICLK} \times \text{QSPIn_CLK_CFG}.\text{lo}
\]

14.4.4 **Clock Phase and Polarity Control**

QSPIn supports four combinations of clock and phase polarity as shown in Table 14-5, below. Clock polarity is controlled using the bit QSPIn_CTRL2.clkpol and determines if the clock is active high or active low as shown in Figure 14-6. 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, QSPIn_CTRL2.phase = 0, dictates the SPI data is sampled on the initial SPI clock edge regardless of clock polarity. Phase 1, QSPIn_CTRL2.phase = 1, results in data sample occurring on the second edge of the clock regardless of 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 14-5. SPI Modes Clock Phase and Polarity Operation**

<table>
<thead>
<tr>
<th>SPI Mode</th>
<th>QSPIn_CTRL2_phase</th>
<th>QSPIn_CTRL2_clkpol</th>
<th>SCK Transmit Edge</th>
<th>SCK Receive Edge</th>
<th>SCK Idle State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Falling</td>
<td>Rising</td>
<td>Low</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Rising</td>
<td>Falling</td>
<td>High</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>Rising</td>
<td>Falling</td>
<td>Low</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>Falling</td>
<td>Rising</td>
<td>High</td>
</tr>
</tbody>
</table>

**14.4.5 QSPIn 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.

**14.4.6 QSPI Interrupts and Wakeups**

The QSPI supports multiple interrupt sources. Status flags for each interrupt are set regardless of the state of the interrupt enable bit for that event. The event happens once when the condition is satisfied. The status flag must be cleared by firmware by writing a 1 to the interrupt flag.
The following FIFO interrupts are supported:

- Transmit FIFO Empty
- Transmit FIFO Threshold
- Receive FIFO Full
- Receive FIFO threshold
- Transmit FIFO Underrun
  - Slave mode only, master mode stalls the serial clock
- Transmit FIFO Overrun
- Receive FIFO Underrun
- Receive FIFO Overrun (Slave Mode only, Master Mode will stall the clock)

QSPI supports interrupts for the internal state of the QSPI as well as external signals. The following transmission interrupts are supported:

- SSn asserted or deasserted
- SPI transaction complete
- Slave mode transaction aborted
- Multi-master fault

The QSPI port can wake up the microcontroller for low-power modes when the wake event is enabled. QSPI events that can wake the microcontroller are:

- Receive FIFO full
- Transmit FIFO empty
- Receive FIFO threshold
- Transmit FIFO threshold

### 14.5 Registers

See Table 3-1: APB Peripheral Base Address Map for the QSPI0 and QSPI1 Peripheral Base Address. See Table 3-2: AHB Peripheral Base Address Map for the QSPI2 Peripheral Base Address.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register Name</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>QSPIn_DATA</td>
<td>R/W</td>
<td>QSPIn FIFO Data Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>QSPIn_CTRL0</td>
<td>R/W</td>
<td>QSPIn Master Signals Control Register</td>
</tr>
<tr>
<td>[0x0008]</td>
<td>QSPIn_CTRL1</td>
<td>R/W</td>
<td>QSPIn Transmit Packet Size Register</td>
</tr>
<tr>
<td>[0x000C]</td>
<td>QSPIn_CTRL2</td>
<td>R/W</td>
<td>QSPIn Static Configuration Register</td>
</tr>
<tr>
<td>[0x0010]</td>
<td>QSPIn_SS_TIME</td>
<td>R/W</td>
<td>QSPIn Slave Select Timing Register</td>
</tr>
<tr>
<td>[0x0014]</td>
<td>QSPIn_CLK_CFG</td>
<td>R/W</td>
<td>QSPIn Master Clock Configuration Register</td>
</tr>
<tr>
<td>[0x001C]</td>
<td>QSPIn_DMA</td>
<td>R/W</td>
<td>QSPIn DMA Control Register</td>
</tr>
<tr>
<td>[0x0020]</td>
<td>QSPIn_INT_FL</td>
<td>R/W1C</td>
<td>QSPIn Interrupt Flag Register</td>
</tr>
<tr>
<td>[0x0024]</td>
<td>QSPIn_INT_EN</td>
<td>R/W</td>
<td>QSPIn Interrupt Enable Register</td>
</tr>
<tr>
<td>[0x0028]</td>
<td>QSPIn_WAKE_FL</td>
<td>R/W1C</td>
<td>QSPIn Wakeup Flags Register</td>
</tr>
<tr>
<td>[0x002C]</td>
<td>QSPIn_WAKE_EN</td>
<td>R/W</td>
<td>QSPIn Wakeup Enable Register</td>
</tr>
<tr>
<td>[0x0030]</td>
<td>QSPIn_STAT</td>
<td>RO</td>
<td>QSPIn Status Register</td>
</tr>
</tbody>
</table>
### 14.6 Register Details

#### Table 14-7: QSPIn FIFO Data Register

<table>
<thead>
<tr>
<th>QSPIn FIFO Data Register</th>
<th>QSPIn_DATA</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R/W</td>
</tr>
<tr>
<td>31:0</td>
<td>qspififo</td>
<td>R/W</td>
</tr>
</tbody>
</table>

#### Table 14-8: QSPIn Control 0 Register

<table>
<thead>
<tr>
<th>QSPIn Control 0 Register</th>
<th>QSPIn_CTRL0</th>
<th>[0x0004]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td></td>
<td></td>
<td>R/W</td>
</tr>
<tr>
<td>31:19</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ss</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:9</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td>ss_ctrl</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:6</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td>start</td>
<td>R/W1O</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### QSPIn Control 0 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>ss_io</td>
<td>R/W</td>
<td>0</td>
<td>Master Slave Select Signal Direction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set the I/O direction for</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Slave Select is an output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Slave Select is an input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: This field is only used when the QSPIn is configured for Master Mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(QSPIn_CTRL0.mm_en = 1).</td>
</tr>
<tr>
<td>3:2</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>1</td>
<td>mm_en</td>
<td>R/W</td>
<td>0</td>
<td>SPI Master Mode Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This field selects between slave mode and master mode operation for the SPI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>port. Write this field to 0 to operate as an SPI slave. Setting this field</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>to 1 sets the port as an SPI master.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Slave mode SPI operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Master mode SPI operation.</td>
</tr>
<tr>
<td>0</td>
<td>enable</td>
<td>R/W</td>
<td>0</td>
<td>SPI Enable/Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This field enables and disables the QSPIn port. Disable the QSPIn port by</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>setting this field to 0. Disabling the QSPIn port does not affect the QSPIn</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FIFOs or register settings.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: QSPIn port is disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: QSPIn port is enabled</td>
</tr>
</tbody>
</table>

### QSPIn Transmit Packet Size Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>rx_num_char</td>
<td>R/W</td>
<td>0</td>
<td>Number of Receive Characters</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Number of characters to receive in RX FIFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: If the QSPIn port is set to operate in 4-wire mode, this field is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ignored and the QSPIn_CTRL1.tx_num_chars field is used for both the number</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>of characters to receive and transmit.</td>
</tr>
<tr>
<td>15:0</td>
<td>tx_num_char</td>
<td>R/W</td>
<td>0</td>
<td>Number of Transmit Characters</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Number of characters to transmit from TX FIFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: If the QSPIn port is set to operate in 4-wire mode, this field is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>used for both the number of characters to receive and transmit.</td>
</tr>
</tbody>
</table>

### QSPIn Control 2 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:20</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>19:16</td>
<td>ss_pol</td>
<td>R/W</td>
<td>0</td>
<td>Slave Select Polarity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Controls the polarity of each individual SS signal where each bit position</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>corresponds to a SS signal. QSPIn_SS0 is controlled with bit position 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>and QSPIn_SS2 is controlled with bit position 2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>For each bit position,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: SS is active low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: SS is active high</td>
</tr>
</tbody>
</table>
### QSPI In Control 2 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>three_wire</td>
<td>R/W</td>
<td>0</td>
<td><strong>Three-Wire SPI Enable</strong>&lt;br&gt;Set this field to 1 to enable three-wire SPI communication. Set this field to 0 for four-wire full-duplex SPI communication.&lt;br&gt;0: Four-wire full-duplex mode enabled.&lt;br&gt;1: Three-wire mode enabled&lt;br&gt;Note: This field is ignored for Dual SPI, QSPI_CTRL2.data_width =1, and Quad SPI, QSPI_CTRL2.data_width =2, this field is ignored.</td>
</tr>
<tr>
<td>14</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>13:12</td>
<td>data_width</td>
<td>R/W</td>
<td>0b00</td>
<td><strong>SPI Data Width</strong>&lt;br&gt;This field controls the number of data lines used for SPI communications.&lt;br&gt;<strong>Three-wire SPI, data_width = 0</strong>&lt;br&gt;Set this field to 0, indicating QSPI_MOSI (SISO) is used for half duplex communication.&lt;br&gt;<strong>Four-wire full-duplex SPI operation, data_width = 0</strong>&lt;br&gt;Set this field to 0, indicating QSPI_MOSI and QSPI_MISO are used for the SPI data output and input respectively.&lt;br&gt;<strong>Dual SPI, data_width = 1</strong>, uses QSPI_SDIO0 and QSPI_SDIO1 for half-duplex communication.&lt;br&gt;<strong>Quad SPI, data_width = 2</strong>, uses QSPI_SDIO0, QSPI_SDIO1, QSPI_SDIO2 and QSPI_SDIO3 for half-duplex communication.&lt;br&gt;0: 1-bit per SCK cycle (Three-wire half duplex SPI and Four-wire full duplex SPI)&lt;br&gt;1: 2-bits per SCK cycle (Dual SPI)&lt;br&gt;2: 4-bits per SCK cycle (Quad SPI)&lt;br&gt;3: Reserved&lt;br&gt;Note: When this field is set to 0, use the field QSPI_CTRL2.three_wire to select either Three-Wire SPI or Four-Wire SPI operation.</td>
</tr>
<tr>
<td>11:8</td>
<td>numbits</td>
<td>R/W</td>
<td>0</td>
<td><strong>Number of Bits per Character</strong>&lt;br&gt;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.&lt;br&gt;0: 16-bits per character&lt;br&gt;1: 1-bit per character&lt;br&gt;2: 2-bits per character&lt;br&gt;...&lt;br&gt;14: 14-bits per character&lt;br&gt;15: 15-bits per character&lt;br&gt;Note: 1-bit and 9-bit character lengths are not supported in Slave Mode, QSPI_CTRL0.mm_en = 0.&lt;br&gt;Note: For Dual and Quad mode SPI, the character size should be divisible by the number of bits per SCK cycle.</td>
</tr>
<tr>
<td>7:2</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
</tbody>
</table>
### QSPIn Control 2 Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>clkpol</td>
<td>R/W</td>
<td>0</td>
<td><strong>Clock Polarity</strong>&lt;br&gt;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.&lt;br&gt;0: Standard SCK for use in SPI Mode 0 and Mode 1&lt;br&gt;1: Inverted SCK for use in SPI Mode 2 and Mode 3</td>
</tr>
<tr>
<td>0</td>
<td>phase</td>
<td>R/W</td>
<td>0</td>
<td><strong>Clock Phase</strong>&lt;br&gt;0: Data sampled on clock rising edge. Use when in SPI Mode 0 and Mode 2&lt;br&gt;1: Data sampled on clock falling edge. Use when in SPI Mode 1 and Mode 3</td>
</tr>
</tbody>
</table>

### QSPIn Slave Select Timing Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>23:16</td>
<td>inact</td>
<td>R/W</td>
<td>0</td>
<td><strong>Inactive Stretch</strong>&lt;br&gt;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).&lt;br&gt;0: 256&lt;br&gt;1: 1&lt;br&gt;2: 2&lt;br&gt;3:3&lt;br&gt;…&lt;br&gt;254: 254&lt;br&gt;255: 255</td>
</tr>
<tr>
<td>15:8</td>
<td>post</td>
<td>R/W</td>
<td>0</td>
<td><strong>Slave Select Hold Post Last SCK</strong>&lt;br&gt;Number of system clock cycles that SS remains active after the last SCK edge.&lt;br&gt;0: 256&lt;br&gt;1: 1&lt;br&gt;2: 2&lt;br&gt;3:3&lt;br&gt;…&lt;br&gt;254: 254&lt;br&gt;255: 255</td>
</tr>
<tr>
<td>7:0</td>
<td>pre</td>
<td>R/W</td>
<td>0</td>
<td><strong>Slave Select Delay to First SCK</strong>&lt;br&gt;Set the number of system clock cycles the Slave Select is held active prior to the first SCK edge.&lt;br&gt;0: 256&lt;br&gt;1: 1&lt;br&gt;2: 2&lt;br&gt;3:3&lt;br&gt;…&lt;br&gt;254: 254&lt;br&gt;255: 255</td>
</tr>
</tbody>
</table>
Table 14-12: QSPIn Master Clock Configuration Registers

<table>
<thead>
<tr>
<th>QSPIn Master Clock Configuration Register</th>
<th>QSPIn_CLK_CFG</th>
<th>[0x0014]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
<td>Reset</td>
</tr>
<tr>
<td>31:20</td>
<td></td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19:16</td>
<td>scale</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:8</td>
<td>hi</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>lo</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 14-13: QSPIn DMA Control Registers

<table>
<thead>
<tr>
<th>QSPIn DMA Control Register</th>
<th>QSPIn_DMA</th>
<th>[0x001C]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
<td>Reset</td>
</tr>
<tr>
<td>31</td>
<td>rx_dma_en</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29:24</td>
<td>rx_fifo_cnt</td>
<td>R</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>rx_fifo_clear</td>
<td>W</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>rx_fifo_en</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### QSPIn DMA Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>20:16</td>
<td>rx_fifo_level</td>
<td>R/W</td>
<td>0x00</td>
<td>RX FIFO Threshold Level</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set this value to the desired RX FIFO threshold level. When the RX FIFO contains</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the number of bytes or greater than this field, a DMA request is triggered,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>and QSPIn_INT_FL.rx_thresh is set. Valid values are 0 to 30.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: 31 is an invalid setting and reserved for future use.</td>
</tr>
<tr>
<td>15</td>
<td>tx_dma_en</td>
<td>R/W</td>
<td>0</td>
<td>TX DMA Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled. Any pending DMA requests are cleared</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: TX DMA is enabled</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>13:8</td>
<td>tx_fifo_cnt</td>
<td>RO</td>
<td>0</td>
<td>Number of Bytes in the TX FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Read this field to determine the number of bytes currently in the TX FIFO.</td>
</tr>
<tr>
<td>7</td>
<td>tx_fifo_clear</td>
<td>R/W</td>
<td>0</td>
<td>TX FIFO Clear</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set this bit to clear the TX FIFO and all TX FIFO flags in the QSPIn_INT_FL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>register.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: The TX FIFO should be disabled (QSPIn_DMA.tx_fifo_en = 0) prior to setting</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>this field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Setting this field to 0 has no effect.</td>
</tr>
<tr>
<td>6</td>
<td>tx_fifo_en</td>
<td>R/W</td>
<td>0</td>
<td>TX FIFO Enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>4:0</td>
<td>tx_fifo_level</td>
<td>R/W</td>
<td>0x10</td>
<td>TX FIFO Threshold Level</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When the TX FIFO count (QSPIn_DMA.tx_fifo_cnt) falls below this value, a DMA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>request is triggered and QSPIn_INT_FL.tx_thresh is set.</td>
</tr>
</tbody>
</table>

### Table 14-14: QSPIn Interrupt Status Flags Registers

<table>
<thead>
<tr>
<th>QSPIn Interrupt Status Flags Register</th>
<th>QSPIn_INT_FL</th>
<th>[0x0020]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>31:16</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td>15</td>
<td>rx_und</td>
<td>R/1</td>
</tr>
<tr>
<td>14</td>
<td>rx_ovr</td>
<td>R/W1C</td>
</tr>
<tr>
<td>13</td>
<td>tx_und</td>
<td>R/W1C</td>
</tr>
<tr>
<td>12</td>
<td>tx_ovr</td>
<td>R/W1C</td>
</tr>
</tbody>
</table>
### QSPIn Interrupt Status Flags Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>m_done</td>
<td>R/W1C</td>
<td>0</td>
<td>Master Data Transmission Done Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set if SPI is in Master Mode, and all transactions have completed.</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>9</td>
<td>abort</td>
<td>R/W1C</td>
<td>0</td>
<td>Slave Mode Transaction Abort Detected Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set if the SPI is in Slave Mode, and SS is deasserted before a complete</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>character is received.</td>
</tr>
<tr>
<td>8</td>
<td>fault</td>
<td>R/W1C</td>
<td>0</td>
<td>Multi-Master Fault Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set if the SPI is in Master Mode, Multi-Master Mode is enabled, and a Slave</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Select input is asserted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A collision also sets this flag.</td>
</tr>
<tr>
<td>7:6</td>
<td></td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>5</td>
<td>ssd</td>
<td>R/W1C</td>
<td>0</td>
<td>Slave Select Deasserted Flag</td>
</tr>
<tr>
<td>4</td>
<td>ssa</td>
<td>R/W1C</td>
<td>0</td>
<td>Slave Select Asserted Flag</td>
</tr>
<tr>
<td>3</td>
<td>rx_full</td>
<td>R/W1C</td>
<td>0</td>
<td>RX FIFO Full Flag</td>
</tr>
<tr>
<td>2</td>
<td>rx_thresh</td>
<td>R/W1C</td>
<td>0</td>
<td>RX FIFO Threshold Level Crossed Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set when the RX FIFO exceeds the value in QSPIn_DMA.rx_fifo_level.</td>
</tr>
<tr>
<td>1</td>
<td>tx_empty</td>
<td>R/W1C</td>
<td>1</td>
<td>TX FIFO Empty Flag</td>
</tr>
<tr>
<td>0</td>
<td>tx_thresh</td>
<td>R/W1C</td>
<td>0</td>
<td>TX FIFO Threshold Level Crossed Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set when the TX FIFO is less than the value in QSPIn_DMA.tx_fifo_level.</td>
</tr>
</tbody>
</table>

**Table 14-15: QSPIn Interrupt Enable Registers**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td></td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>15</td>
<td>rx_und</td>
<td>R/W</td>
<td>0</td>
<td>RX FIFO Underrun Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt is disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is enabled</td>
</tr>
<tr>
<td>14</td>
<td>rx_ovr</td>
<td>R/W</td>
<td>0</td>
<td>RX FIFO Overrun Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt is disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is enabled</td>
</tr>
<tr>
<td>13</td>
<td>tx_und</td>
<td>R/W</td>
<td>0</td>
<td>TX FIFO Underrun Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt is disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is enabled</td>
</tr>
<tr>
<td>12</td>
<td>tx_ovr</td>
<td>R/W</td>
<td>0</td>
<td>TX FIFO Overrun Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt is disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is enabled</td>
</tr>
<tr>
<td>11</td>
<td>m_done</td>
<td>R/W</td>
<td>0</td>
<td>Master Data Transmission Done Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt is disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is enabled</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
</tbody>
</table>
### QSPIn Interrupt Enable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>abort</td>
<td>R/W</td>
<td>0</td>
<td>Slave Mode Abort Detected Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt is disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is enabled</td>
</tr>
<tr>
<td>8</td>
<td>fault</td>
<td>R/W</td>
<td>0</td>
<td>Multi-Master Fault Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt is disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is enabled</td>
</tr>
<tr>
<td>7:6</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>5</td>
<td>ssd</td>
<td>R/W</td>
<td>0</td>
<td>Slave Select Deasserted Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt is disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is enabled</td>
</tr>
<tr>
<td>4</td>
<td>ssa</td>
<td>R/W</td>
<td>0</td>
<td>Slave Select Asserted Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt is disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is enabled</td>
</tr>
<tr>
<td>3</td>
<td>rx_full</td>
<td>R/W</td>
<td>0</td>
<td>RX FIFO Full Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt is disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is enabled</td>
</tr>
<tr>
<td>2</td>
<td>rx_thresh</td>
<td>R/W</td>
<td>0</td>
<td>RX FIFO Threshold Level Crossed Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt is disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is enabled</td>
</tr>
<tr>
<td>1</td>
<td>tx_empty</td>
<td>R/W</td>
<td>0</td>
<td>TX FIFO Empty Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt is disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is enabled</td>
</tr>
<tr>
<td>0</td>
<td>tx_thresh</td>
<td>R/W</td>
<td>0</td>
<td>TX FIFO Threshold Level Crossed Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt is disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt is enabled</td>
</tr>
</tbody>
</table>

### Table 14-16: QSPIn Wakeup Status Flags Registers

<table>
<thead>
<tr>
<th>QSPIn Wakeup Flags Register</th>
<th>QSPIn_WAKE_FL</th>
<th>[0x0028]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:4</td>
<td>-</td>
<td>R/W</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>3</td>
<td>rx_full</td>
<td>R/W1C</td>
<td>Wake on RX FIFO Full Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: Wake condition has not occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Wake condition occurred.</td>
</tr>
<tr>
<td>2</td>
<td>rx_thresh</td>
<td>R/W1C</td>
<td>Wake on RX FIFO Threshold Level Crossed Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: Wake condition has not occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Wake condition occurred.</td>
</tr>
<tr>
<td>1</td>
<td>tx_empty</td>
<td>R/W1C</td>
<td>Wake on TX FIFO Empty Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: Wake condition has not occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Wake condition occurred.</td>
</tr>
<tr>
<td>0</td>
<td>tx_thresh</td>
<td>R/W1C</td>
<td>Wake on TX FIFO Threshold Level Crossed Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0: Wake condition has not occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1: Wake condition occurred.</td>
</tr>
</tbody>
</table>
Table 14-17: QSPIn Wakeup Enable Registers

<table>
<thead>
<tr>
<th>QSPIn Wakeup Enable Register</th>
<th>QSPIn_WAKE_EN</th>
<th>[0x002C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 31:4 | - | R/W | 0 | Reserved for Future Use  
Do not modify this field. |
| 3 | rx_full | R/W | 0 | Wake on RX FIFO Full Enable  
0: Wake event is disabled  
1: Wake event is enabled. |
| 2 | rx_thresh | R/W | 0 | Wake on RX FIFO Threshold Level Crossed Enable  
0: Wake event is disabled  
1: Wake event is enabled. |
| 1 | tx_empty | R/W | 0 | Wake on TX FIFO Empty Enable  
0: Wake event is disabled  
1: Wake event is enabled. |
| 0 | tx_thresh | R/W | 0 | Wake on TX FIFO Threshold Level Crossed Enable  
0: Wake event is disabled  
1: Wake event is enabled. |

Table 14-18: QSPIn Slave Select Timing Registers

<table>
<thead>
<tr>
<th>QSPIn Status Register</th>
<th>QSPIn_STAT</th>
<th>[0x0030]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 31:1 | - | R/W | 0 | Reserved for Future Use  
Do not modify this field. |
| 0 | busy | R | 0 | SPI Active Status  
0: QSPIn is not active. In master mode the busy flag is cleared when the last character is sent. In slave mode the busy field is cleared when the configured slave select input is deasserted.  
1: QSPIn is active. In master mode the busy flag is set when a transaction starts. In slave mode the busy flag is set when a configured slave select input is asserted. |
15. HTimer (HT)

15.1 Overview

The HTimer (HT) is a 40-bit binary timer similar to the real-time clock but is driven by a high-speed internal clock source. The timer provides a short-interval, auto-reload alarm and a long-interval alarm. Configurable alarm settings allow it to be used as a low-power wakeup timer.

The HT clock source is the 7.768 MHz internal oscillator.

Two registers combine to create the timer. The HTIMER_SEC.rts field contains the most significant bits and the HTIMER_SSEC.rtss field contains the least significant bits. The HTIMER_SEC.rts field is incremented each time HTIMER_SSEC.rtss rolls over.

The peripheral does not have a dedicated clock output equivalent to the 32KCAL.

A programmable long-interval alarm is usable with HTIMER_SEC.rts to provide a single event/alarm timer. When the counter is started, it counts continuously unless it is disabled, and reads of the counter registers do not affect the count.

A separate 32-bit auto-reload short-interval alarm counter register (HTIMER_RSSA) can generate repeating interval alarms.

15.2 Alarm Functions

The timer provides two alarm functions:

The long interval alarm is generated when HTIMER_RAS.ras matches HTIMER_SEC.rts[19:0].

The short-interval alarm provides an internal 32-bit auto-reload counter that increments on each transition of HTIMER_SSEC.rtss. The counter always increments from the value in HTIMER_RSSA.rssa up to the maximum value of HTIMER_SSEC.rtss. When the internal counter rolls over to 0, an alarm is generated, the internal counter is reloaded with HTIMER_RSSA.rssa and continues incrementing.

15.2.1 Long-Interval Alarm

The long interval counter increments once each time HTIMER_SSEC.rtss rolls over to 0. The alarm is triggered when the HTIMER_SEC.rts[19:0] matches HTIMER_RAS.ras. Hardware will then set the HTIMER_RAS.tod_fl bit and an interrupt will be generated if software has set HTIMER_RAS.tod_en.

You must disable the long-interval alarm before changing the HTIMER_RAS.tod field.

15.2.2 Short-Interval Alarm

The HTIMER_RSSA.rts and HTIMER_CTRL.alarm_ss_en fields control the short-interval alarm. Writing HTIMER_RSSA sets the starting value for the short-interval alarm counter. Writing the Short-Interval Alarm Enable (HTIMER_CTRL.alarm_ss_en) bit to 1 enables the short-interval alarm. Once enabled, the short-interval alarm begins up-counting from the HTIMER_RSSA value. When the counter rolls over from 0xFFFF FFFF to 0x0000 0000, hardware sets the HTIMER_CTRL.alarm_ss_fl bit triggering the alarm. At the same time, hardware also reloads the counter with the value previously written to HTIMER_RSSA.rssa.

You must disable the short-interval interval alarm, HTIMER_CTRL.alarm_ss_en, prior to changing the interval alarm value, HTIMER_RSSA.

The delay (uncertainty) associated with enabling the short-interval alarm is propagated to the first interval alarm. Thereafter, if the interval alarm remains enabled, the alarm triggers after each short-interval interval as defined without the first alarm uncertainty because the short-interval alarm is an auto-reload timer. Enabling the short-interval alarm with the short-interval alarm register set to 0 (HTIMER_RSSA.rssa = 0) results in the maximum short-interval alarm interval.
15.3 Register Access Control

The hardware provides a collision-protection mechanism that prevents software from reading registers at the same time they are being updated by hardware, and vice versa.

15.3.1 Register Write Protection

The HTIMER_CTRL.busy bit is a read-only status bit controlled by hardware and set when any of the following conditions occur:

- System Reset.
- Software writes to the HTIMER_SEC.rts.
- Software modifies the HTIMER_CTRL.enable, HTIMER_CTRL.alarm_tod_en, or HTIMER_CTRL.alarm_ss_en bits.

When the HTIMER_CTRL.busy bit is set by hardware, writes to the above bits and count registers are blocked by hardware. The HTIMER_CTRL.busy bit remains active until the register or bit is synchronized by hardware. The synchronization by hardware occurs on the next timer tick. The HTIMER_CTRL.busy bit is set for a maximum of one timer tick. Therefore, a software write is not complete until hardware clears the HTIMER_CTRL.busy bit.

Once the HTIMER_CTRL.busy bit is cleared to 0, additional writes are completed as permitted by individual count or alarm-enable bits.

15.3.2 Register Read Protection

The HTIMER_CTRL.ready bit indicates when the count registers contain valid data. Hardware clears the HTIMER_CTRL.ready bit approximately one timer tick before the ripple occurs through the counter registers (HTIMER_SEC.rts and HTIMER_SSEC) and is set again immediately after the ripple occurs. The period of the HTIMER_CTRL.ready bit set/clear activity provides a large window during which the counter registers are readable. Software can clear the HTIMER_CTRL.ready bit at any time and the bit remains clear until set by hardware when the next ripple occurs. A separate Ready Enable (HTIMER_CTRL.ready_int_en) bit is provided to generate an interrupt when hardware sets the HTIMER_CTRL.ready bit. You can use this interrupt to signal the start of a new timer read window.

15.3.3 Count Register Access

Values read from the count registers (HTIMER_SEC.rts and HTIMER_SSEC) are valid only when the HTIMER_CTRL.ready = 1.

To write the count registers, disable the timer by clearing (HTIMER_CTRL.enable) to 0. Clearing the HTIMER_CTRL.enable bit is permitted only when the Write Enable (HTIMER_CTRL.write_en) bit is set to 1 and is governed by the HTIMER_CTRL.busy bit signaling process (that is, the HTIMER_CTRL.busy bit is 0). Writes to each count register must occur only when the HTIMER_CTRL.busy bit reads 0.

15.3.4 Alarm Register Access

The alarm registers HTIMER_RAS and HTIMER_RSSA are readable at any time.

Set HTIMER_CTRL.alarm_ss_en = 0 before writing to HTIMER_RSSA.rssa. Set HTIMER_CTRL.alarm_tod_en = 0 before writing to HTIMER_RAS.

Clearing these bits requires monitoring the HTIMER_CTRL.busy bit to assess completion of the write. Once the alarm is disabled, update the associated alarm registers using software.
15.4 Registers

See Table 3-1: APB Peripheral Base Address Map for the HTimer Register Peripheral Base Addresses. All fields are reset on peripheral, system, or power-on reset events unless otherwise specified.

Table 15-1. HTimer Registers Summary

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>HTIMER_SEC</td>
<td>HTimer Long-Interval Counter Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>HTIMER_SSEC</td>
<td>HTimer Short-Interval Counter Register</td>
</tr>
<tr>
<td>[0x0008]</td>
<td>HTIMER_RAS</td>
<td>HTimer Long-Interval Alarm Register</td>
</tr>
<tr>
<td>[0x000C]</td>
<td>HTIMER_RSSA</td>
<td>HTimer Short-Interval Alarm Register</td>
</tr>
<tr>
<td>[0x0010]</td>
<td>HTIMER_CTRL</td>
<td>HTimer Control Register</td>
</tr>
</tbody>
</table>

15.5 Register Details

Table 15-2: HTimer Long-Interval Counter Register

<table>
<thead>
<tr>
<th>HTimer Long-Interval Counter</th>
<th>HTIMER_SEC</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td>31:0</td>
<td>rts</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 15-3: HTimer Short-Interval Counter Register

<table>
<thead>
<tr>
<th>HTimer Short-Interval Counter</th>
<th>HTIMER_SSEC</th>
<th>[0x0004]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td>31:8</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td>7:0</td>
<td>rtss</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 15-4: HTimer Long-Interval Alarm Register

<table>
<thead>
<tr>
<th>HTimer Long-Interval Alarm</th>
<th>HTIMER_RAS</th>
<th>[0x0008]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td>31:20</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td>19:0</td>
<td>ras</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 15-5: HTimer Short-Interval Alarm Register

<table>
<thead>
<tr>
<th>HTimer Short-Interval Alarm</th>
<th>HTIMER_RSSA</th>
<th>[0x000C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td>31:0</td>
<td>rssa</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 15-6: HTimer Control Register

<table>
<thead>
<tr>
<th>Bit Range</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use. Do not modify this field from its default value.</td>
</tr>
<tr>
<td>15</td>
<td>write_en</td>
<td>R/W</td>
<td>0</td>
<td>Write Enable. Software must set this bit to 1 before writing to HTIMER_CTRL.enable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Writes to HTIMER_CTRL.enable are allowed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Writes to HTIMER_CTRL.enable are ignored.</td>
</tr>
<tr>
<td>14:8</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use. Do not modify this field from its default value.</td>
</tr>
<tr>
<td>7</td>
<td>alarm_ss_en</td>
<td>R/W</td>
<td>0</td>
<td>Short-Interval Alarm Interrupt Flag. This flag is a wake-up source for the processor.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No short-interval alarm pending.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Short-interval interrupt pending.</td>
</tr>
<tr>
<td>6</td>
<td>alarm_tod_en</td>
<td>R/W</td>
<td>0</td>
<td>Long-Interval Alarm Interrupt Flag. This flag is a wake-up source for the processor.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No long-interval alarm interrupt pending.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Long-interval interrupt pending.</td>
</tr>
<tr>
<td>5</td>
<td>ready_int_en</td>
<td>R/W</td>
<td>0</td>
<td>Timer Ready Interrupt Enable. This interrupt flag is set when the timer ready bit is set by hardware.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled.</td>
</tr>
<tr>
<td>4</td>
<td>ready</td>
<td>R/WOO</td>
<td>0</td>
<td>Timer Ready. This bit is set to 1 by hardware when HTIMER_SEC.rts is updated. Software can clear this bit at any time. Hardware automatically clears this bit just prior to updating the HTIMER_SEC.rts, indicating the timer is busy.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: HTIMER_SEC register not updated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: HTIMER_SEC register updated.</td>
</tr>
<tr>
<td>3</td>
<td>busy</td>
<td>RO</td>
<td>0</td>
<td>Timer Busy Flag. This bit is set by hardware when changes to the registers are synchronized. The bit is automatically cleared by hardware when the synchronization is complete. Software should poll this field for 0 after changing registers to ensure the change is complete prior to making any other register changes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Not busy.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Busy.</td>
</tr>
<tr>
<td>2</td>
<td>alarm_ss_en</td>
<td>R/W</td>
<td>0</td>
<td>Short-Interval Alarm Interrupt Enable. Set this bit to 1 to enable the short-interval alarm interrupt. Check the HTIMER_CTRL.busy flag after writing this bit to determine when the register synchronizations are complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Short-interval alarm interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Short-interval alarm interrupt disabled.</td>
</tr>
<tr>
<td>1</td>
<td>alarm_tod_en</td>
<td>R/W</td>
<td>0</td>
<td>Long-Interval Alarm Interrupt Enable. Set this bit to 1 to enable the long-interval alarm interrupt. Check the HTIMER_CTRL.busy flag after writing to this bit to determine when the timer synchronization is complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Long-interval alarm interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Long-interval alarm interrupt is enabled.</td>
</tr>
</tbody>
</table>
### HTimer Control

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>enable</td>
<td>R/W</td>
<td>0</td>
<td>HT Enable</td>
</tr>
</tbody>
</table>

Enables and disables the timer. Software must write `HTIMER_CTRL.write_en = 1` before changing this field. `HTIMER_CTRL.busy` must read 0 before writing to this bit. After writing to this bit, check the `HTIMER_CTRL.busy` flag for 0 to determine when the HT synchronization is complete.

- 0: HT disabled.
- 1: HT enabled.
16. Timers

Multiple 32-bit, reloadable timers are provided. Each timer provides multiple operating modes:

- One-Shot: Timer counts up to terminal value then halts.
- Continuous: Timer counts up to terminal value then repeats.
- Counter: Timer counts input edges received on timer input pin.
- Pulse Width Modulated (PWM) / PWM Differential.
- Capture: Captures a snapshot of the current timer count when timer input edge transitions.
- Compare: Timer pin toggles when timer exceeds terminal count.
- Gated: Timer increments only when timer input pin is asserted.
- Capture/Compare: Timer counts when timer input is asserted, captures timer count when input is deasserted.

The MAX32665–MAX32668 provide six instances of the timer peripheral (TMR0, TMR1, TMR2, TMR3, TMR3, TMR5).

16.1 Features

- 32-bit reload counter
- Programmable prescaler with values from 1 to 4096
- Non-overlapping PWM output generation with configurable off-time
- Capture, compare, and capture/compare capability
- Timer pin available as alternate function
- Configurable Input pin for event triggering, clock gating, or capture signal
- Timer output pin for event output and PWM signal generation
- Independent interrupt

16.2 Basic Operation

The timer modes operate by incrementing the TMRn_CNT register, driven by either the timer clock, an external stimulus on the timer pin, or a combination of both. The TMRn_CNT register is always readable, even while the timer is enabled and counting.

Each timer mode has a user-configurable timer period, which terminates on the timer clock cycle following the end of timer period condition. Each timer mode has a different response at the end of a timer period, which can include changing the state of the timer pin, capturing a timer value, reloading TMRn_CNT with a new starting value, or disabling the counter. The end of a timer period will always set the corresponding interrupt bit and can generate an interrupt, if enabled.

In most modes the timer peripheral automatically sets TMRn_CNT to 0x0000 0001 at the end of a timer period, but TMRn_CNT is set to 0x0000 0000 following a system reset. This means the first timer period following a system reset will be one timer clock longer than subsequent timer periods if TMRn_CNT is not initialized to 0x0000 0001 during the timer configuration step.

Clocking of timer functions is driven by the timer clock frequency, f_CNT_CLK. The timer clock frequency is a user-configurable, division of the system peripheral clock, PCLK. Each timer has an independent prescaler, allowing timers to operate at different frequencies. The prescaler can be set from 1 to 4096 using the TMRn_CN.pres3: TMRn_CN.pres fields. Unless otherwise mentioned, the timer clock is generated as follows:

\[
E_{\text{q16-1}}: \text{Timer Peripheral Clock Equation} \\
E_{\text{q16-1}}: f_{\text{CNT_CLK}} = \frac{f_{\text{PCLK}}}{\text{prescaler}}
\]

Application firmware writes to the timer registers and external events on timer pins will be asynchronous events to the slower timer clock frequency. These events are latched on the next rising edge of the timer clock. Since it is not possible to observe the timer clock directly, input events may have up to 0.5 timer clock delay before being recognized.
16.3 Timer Pin Functionality

Most timers have an associated timer pin that can function as an optional input or output depending on the selected timer mode. The timer pin functionality is mapped as an alternate function that is shared with a GPIO. Timer pin assignments are detailed in the data sheet for the specific device.

When the timer pin alternate function is enabled, the timer pin will have the same electrical characteristics, such as pullup/pulldown strength, drive strength, etc. as the GPIO mode settings for that pin. 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. The pin characteristics must be configured before enabling the timer. Consult the GPIO section for details on how to configure the electrical characteristics for the pin.

Each timer has a dedicated interrupt flag, TMRn_INT.irq, which is set at the end of a timer period. If enabled, an interrupt will be generated. The interrupt flag can be cleared by writing any value to TMRn_INT.irq.

16.4 One-Shot Mode (000b)

In One-shot mode the timer peripheral increments TMRn_CNT until it matches TMRn_CMP and then stops incrementing and disables the timer. The timer can optionally output a pulse on the timer pin at the end of the timer period. In this mode, the timer must be re-enabled to start another one-shot mode event.

**Figure 16-1: One-Shot Mode Diagram**

* TMR_CNT AUTOmatically RELOADS WITH 0X0000_0001 AT THE END OF THE TIMER PERIOD, BUT SOFTWARE CAN WRITE ANY INITIAL VALUE TO TMR_CNT BEFORE THE TIMER IS ENABLED.
** THe DEFAULT VALUE OF TMR_CNT FOR THE FIRST PERIOD AFTER A SYSTEM RESET IS 0X0000_0000 UNLESS CHANGED BY SOFTWARE.
16.4.1 One-Shot Mode Timer Period

The timer period ends on the timer clock following $TMRn_CNT = TMRn_CMP$.

The timer peripheral automatically performs the following actions at the end of the timer period:

1. $TMRn_CNT$ is reset to 0x0000 0001.
2. The timer is disabled by setting $TMRn_CN.ten = 0$.
3. If the timer output is enabled, the timer pin is driven to its active state for one timer clock. It then returns to its inactive state.
4. The timer interrupt bit $TMRn_INT.irq$ will be set. An interrupt will be generated if enabled.

16.4.2 One-Shot Mode Configuration

Configure the timer for One-Shot mode by doing the following:

1. Set $TMRn_CN.ten = 0$ to disable the timer. Set $TMRn_CN.tmode$ to 000b to select One-shot mode.
2. Set $TMRn_CN.pres3:TMRn_CN.pres$ to set the prescaler that determines the timer frequency.
3. If using the timer pin:
   a. Configure the pin as a timer output and configure the electrical characteristics as needed.
   b. Set $TMRn_CN.tpol$ to match the desired (inactive) state.
4. If using the timer interrupt, enable the interrupt and set the interrupt priority.
5. Write an initial value to $TMRn_CNT$, if desired. This effects only the first period; subsequent timer periods always reset $TMRn_CNT = 0x0000 0001$.
6. Write the compare value to $TMRn_CMP$.
7. Set $TMRn_CN.ten = 1$ to enable the timer.

The timer period is calculated using the following equation:

Equation 16-2: One-shot Mode Timer Period

$$\text{One-shot mode timer period in seconds} = \frac{TMRn_CMP - TMRn_CNT_{\text{INITIAL VALUE}} + 1}{f_{CNT\_CLK} \ (Hz)}$$

16.5 Continuous Mode (001b)

In Continuous mode, the timer peripheral increments $TMRn_CNT$ until it matches $TMRn_CMP$, resets $TMRn_CNT$ to 0x0000 0001, and continues incrementing. The timer peripheral can optionally toggle the state of the timer pin at the end of the timer period.
Figure 16-2: Continuous Mode Diagram

* TMR_CNT automatically reloads with 0x0000_0001 at the end of the timer period, but software can write any initial value to TMR_CNT before the timer is enabled.
** The default value of TMR_CNT for the first period after a system reset is 0x0000_0000 unless changed by software.

16.5.1 Continuous Mode Timer Period

The timer period ends on the timer clock following \( TMR_n.CNT = TMR_n.CMP \).
The timer peripheral automatically performs the following actions at the end of the timer period:

1. \( TMR_n_CNT \) is reset to 0x0000 0001. The timer remains enabled and continues incrementing.
2. If the timer output is enabled, the timer pin toggles state (low to high or high to low).
3. The timer interrupt bit \( TMRn_INT.irq \) will be set. An interrupt will be generated if enabled.

### 16.5.2 Continuous Mode Configuration

Configure the timer for Continuous mode by performing the steps following:

1. Set \( TMRn_CN.ten \) = 0 to disable the timer.
2. Set \( TMRn_CN.tmode \) to 001b to select Continuous mode.
3. Set \( TMRn_CN.pres3:TMRx_CN.pres \) to set the prescaler that determines the timer frequency, \( f_{CNT,CLK} \).
4. If using the timer pin:
   a. Configure the pin as a timer output and configure the electrical characteristics as needed.
   b. Set \( TMRn_CN.tpol \) to match the desired inactive state.
5. If using the timer interrupt, enable the interrupt and set the interrupt priority.
6. Write an initial value to \( TMRn_CNT \), if desired. The initial value is only used for the first period; subsequent timer periods always reset the \( TMRn_CNT \) register to 1.
7. Write the compare value to \( TMRn_CMP \).
8. Set \( TMRn_CN.ten \) to 1 to enable the timer.

The Continuous Mode Timer Period is calculated using \( \text{Equation 16-3} \).

### Equation 16-3: Continuous Mode Timer Period

Continuous mode timer period in seconds = \( \frac{TMR_{CMP} - TMR_{CNT,INITIAL,VALUE} + 1}{f_{CNT,CLK} \text{ (Hz)}} \)
16.6 Counter Mode (010b)

In Counter mode, the timer peripheral increments TMRn_CNT when a transition occurs on the timer pin. When TMRn_CNT = TMRn_CMP, the interrupt bit is set and the TMRn_CNT register is set to 0x0000 0001 and continues incrementing. The timer can be configured to increment on either the rising edge or the falling edge, but not both.

The timer prescaler setting has no effect in this mode. The frequency of the timer’s input signal (f_{CTR_CLK}) must not exceed 25 percent of the PCLK frequency as shown in the following equation:

Equation 16-4: Counter Mode Maximum Clock Frequency

\[ f_{CTR,CLK} \leq \frac{f_{PCLK} (Hz)}{4} \]

Figure 16-3: Counter Mode Diagram

* TMR_CNT is reloaded with 0x0000 0001 at the end of each timer period. Firmware sets the initial value in TMR_CNT prior to enabling the timer.
16.6.1 Counter Mode Timer Period

The timer period ends on the rising edge of PCLK following \( TMRn\_CNT = TMRn\_CMP \).

The timer peripheral automatically performs the following actions at the end of the timer period:

1. \( TMRn\_CNT \) is reset to 0x0000 0001. The timer remains enabled and continues incrementing on selected transitions of the timer pin.
2. The timer interrupt bit \( TMRn\_INT.irq \) will be set. An interrupt will be generated if enabled.

16.6.2 Counter Mode Configuration

Configure the timer for Counter mode by doing the following:

1. Set \( TMRn\_CN.ten = 0 \) to disable the timer.
2. Set \( TMRn\_CN.tmode \) to 010b to select Counter mode.
3. Configure the timer pin:
   a. Configure the pin as a timer input and configure the electrical characteristics as needed.
   b. Set \( TMRn\_CN.tpol \) to match the desired initial (inactive) state.
4. If using the timer interrupt, enable the interrupt and set the interrupt priority.
5. Write an initial value to \( TMRn\_CNT \), if desired. The initial value is only used for the first timer period; subsequent timer periods always reset \( TMRn\_CNT \) to 1.
6. Write the compare value to \( TMRn\_CMP \).
7. Set \( TMRn\_CN.ten = 1 \) to enable the timer.

In Counter mode, the number of timer input transitions since timer start is calculated using the following equation:

\[
\text{Counter mode timer input transitions} = TMR\_CNT_{\text{CURRENT}} \_\text{VALUE} - TMR\_CNT_{\text{INITIAL}} \_\text{VALUE}
\]
16.7 PWM Mode (011b)

In PWM mode, the timer sends a Pulse-Width Modulated (PWM) output using the timer’s output signal. The timer first counts up to the match value stored in the TMRn_PWM register. At the end of the cycle where the TMRn_CNT value matches the TMRn_PWM value, the timer’s output toggles state. The timer continues counting until it reaches the TMRn_CMP value.

16.7.1 PWM Mode Timer Period

The timer period ends on the rising edge of PCLK following TMRn_CNT = TMRn_CMP.

The timer peripheral automatically performs the following actions at the end of the timer period:

1. The TMRn_CNT is reset to 0x0000 0001, and the timer resumes counting.
2. The timer output signal is toggled.
3. The timer interrupt bit TMRn_INT.irq will be set. An interrupt will be generated if enabled.

When TMRn_CN.tpol = 0, the timer output signal starts low and then transitions to high when the TMRn_CNT value matches the TMRn_PWM value. The timer output signal remains high until the TMRn_CNT value reaches the TMRn_CMP value, resulting in the timer output signal transitioning low, and the TMRn_CNT value resetting to 0x0000 0001.

When TMRn_CN.tpol = 1, the timer output signal starts high and transitions low when the TMRn_CNT value matches the TMRn_PWM value. The timer output signal remains low until the TMRn_CNT value reaches the TMRn_CMP value, resulting in the timer output signal transitioning high, and the TMRn_CNT value resetting to 0x0000 0001.

16.7.2 PWM Mode Configuration

Complete the following steps to configure a timer for PWM mode and initiate the PWM operation:

1. Set TMRn_CN.ten = 0 to disable the timer.
2. Set TMRn_CN.tmode to 011b to select PWM mode.
3. Set TMRn_CN.pres3:TMn_CN.pres to set the prescaler that determines the timer frequency.
4. Configure the timer pin:
5. Configure the pin as a timer input and configure the electrical characteristics as needed.
6. Set TMRn_CN.tpol to match the desired initial (inactive) state.
   a. Set TMRn_CN.tpol to select the initial logic level (high or low) and PWM transition state for the timer’s output.
   b. Set TMRn_CNT to the starting count, typically 0x0000 0001. The initial TMRn_CNT value only effects the initial period in PWM mode with subsequent periods always setting TMRn_CNT to 0x0000 0001.
   c. Set the TMRn_PWM value to the transition period count.
7. Set the TMRn_CMP value for the PWM second transition period. Note: TMRn_CMP must be greater than the TMRn_PWM value.
8. Optionally enable the timer’s interrupt in the Interrupt Controller, and set the timer’s interrupt priority.
9. Set TMRn_CN.ten to 1 to enable the timer and start the PWM.

The PWM period is calculated using the following equation:

**Equation 16-6: Timer PWM Period**

\[
\text{PWM period in seconds} = \frac{\text{TMR_CNT}}{\text{f_{CNT,CLK}} \text{ (Hz)}}
\]

If an initial starting value other than 0x0000 0001 is loaded into the TMRn_CNT register, use the One-Shot mode equation, **Equation 16-2**, to determine the initial PWM period.
If $TMRn_{\_CN}.tpol$ is 0, the ratio of the PWM output high time to the total period is calculated using Equation 16-7, below.

Equation 16-7: Timer PWM Output High Time Ratio with Polarity 0

$$\text{PWM output high time ratio (\%)} = \frac{(TMR_{\_CMP} - TMR_{\_PWM})}{TMR_{\_CMP}} \times 100$$

If $TMRn_{\_CN}.tpol$ is set to 1, the ratio of the PWM output high time to the total period is calculated using Equation 16-8.

Equation 16-8: Timer PWM Output High Time Ratio with Polarity 1

$$\text{PWM output high time ratio (\%)} = \frac{TMR_{\_PWM}}{TMR_{\_CMP}} \times 100$$
16.8 Capture Mode (100b)

Capture mode most often used to measure the time between events. The timer increments from an initial value until an edge transition occurs on the timer pin. This triggers the ‘capture’ event which copies TMRn_CNT to the TMRn_PWM.pwm register, resets TMRn_CNT = 0x0000 0001, and continues incrementing. Also, if the timer pin does not go active before TMRn_CNT = TMRn_CMP, the timer will reset TMRn_CNT = 0x0000 0001, and continue incrementing. Either event will set the timer interrupt bit.

*Figure 16-4: Capture Mode Diagram*

* TMR_CNT AUTOMATICALLY RELOADS WITH 0X0000_0001 AFTER A CAPTURE EVENT OR WHEN TMR_CNT = TMR_CMP, BUT SOFTWARE CAN WRITE ANY INITIAL VALUE TO TMR_CNT BEFORE THE TIMER IS ENABLED.
** THE DEFAULT VALUE OF TMR_CN FOR THE FIRST PERIOD AFTER A SYSTEM RESET IS 0X0000_0000 UNLESS CHANGED BY SOFTWARE.
16.8.1 Capture Mode Timer Period

Two timer period events are possible in Capture Mode:

The Capture event occurs on the timer clock following the selected transition on the timer pin. The timer peripheral automatically performs the following actions:

1. The value in \( TMRn\_CNT \) is copied to \( TMRn\_PWM \)
2. The timer interrupt bit \( TMRn\_INT\_irq \) will be set. An interrupt will be generated if enabled.
3. The timer remains enabled and continues incrementing.
4. The timer period ends on the timer clock following \( TMRn\_CNT = TMRn\_CMP \).

The timer period event occurs on the timer clock \( TMRn\_CNT = TMRn\_CMP \). The timer peripheral automatically performs the following actions when an end of timer period event occurs:

1. The value in \( TMRn\_CNT \) is reset to 0x0000 00001. The timer remains enabled and continues incrementing.
2. The timer interrupt bit \( TMRn\_INT\_irq \) will be set. An interrupt will be generated if enabled.

16.8.2 Capture Mode Configuration

Configure the timer for Capture mode by doing the following:

1. Disable the timer by setting \( TMRn\_CN\_ten \) to 0.
2. Select Counter mode by setting \( TMRn\_CN\_tmode \) to 010b.
3. Set \( TMRn\_CN\_pres3:TMRn\_CN\_pres \) to set the prescaler that determines the timer frequency.
4. If using the timer pin:
   a. Configure the pin as a timer output and configure the electrical characteristics as needed.
   b. Set \( TMRn\_CN\_tpol \) to match the desired (inactive) state.
5. If using the timer interrupt, enable the interrupt and set the interrupt priority.
6. Write the initial value to \( TMRn\_CNT \). This effects only the first period; subsequent periods always begin with 0x0000 0001.
7. Write the compare value to \( TMRn\_CMP \).
8. Set \( TMRn\_CN\_ten = 1 \) to enable the timer.

The timer period is calculated using the following equation:

\[
\text{Equation 16-9: Capture Mode Elapsed Time Calculation in Seconds}
\]

\[
\text{Capture elapsed time in seconds} = \frac{TMR\_PWM - TMR\_CNT_{INITIAL\_VALUE}}{f_{CNT\_CLK}}
\]

Note: The capture elapsed time calculation is only valid after the capture event occurs, and the timer stores the captured count in the TMRn_PWM register.

16.9 Compare Mode (101b)

In Compare mode the timer peripheral increments continually, allowing the timer to be a programmable 32-bit programmable period timer. 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.
**16.9.1 Compare Mode Timer Period**

The timer period ends on the timer clock following \( TMRn\_CNT = TMRn\_CMP \).

The timer peripheral automatically performs the following actions at the end of the timer period:

1. The timer remains enabled and continues incrementing. Unlike other modes, \( TMRn\_CNT \) is not reset to 0x0000 0001 at the end of the timer period.
2. If the timer output is enabled, then the timer pin toggles state (low to high or high to low).
3. The timer interrupt bit \( TMRn\_INT.irq \) will be set. An interrupt is generated if enabled.
### 16.9.2 Compare Mode Configuration

Configure the timer for Compare mode by doing the following:

1. Set `TMRn_CN.ten = 0` to disable the timer.
2. Set `TMRn_CN.tmode` to 011b to select Compare mode.
3. Set `TMRn_CN.pres3:TMRn_CN.pres` to set the prescaler that determines the timer frequency.
4. If using the timer pin:
   a. Configure the pin for the timer output alternate function and configure the electrical characteristics as needed.
   b. Set `TMRn_CN.tpol` to match the desired (inactive) state.
5. If using the timer interrupt, enable the interrupt and set the interrupt priority.
6. Write the initial value to `TMRn_CNT`. This effects only the first period as the counter increments continuously, rolling over to 0x0000 0000 and continuing.
7. Write the compare value to `TMRn_CMP`.
8. Set `TMRn_CN.ten = 1` to enable the timer.

The Compare Mode timer period is calculated using *Equation 16-10*.

*Equation 16-10: Compare Mode Timer Period*

\[
\text{Compare mode timer period in seconds} = \frac{TMR_{\text{CMP}} - TMR_{\text{CNT\_INITIAL\_VALUE}} + 1}{f_{\text{CNT\_CLK}} \text{ (Hz)}}
\]
16.10 Gated Mode (110b)

Gated mode is similar to continuous mode, except that $TMRn\_CNT$ only increments when the timer pin is in its active state.

Figure 16-6: Gated Mode Diagram

* $TMR\_CNT$ automatically reloads with 0X0000_0001 at the end of the timer period, but software can write any initial value to $TMR\_CNT$ before the timer is enabled.

** The default value of $TMR\_CNT$ for the first period after a system reset is 0X0000_0000 unless changed by software.
16.10.1 Gated Mode Timer Period

The timer period ends when $\text{TMR}_n\_\text{CNT} = \text{TMR}_n\_\text{CMP}$ and the timer automatically performs the following actions:

1. $\text{TMR}_n\_\text{CNT}$ is reset to 0x0000 0001. The timer remains enabled and continues incrementing.
2. The timer interrupt bit $\text{TMR}_n\_\text{INT}.\text{irq}$ will be set. An interrupt will be generated if enabled.

16.10.2 Gated Mode Configuration

Configure the timer for Gated mode by doing the following:

1. Set $\text{TMR}_n\_\text{CN}.\text{ten} = 0$ to disable the timer.
2. Set $\text{TMR}_n\_\text{CN}.\text{tmode}$ to 110b to select Gated mode.
3. Set $\text{TMR}_n\_\text{CN}.\text{pres3:TM}\text{RN}_n\_\text{CN}.\text{pres}$ to set the prescaler that determines the timer frequency.
4. Configure the timer pin:
   a. Configure the pin as a timer input and configure the electrical characteristics as needed.
   b. Set $\text{TMR}_n\_\text{CN}.\text{tpol}$ to match the desired initial (inactive) state.
5. If using the timer interrupt, enable the interrupt and set the interrupt priority.
6. Write an initial value to $\text{TMR}_n\_\text{CNT}$, if desired. This effects only the first period; subsequent timer periods always reset $\text{TMR}_n\_\text{CNT} = 0x0000 0001$.
7. Write the compare value to $\text{TMR}_n\_\text{CMP}$.
8. Set $\text{TMR}_n\_\text{CN}.\text{ten} = 1$ to enable the timer.
16.11 Capture/Compare Mode (111b)

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 \texttt{TMRn\_CN\_tpol} bit.

Each subsequent transition, after the first transition of the timer input signal, captures the \texttt{TMRn\_CNT} value, writing it to the \texttt{TMRn\_PWM} register (capture event). When a capture event occurs, a timer interrupt is generated, the \texttt{TMRn\_CNT} value is reset to 0x0000 0001, and the timer resumes counting.

If no capture event occurs, the timer counts up to the \texttt{TMRn\_CMP} value. At the end of the cycle where the \texttt{TMRn\_CNT} equals the \texttt{TMRn\_CMP} value, a timer interrupt is generated, the \texttt{TMRn\_CNT} value is reset to 0x0000 0001, and the timer resumes counting.

16.11.1 Capture/Compare Timer Period

The timer period ends when the selected transition occurs on the timer pin, or on the clock cycle following $\texttt{TMRn\_CNT} = \texttt{TMRn\_CMP}$.

The timer peripheral automatically performs the following actions at the end of the timer period:

If the end of the timer period was caused by a transition on the timer pin:

1. The value in \texttt{TMRn\_CNT} is copied to \texttt{TMRn\_PWM}.
2. \texttt{TMRn\_CNT} is reset to 0x0000 0001. The timer remains enabled and continues incrementing.
3. The timer interrupt bit, \texttt{TMRn\_INT\_irq}, is set. If the timer’s interrupt is enabled a Timer IRQ is generated automatically.
4. If the end of the timer period was caused by a transition on the timer pin:
5. \texttt{TMRn\_CNT} is reset to 0x0000 0001. The timer remains enabled and continues incrementing.
6. The timer interrupt bit \texttt{TMRn\_INT\_irq} will be set. An interrupt will generated if enabled.

16.11.2 Capture/Compare Configuration

Configure the timer for Capture/Compare mode by doing the following:

1. Set \texttt{TMRn\_CN\_ten} = 0 to disable the timer.
2. Set \texttt{TMRn\_CN\_tmode} to 111b to select Capture/Compare mode.
3. Set \texttt{TMRn\_CN\_pres3:TMRn\_CN\_pres} to set the prescaler that determines the timer frequency.
4. Configure the timer pin:
   a. Configure the pin as a timer input and configure the electrical characteristics as needed.
   b. Set \texttt{TMRn\_CN\_tpol} to select the positive edge \texttt{(TMRn\_CN\_tpol = 0)} or negative edge \texttt{(TMRn\_CN\_tpol = 0)} transition causes the capture event.
5. If using the timer interrupt, enable the interrupt and set the interrupt priority.
6. Write an initial value to \texttt{TMRn\_CNT}, if desired. This effects only the first period; subsequent timer periods always reset \texttt{TMRn\_CNT} = 0x0000 0001.
7. Set \texttt{TMRn\_CN\_ten} to 1 to enable the timer. Counting starts after the first transition of the timer’s input signal.

Note: No interrupt is generated by the first transition of the input signal.

In Capture/Compare mode, the elapsed time from the timer start to the capture event is calculated using \textit{Equation 16-11}, below.
Equation 16-11: Capture Mode Elapsed Time

\[ \text{Capture elapsed time in seconds} = \frac{T_{\text{MR_PWM}} - T_{\text{MR_CNT INITIAL_CNT_VALUE}}}{f_{\text{CNT_CLK}} \text{ (Hz)}} \]

16.12 Timer Registers

Address offsets for the timer registers are shown in Table 16-1. Register fields marked as Reserved for Future Use should not be modified. All Timer instances contain an identical set of registers. Register names for a specific instance are defined by appending the instance number to the peripheral name. For example, the Timer Count Register for Timer 0 is TMR0_CNT while the Timer Count Register for Timer 1 is TMR1_CNT, etc.

See Table 3-1: APB Peripheral Base Address Map for the Timer 0 (TMR0_) to Timer 5 (TMR5_) Peripheral Base Addresses.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register Name</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>TMRn_CNT</td>
<td>R/W</td>
<td>Timer Counter Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>TMRn_CMP</td>
<td>R/W</td>
<td>Timer Compare Register</td>
</tr>
<tr>
<td>[0x0008]</td>
<td>TMRn_PWM</td>
<td>R/W</td>
<td>Timer PWM Register</td>
</tr>
<tr>
<td>[0x000C]</td>
<td>TMRn_INT</td>
<td>R/W</td>
<td>Timer Interrupt Register</td>
</tr>
<tr>
<td>[0x0010]</td>
<td>TMRn_CN</td>
<td>R/W</td>
<td>Timer Control Register</td>
</tr>
<tr>
<td>[0x0014]</td>
<td>TMRn_NOLCMP</td>
<td>R/W</td>
<td>Timer Non-Overlapping Compare Register</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0 | count | R/W | 0 | Timer Count Value
The current count value for the timer. This field increments as the timer counts. Reads of this register are always valid. Prior to writing this field, disable the timer by clearing bit TMRn_CN.ten. |

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0 | compare | R/W | 0 | Timer Compare Value
The value in this register is used as the compare value for the timer’s count value. The compare field meaning is determined by the specific mode of the timer. See the timer mode’s detailed configuration section for compare usage and meaning. |

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0 | pwm | R/W | 0 | Timer PWM Match
In PWM mode, this field sets the count value for the first transition period of the PWM cycle. At the end of the cycle where TMRn_CNT equals TMRn_CMP, the PWM output transitions to the second period of the PWM cycle. The second PWM period count is stored in the TMRn_CMP register. The value set for TMRn_PWM.pwm must be less than the value set in TMRn_CMP for PWM mode operation. |

Table 16-3: Timer Capture Value

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0 | pwm | R/W | 0 | Timer Capture Value
In Capture, Compare, and Capture/Compare modes, this field is used to store the TMRn_CNT value when a Capture, Compare, or Capture/Compare event occurs. |
### Table 16-4: Timer Interrupt Registers

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:1</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field from its default value.</td>
</tr>
<tr>
<td>0</td>
<td>irq</td>
<td>RW</td>
<td>0</td>
<td>Timer Interrupt</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If set, this field indicates a timer interrupt condition occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Writing any value to this bit clears the timer’s interrupt.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Timer interrupt is not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Timer interrupt occurred.</td>
</tr>
</tbody>
</table>

### Table 16-5: Timer Control Registers

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:13</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field from its default value.</td>
</tr>
<tr>
<td>12</td>
<td>pwmckbd</td>
<td>R/W</td>
<td>1</td>
<td>PWM Output $\phi_A'$ Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Disable PWM Output $\phi_A'$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Enable PWM Output $\phi_A'$</td>
</tr>
<tr>
<td>11</td>
<td>nollpol</td>
<td>R/W</td>
<td>0</td>
<td>PWM Output $\phi_A'$ Polarity Bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Output $\phi_A'$ inverted</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Output $\phi_A'$ non-inverted</td>
</tr>
<tr>
<td>10</td>
<td>nolhpol</td>
<td>R/W</td>
<td>0</td>
<td>PWM Output $\phi_A$ Polarity Bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Output $\phi_A$ inverted</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Output $\phi_A$ non-inverted</td>
</tr>
<tr>
<td>9</td>
<td>pwmsync</td>
<td>R/W</td>
<td>0</td>
<td>PWM Synchronization Mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: PWM synchronization mode enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: PWM synchronization mode disabled</td>
</tr>
<tr>
<td>8</td>
<td>pres3</td>
<td>R/W</td>
<td>0</td>
<td>Timer Prescale Select MSB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>See <a href="#">TMRn_CN_pres</a> for details on this field’s usage.</td>
</tr>
<tr>
<td>7</td>
<td>ten</td>
<td>R/W</td>
<td>0</td>
<td>Timer Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Timer enabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Timer disabled</td>
</tr>
<tr>
<td>6</td>
<td>tpol</td>
<td>R/W</td>
<td>0</td>
<td>Timer Polarity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Selects the polarity of the timer’s input and output signal.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This setting is not used if the GPIO is not configured for the alternate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>function. The tpol field meaning is determined by the specific mode of the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>timer. See the mode’s detailed configuration section for tpol usage.</td>
</tr>
</tbody>
</table>
### Timer Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 5:3  | pres  | R/W    | 0     | Timer Prescaler Select  
Sets the timer’s prescaler value. The prescaler divides the PCLK input to the timer and sets the timer’s count clock, \( f_{CNT\_CLK} = \frac{PCLK (Hz)}{\text{prescaler}} \). The timer’s prescaler setting is a 4-bit value with \( \text{pres3} \) as the most significant bit and \( \text{pres} \) as the three least significant bits. The table below shows the prescaler values based on \( \text{pres3}:\text{pres} \). |
|      |       |        |       | \[ \begin{array}{cccc} 
\text{pres3} & \text{pres} & \text{Prescaler} & f_{CNT\_CLK} \\
0 & 0b000 & 1 & \frac{PCLK (Hz)}{1} \\
0 & 0b001 & 2 & \frac{PCLK (Hz)}{2} \\
0 & 0b010 & 4 & \frac{PCLK (Hz)}{4} \\
0 & 0b011 & 8 & \frac{PCLK (Hz)}{8} \\
0 & 0b100 & 16 & \frac{PCLK (Hz)}{16} \\
0 & 0b101 & 32 & \frac{PCLK (Hz)}{32} \\
0 & 0b110 & 64 & \frac{PCLK (Hz)}{64} \\
0 & 0b111 & 128 & \frac{PCLK (Hz)}{128} \\
1 & 0b000 & 256 & \frac{PCLK (Hz)}{256} \\
1 & 0b010 & 512 & \frac{PCLK (Hz)}{512} \\
1 & 0b100 & 1024 & \frac{PCLK (Hz)}{1024} \\
1 & 0b101 & 2048 & \frac{PCLK (Hz)}{2048} \\
1 & 0b110 & 4096 & \frac{PCLK (Hz)}{4096} \\
1 & 0b111 & Reserved & Reserved \\
\end{array} \] |
| 2:0  | tmode | R/W    | 0     | Timer Mode Select  
Sets the timer’s operating mode. |
|      |       |        |       | \[ \begin{array}{cc} 
\text{tmode} & \text{Timer Mode} \\
0b000 & One-Shot \\
0b001 & Continuous \\
0b010 & Counter \\
0b011 & PWM \\
0b100 & Capture \\
0b101 & Compare \\
0b110 & Gated \\
0b111 & Capture/Compare \\
\end{array} \] |
Table 16-6: Timer Non-Overlapping Compare Registers

<table>
<thead>
<tr>
<th>Timer Non-Overlapping Compare Register</th>
<th>TMRn_NOLCMP</th>
<th>[0x0014]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 31:16 | - | RO | 0 | Reserved for Future Use  
Do not modify this field from its default value. |
| 15:8 | nolhcmp | R/W | 0 | Non-Overlapping High Compare  
The 8-bit timer count value of non-overlapping time between the falling edge of PWM output $\phi_{A'}$ and the next rising edge of PWM output $\phi_A$. |
| 7:0 | nollcmp | R/W | 0 | Non-Overlapping Low Compare  
The 8-bit timer count value of non-overlapping time between the falling edge of PWM output $\phi_A$ and next rising edge of PWM output $\phi_{A'}$. |
17. **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.

17.1 **Instances**

The device provides 16 instances of the pulse train engine peripheral.

- PT0
- PT1
- PT2
- PT3
- PT4
- PT5
- PT6
- PT7
- PT8
- PT9
- PT10
- PT11
- PT12
- PT13
- PT14
- PT15

All peripheral registers share a common register set.

17.2 **Pulse Train Engine Features**

The pulse train outputs with individually programmable modes, patterns and output enables. The pulse train engine uses the Peripheral Clock (PCLK), \( f_{PTE_{CLK}} = f_{PCLK} \), ensuring all pulse train outputs use the same clock source.

- Independent or synchronous pulse train output operation
- Atomic Enable and Atomic Disable
- Synchronous enable or disable of pulse train output(s) without modification to non-intended pulse train outputs
- Multiple Output Modes:
  - Square Wave Output mode generates a repeating square wave (50% duty cycle)
  - Pattern Output mode for generating a customizable output wave based on a programmable bit pattern from 2 to 32 output cycles
- Global clock for all generated outputs
- Individual rate configuration for each pulse train output
- Configuration registers are modifiable while the pulse train engine is running
- Pulse train outputs can be halted and resumed at the same point

17.3 **Engine**

The pulse train engine uses the Peripheral Clock as the peripheral input clock,. Each pulse train output is individually configurable and independently controlled.

The following sections describe the available configuration options for each individual pulse train output.
17.3.1 Pulse Train Output Modes

Each pulse train output supports the following modes:

- Pulse Train Mode
- Bit Patter Length
- Square Wave Mode

17.3.1.1 Pulse Train Mode

When pulse train x (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 $PTn\_RATE\_LENGTH\_mode$.

$PTn\_RATE\_LENGTH\_mode = 1$ (PTn configured in Square Wave mode)

$PTn\_RATE\_LENGTH\_mode > 1$ (PTn configured in pulse train mode. The value of mode is the pattern bit length.)

$PTn\_RATE\_LENGTH\_mode = 0$ (PTn bit length configured for pulse train mode, 32-bit pattern)

If in pulse train Mode, Set the Bit Pattern

If an output is set to pulse train mode, then configure a custom bit pattern from 2-bits to 32-bits in length in the 32-bit register $PTn\_TRAIN$. The pattern is shifted out least significant bit (LSB) first. If the output is configured in Square Wave mode, then the $PTn\_TRAIN$ register is ignored.

Equation 17-1: Pulse Train Mode Output Function

$$PTn\_TRAIN = \text{[Bit pattern for PTn]}$$

Synchronize Two or More Outputs, if Needed

The write-only register $PTG\_RESYNC$ “PT Global Resync” allows two or more outputs to be reset and synchronized. Write to any bit in $PTG\_RESYNC$ to simultaneously reset any outputs in pulse train mode to the beginning of the pattern (the LSB) set in the $PTn\_TRAIN$ bit-pattern register, and reset the output to 0 for outputs in Square Wave mode.

17.3.1.2 Pulse Train Loop Mode

By default, a pulse train engine runs indefinitely until it is disabled by firmware.

A pulse train engine can be configured to repeat its pattern a specified number of times, called Loop mode. To select Loop mode, write a non-zero value to the 16-bit field $PTn\_LOOP\_count$. When the pulse train engine is enabled, this field decrements by 1 each time a complete pattern is shifted through the output pin. When the count reaches 0, the output is halted, and the corresponding flag in the $PTG\_INTFL$ register is set.

17.3.1.3 Pulse Train Loop Delay

If the pulse train is configured in Loop mode, a delay can be inserted after each repeated output pattern. To enable a delay, write the 12-bit field $PTn\_LOOP\_delay$ with the number of Peripheral Clock cycles to delay between the most significant bit (MSB) of the last pattern to the least-significant bit (LSB) of the next pattern. During this delay, the output is held at the MSB of the last pattern. If the loop counter has not reached 0, then it is decremented when the next pattern starts.

17.3.1.4 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 \textit{PTn\_RESTART} register for each pulse train engine. Note that the configuration for automatic restart is set using the pulse engine(s) triggered by the automatic restart, not the pulse train engine(s) that trigger the automatic restart. For example, the \textit{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 \textit{PTn\_RESTART.on\_pt\_n\_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 \textit{PTn\_RESTART.pt\_n\_select}.

If \textit{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 \textit{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:

\begin{itemize}
  \item \textbf{17.3.1.5 No Automatic Restart}
  \item Automatic Restart triggered by a stop event from pulse train x only
  \item Automatic Restart triggered by a stop event from pulse train y only
  \item Automatic Restart triggered by a stop event from both pulse train x and pulse train y
\end{itemize}

\section*{17.4 Enabling and Disabling a Pulse Train Output}

The \textit{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 \textit{PTG\_ENABLE} register. Halt a pulse train output by clearing the respective bit in the \textit{PTG\_ENABLE} register.

\textit{Note: Prior to changing a pulse train output’s configuration the corresponding pulse train output should be halted to prevent unexpected behavior.}

\section*{17.5 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 \textit{PTG\_ENABLE} register does not perform atomic access directly. Atomic operations are supported using the registers \textit{PTG\_SAFE\_EN}, \textit{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 \textit{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.

\subsection*{17.5.1 Pulse Train Atomic Enable}

\textit{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 \textit{PTG\_ENABLE} register to 1, which enables the corresponding pulse train engine. Writing a 0 to any bit position in the \textit{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.

17.5.2 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.

17.6 Pulse Train Halt and Disable

Once a pulse train engine is enabled and running, it continues to run until one of the following events stops the output:

The corresponding enable bit in the PTG_ENABLE register is cleared to 0 to halt the output.

A 1 is written to the corresponding disable bit in the PTG_SAFE_DIS register to halt the output.

The corresponding resync bit in the PTG_RESYNC register is cleared to 0 to halt and reset the output.

PTn_LOOP was initialized to a non-zero value, and the loop count has reached 0 (this has no effect in Square Wave mode; it only applies to pulse train mode).

When a pulse train is halted, the corresponding enable bit in PTG_ENABLE is automatically cleared to 0.

17.7 Pulse Train Interrupts

Each pulse train can generate an interrupt only if it is configured in pulse train mode, and the loop counter PTG_SAFE_DIS was initialized to a non-zero number. When PTG_SAFE_DIS counts down to 0, the corresponding status flag in the PTG_INTFL register is set. If the corresponding interrupt enable bit in the PTG_INTEN register is set, the event also generates an interrupt.

17.8 Registers

See Table 3-1: APB Peripheral Base Address Map for this peripheral/module’s base address. If multiple instances are provided, each will have a unique base address. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific reset, if applicable.

Table 17-1: Pulse Train Engine Register Summary

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>PTG_ENABLE</td>
<td>PT Global Enable/Disable Control</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>PTG_RESYNC</td>
<td>PT Global Resync</td>
</tr>
<tr>
<td>[0x0008]</td>
<td>PTG_INTFL</td>
<td>PT Stopped Global Status Flags</td>
</tr>
<tr>
<td>[0x000C]</td>
<td>PTG_INTEN</td>
<td>PT Global Interrupt Enable</td>
</tr>
<tr>
<td>[0x0010]</td>
<td>PTG_SAFE_EN</td>
<td>PT Global Safe Enable</td>
</tr>
<tr>
<td>[0x0014]</td>
<td>PTG_SAFE_DIS</td>
<td>PT Global Safe Disable</td>
</tr>
<tr>
<td>[0x0020]</td>
<td>PTn_RATE_LENGTH</td>
<td>PT0 Configuration</td>
</tr>
<tr>
<td>[0x0024]</td>
<td>PTn_TRAIN</td>
<td>PT0 Pulse Train Mode Bit Pattern</td>
</tr>
<tr>
<td>[0x0028]</td>
<td>PTn_LOOP</td>
<td>PT0 Loop Control</td>
</tr>
<tr>
<td>[0x002C]</td>
<td>PTn_RESTART</td>
<td>PT0 Automatic Restart</td>
</tr>
<tr>
<td>[0x0030]</td>
<td>PTn_RATE_LENGTH</td>
<td>PT1 Configuration</td>
</tr>
<tr>
<td>Offset</td>
<td>Register</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>0x0034</td>
<td>PTn_TRAIN</td>
<td>PT1 Pulse Train Mode Bit Pattern</td>
</tr>
<tr>
<td>0x0038</td>
<td>PTn_LOOP</td>
<td>PT1 Loop Control</td>
</tr>
<tr>
<td>0x003C</td>
<td>PTn_RESTART</td>
<td>PT1 Automatic Restart</td>
</tr>
<tr>
<td>0x0040</td>
<td>PTn_RATE_LENGTH</td>
<td>PT2 Configuration</td>
</tr>
<tr>
<td>0x0044</td>
<td>PTn_TRAIN</td>
<td>PT2 Pulse Train Mode Bit Pattern</td>
</tr>
<tr>
<td>0x0048</td>
<td>PTn_LOOP</td>
<td>PT2 Loop Control</td>
</tr>
<tr>
<td>0x004C</td>
<td>PTn_RESTART</td>
<td>PT2 Automatic Restart</td>
</tr>
<tr>
<td>0x0050</td>
<td>PTn_RATE_LENGTH</td>
<td>PT3 Configuration</td>
</tr>
<tr>
<td>0x0054</td>
<td>PTn_TRAIN</td>
<td>PT3 Pulse Train Mode Bit Pattern</td>
</tr>
<tr>
<td>0x0058</td>
<td>PTn_LOOP</td>
<td>PT3 Loop Control</td>
</tr>
<tr>
<td>0x005C</td>
<td>PTn_RESTART</td>
<td>PT3 Automatic Restart</td>
</tr>
<tr>
<td>0x0060</td>
<td>PTn_RATE_LENGTH</td>
<td>PT4 Configuration</td>
</tr>
<tr>
<td>0x0064</td>
<td>PTn_TRAIN</td>
<td>PT4 Pulse Train Mode Bit Pattern</td>
</tr>
<tr>
<td>0x0068</td>
<td>PTn_LOOP</td>
<td>PT4 Loop Control</td>
</tr>
<tr>
<td>0x006C</td>
<td>PTn_RESTART</td>
<td>PT4 Automatic Restart</td>
</tr>
<tr>
<td>0x0070</td>
<td>PTn_RATE_LENGTH</td>
<td>PT5 Configuration</td>
</tr>
<tr>
<td>0x0074</td>
<td>PTn_TRAIN</td>
<td>PT5 Pulse Train Mode Bit Pattern</td>
</tr>
<tr>
<td>0x0078</td>
<td>PTn_LOOP</td>
<td>PT5 Loop Control</td>
</tr>
<tr>
<td>0x007C</td>
<td>PTn_RESTART</td>
<td>PT5 Automatic Restart</td>
</tr>
<tr>
<td>0x0080</td>
<td>PTn_RATE_LENGTH</td>
<td>PT6 Configuration</td>
</tr>
<tr>
<td>0x0084</td>
<td>PTn_TRAIN</td>
<td>PT6 Pulse Train Mode Bit Pattern</td>
</tr>
<tr>
<td>0x0088</td>
<td>PTn_LOOP</td>
<td>PT6 Loop Control</td>
</tr>
<tr>
<td>0x008C</td>
<td>PTn_RESTART</td>
<td>PT6 Automatic Restart</td>
</tr>
<tr>
<td>0x0090</td>
<td>PTn_RATE_LENGTH</td>
<td>PT7 Configuration</td>
</tr>
<tr>
<td>0x0094</td>
<td>PTn_TRAIN</td>
<td>PT7 Pulse Train Mode Bit Pattern</td>
</tr>
<tr>
<td>0x0098</td>
<td>PTn_LOOP</td>
<td>PT7 Loop Control</td>
</tr>
<tr>
<td>0x009C</td>
<td>PTn_RESTART</td>
<td>PT7 Automatic Restart</td>
</tr>
<tr>
<td>0x00A0</td>
<td>PTn_RATE_LENGTH</td>
<td>PT8 Configuration</td>
</tr>
<tr>
<td>0x00A4</td>
<td>PTn_TRAIN</td>
<td>PT8 Pulse Train Mode Bit Pattern</td>
</tr>
<tr>
<td>0x00A8</td>
<td>PTn_LOOP</td>
<td>PT8 Loop Control</td>
</tr>
<tr>
<td>0x00AC</td>
<td>PTn_RESTART</td>
<td>PT8 Automatic Restart</td>
</tr>
<tr>
<td>0x00B0</td>
<td>PTn_RATE_LENGTH</td>
<td>PT9 Configuration</td>
</tr>
<tr>
<td>0x00B4</td>
<td>PTn_TRAIN</td>
<td>PT9 Pulse Train Mode Bit Pattern</td>
</tr>
<tr>
<td>0x00B8</td>
<td>PTn_LOOP</td>
<td>PT9 Loop Control</td>
</tr>
<tr>
<td>0x00BC</td>
<td>PTn_RESTART</td>
<td>PT9 Automatic Restart</td>
</tr>
<tr>
<td>0x00C0</td>
<td>PTn_RATE_LENGTH</td>
<td>PT10 Configuration</td>
</tr>
<tr>
<td>0x00C4</td>
<td>PTn_TRAIN</td>
<td>PT10 Pulse Train Mode Bit Pattern</td>
</tr>
<tr>
<td>0x00C8</td>
<td>PTn_LOOP</td>
<td>PT10 Loop Control</td>
</tr>
<tr>
<td>0x00CC</td>
<td>PTn_RESTART</td>
<td>PT10 Automatic Restart</td>
</tr>
<tr>
<td>0x00D0</td>
<td>PTn_RATE_LENGTH</td>
<td>PT11 Configuration</td>
</tr>
<tr>
<td>0x00D4</td>
<td>PTn_TRAIN</td>
<td>PT11 Pulse Train Mode Bit Pattern</td>
</tr>
<tr>
<td>0x00D8</td>
<td>PTn_LOOP</td>
<td>PT11 Loop Control</td>
</tr>
<tr>
<td>0x00DC</td>
<td>PTn_RESTART</td>
<td>PT11 Automatic Restart</td>
</tr>
<tr>
<td>0x00E0</td>
<td>PTn_RATE_LENGTH</td>
<td>PT12 Configuration</td>
</tr>
<tr>
<td>0x00E4</td>
<td>PTn_TRAIN</td>
<td>PT12 Pulse Train Mode Bit Pattern</td>
</tr>
<tr>
<td>0x00E8</td>
<td>PTn_LOOP</td>
<td>PT12 Loop Control</td>
</tr>
<tr>
<td>0x00EC</td>
<td>PTn_RESTART</td>
<td>PT12 Automatic Restart</td>
</tr>
</tbody>
</table>
### 17.9 Register Details

**Table 17-2: Pulse Train Engine Global Enable/Disable Register**

<table>
<thead>
<tr>
<th>PT Global Enable/Disable Control</th>
<th>PTG_ENABLE</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bits</strong></td>
<td><strong>Field</strong></td>
<td><strong>Access</strong></td>
</tr>
</tbody>
</table>
| 31:16 | - | RO | 0 | **Reserved for Future Use**  
Do not modify this field from its default value. |
| 15 | enable_pt15 | R/W | 0 | **Enable PT15**  
0: Disable  
1: Enable  
Note: *Disabling an active pulse train halts the output and does not generate a Stop Event.* |
| 14 | enable_pt14 | R/W | 0 | **Enable PT14**  
0: Disable  
1: Enable  
Note: *Disabling an active pulse train halts the output and does not generate a Stop Event.* |
| 13 | enable_pt13 | R/W | 0 | **Enable PT13**  
0: Disable  
1: Enable  
Note: *Disabling an active pulse train halts the output and does not generate a Stop Event.* |
| 12 | enable_pt12 | R/W | 0 | **Enable PT12**  
0: Disable  
1: Enable  
Note: *Disabling an active pulse train halts the output and does not generate a Stop Event.* |
| 11 | enable_pt11 | R/W | 0 | **Enable PT11**  
0: Disable  
1: Enable  
Note: *Disabling an active pulse train halts the output and does not generate a Stop Event.* |
| 10 | enable_pt10 | R/W | 0 | **Enable PT10**  
0: Disable  
1: Enable  
Note: *Disabling an active pulse train halts the output and does not generate a Stop Event.* |
<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>enable_pt9</td>
<td>R/W</td>
<td>0</td>
<td>Enable PT9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</td>
</tr>
<tr>
<td>8</td>
<td>enable_pt8</td>
<td>R/W</td>
<td>0</td>
<td>Enable PT8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</td>
</tr>
<tr>
<td>7</td>
<td>enable_pt7</td>
<td>R/W</td>
<td>0</td>
<td>Enable PT7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</td>
</tr>
<tr>
<td>6</td>
<td>enable_pt6</td>
<td>R/W</td>
<td>0</td>
<td>Enable PT6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</td>
</tr>
<tr>
<td>5</td>
<td>enable_pt5</td>
<td>R/W</td>
<td>0</td>
<td>Enable PT5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</td>
</tr>
<tr>
<td>4</td>
<td>enable_pt4</td>
<td>R/W</td>
<td>0</td>
<td>Enable PT4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</td>
</tr>
<tr>
<td>3</td>
<td>enable_pt3</td>
<td>R/W</td>
<td>0</td>
<td>Enable PT3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</td>
</tr>
<tr>
<td>2</td>
<td>enable_pt2</td>
<td>R/W</td>
<td>0</td>
<td>Enable PT2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</td>
</tr>
<tr>
<td>1</td>
<td>enable_pt1</td>
<td>R/W</td>
<td>0</td>
<td>Enable PT1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</td>
</tr>
<tr>
<td>0</td>
<td>enable_pt0</td>
<td>R/W</td>
<td>0</td>
<td>Enable PT0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Disabling an active pulse train halts the output and does not generate a Stop Event.</td>
</tr>
</tbody>
</table>

If $PTn\_LOOP\_count$ loop counter is set to a non-zero number, when the loop counter counts down to zero then the pulse train engine stops, and the corresponding enable bit is cleared.
### Table 17-3: Pulse Train Engine Resync Register

<table>
<thead>
<tr>
<th>PT Resync Register</th>
<th>PTG_RESYNC</th>
<th>[0x0004]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bits</strong></td>
<td><strong>Field</strong></td>
<td><strong>Access</strong></td>
</tr>
</tbody>
</table>
| 31:16 | - | RO | 0 | **Reserved for Future Use**
|  |  |  | | Do not modify this field from its default value. |
| 15 | pt15 | WO | - | **Resync Control for PT15**
|  |  |  | | Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. |
|  |  |  | | Setting multiple bits simultaneously in this register synchronizes the set outputs. |
|  |  |  | | 1: Reset/restart the pulse train |
|  |  |  | | 0: No effect |
|  |  |  | | **Note:** Writing 1 has no effect if the corresponding pulse train is disabled. |
| 14 | pt14 | WO | - | **Resync Control for PT14**
|  |  |  | | Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. |
|  |  |  | | Setting multiple bits simultaneously in this register synchronizes the set outputs. |
|  |  |  | | 1: Reset/restart the pulse train |
|  |  |  | | 0: No effect |
|  |  |  | | **Note:** Writing 1 has no effect if the corresponding pulse train is disabled. |
| 13 | pt13 | WO | - | **Resync Control for PT13**
|  |  |  | | Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. |
|  |  |  | | Setting multiple bits simultaneously in this register synchronizes the set outputs. |
|  |  |  | | 1: Reset/restart the pulse train |
|  |  |  | | 0: No effect |
|  |  |  | | **Note:** Writing 1 has no effect if the corresponding pulse train is disabled. |
| 12 | pt12 | WO | - | **Resync Control for PT12**
|  |  |  | | Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0. |
|  |  |  | | Setting multiple bits simultaneously in this register synchronizes the set outputs. |
|  |  |  | | 1: Reset/restart the pulse train |
|  |  |  | | 0: No effect |
|  |  |  | | **Note:** Writing 1 has no effect if the corresponding pulse train is disabled. |
| 11 | pt11 | WO | - | **Resync Control for PT11**
<p>|  |  |  | | 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 |
|  |  |  | | <strong>Note:</strong> Writing 1 has no effect if the corresponding pulse train is disabled. |</p>
<table>
<thead>
<tr>
<th>PT Resync Register</th>
<th>PTG_RESYNC</th>
<th>[0x0004]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 10 | pt10 | WO | - | **Resync Control for PT10**
Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0.
Setting multiple bits simultaneously in this register synchronizes the set outputs.
1: Reset/restart the pulse train
0: No effect
*Note: Writing 1 has no effect if the corresponding pulse train is disabled.* |
| 9 | pt9 | WO | - | **Resync Control for PT9**
Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0.
Setting multiple bits simultaneously in this register synchronizes the set outputs.
1: Reset/restart the pulse train
0: No effect
*Note: Writing 1 has no effect if the corresponding pulse train is disabled.* |
| 8 | pt8 | WO | - | **Resync Control for PT8**
Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0.
Setting multiple bits simultaneously in this register synchronizes the set outputs.
1: Reset/restart the pulse train
0: No effect
*Note: Writing 1 has no effect if the corresponding pulse train is disabled.* |
| 7 | pt7 | WO | - | **Resync Control for PT7**
Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0.
Setting multiple bits simultaneously in this register synchronizes the set outputs.
1: Reset/restart the pulse train
0: No effect
*Note: Writing 1 has no effect if the corresponding pulse train is disabled.* |
| 6 | pt6 | WO | - | **Resync Control for PT6**
Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0.
Setting multiple bits simultaneously in this register synchronizes the set outputs.
1: Reset/restart the pulse train
0: No effect
*Note: Writing 1 has no effect if the corresponding pulse train is disabled.* |
<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>pt5</td>
<td>WO</td>
<td>-</td>
<td>Resync Control for PT5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting multiple bits simultaneously in this register synchronizes the set outputs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Reset/restart the pulse train</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No effect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</td>
</tr>
<tr>
<td>4</td>
<td>pt4</td>
<td>WO</td>
<td>-</td>
<td>Resync Control for PT4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting multiple bits simultaneously in this register synchronizes the set outputs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Reset/restart the pulse train</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No effect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</td>
</tr>
<tr>
<td>3</td>
<td>pt3</td>
<td>WO</td>
<td>-</td>
<td>Resync Control for PT3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting multiple bits simultaneously in this register synchronizes the set outputs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Reset/restart the pulse train</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No effect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</td>
</tr>
<tr>
<td>2</td>
<td>pt2</td>
<td>WO</td>
<td>-</td>
<td>Resync Control for PT2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting multiple bits simultaneously in this register synchronizes the set outputs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Reset/restart the pulse train</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No effect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</td>
</tr>
<tr>
<td>1</td>
<td>pt1</td>
<td>WO</td>
<td>-</td>
<td>Resync Control for PT1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting multiple bits simultaneously in this register synchronizes the set outputs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Reset/restart the pulse train</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No effect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: Writing 1 has no effect if the corresponding pulse train is disabled.</td>
</tr>
<tr>
<td>PT Resync Register</td>
<td>PTG_RESYNC</td>
<td>[0x0004]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>---------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
<td>Reset</td>
<td>Description</td>
</tr>
</tbody>
</table>
| 0                | pt0       | WO      | -     | Resync Control for PT0  
Write 1 to reset the output of the pulse train. For pulse train mode the output is restarted to the beginning of the output pattern. For Square Wave mode the output is reset to 0.  
Setting multiple bits simultaneously in this register synchronizes the set outputs.  
1: Reset/restart the pulse train  
0: No effect  
Note: Writing 1 has no effect if the corresponding pulse train is disabled. |

### Table 17-4: Pulse Train Engine Stopped Interrupt Flag Register

<table>
<thead>
<tr>
<th>PT Stopped Interrupt Flag Register</th>
<th>PTG_INTFL</th>
<th>[0x0008]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 31:16                        | -         | RO      | 0     | Reserved for Future Use  
Do not modify this field from its default value. |
| 15                           | pt15      | R/W1C   | 0     | PT15 Stopped Status Flag  
This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used.  
Write 1 to clear.  
1: Pulse Train is stopped. |
| 14                           | pt14      | R/W1C   | 0     | PT14 Stopped Status Flag  
This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used.  
Write 1 to clear.  
1: Pulse Train is stopped. |
| 13                           | pt13      | R/W1C   | 0     | PT13 Stopped Status Flag  
This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used.  
Write 1 to clear.  
1: Pulse Train is stopped. |
| 12                           | pt12      | R/W1C   | 0     | PT12 Stopped Status Flag  
This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used.  
Write 1 to clear.  
1: Pulse Train is stopped. |
| 11                           | pt11      | R/W1C   | 0     | PT11 Stopped Status Flag  
This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used.  
Write 1 to clear.  
1: Pulse Train is stopped. |
| 10                           | pt10      | R/W1C   | 0     | PT10 Stopped Status Flag  
This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used.  
Write 1 to clear.  
1: Pulse Train is stopped. |
### PT Stopped Interrupt Flag Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 9    | pt9   | R/W/C  | 0     | PT9 Stopped Status Flag  
This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear.  
1: Pulse Train is stopped. |
| 8    | pt8   | R/W/C  | 0     | PT8 Stopped Status Flag  
This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear.  
1: Pulse Train is stopped. |
| 7    | pt7   | R/W/C  | 0     | PT7 Stopped Status Flag  
This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear.  
1: Pulse Train is stopped. |
| 6    | pt6   | R/W/C  | 0     | PT6 Stopped Status Flag  
This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear.  
1: Pulse Train is stopped. |
| 5    | pt5   | R/W/C  | 0     | PT5 Stopped Status Flag  
This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear.  
1: Pulse Train is stopped. |
| 4    | pt4   | R/W/C  | 0     | PT4 Stopped Status Flag  
This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear.  
1: Pulse Train is stopped. |
| 3    | pt3   | R/W/C  | 0     | PT3 Stopped Status Flag  
This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear.  
1: Pulse Train is stopped. |
| 2    | pt2   | R/W/C  | 0     | PT2 Stopped Status Flag  
This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear.  
1: Pulse Train is stopped. |
| 1    | pt1   | R/W/C  | 0     | PT1 Stopped Status Flag  
This bit is set to 1 by hardware when the corresponding pulse train is in Pulse Train Mode and the loop counter reaches 0. In Square Wave mode, this field is not used. Write 1 to clear.  
1: Pulse Train is stopped. |
### PT Stopped Interrupt Flag Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>pt0</td>
<td>R/WIC</td>
<td>0</td>
<td>PT0 Stopped Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Pulse Train is stopped.</td>
</tr>
</tbody>
</table>

#### PT Interrupt Enable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field from its default value.</td>
</tr>
<tr>
<td>15</td>
<td>pt15</td>
<td>R/W</td>
<td>0</td>
<td>PT15 Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>14</td>
<td>pt14</td>
<td>R/W</td>
<td>0</td>
<td>PT14 Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>13</td>
<td>pt13</td>
<td>R/W</td>
<td>0</td>
<td>PT13 Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>12</td>
<td>pt12</td>
<td>R/W</td>
<td>0</td>
<td>PT12 Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>11</td>
<td>pt11</td>
<td>R/W</td>
<td>0</td>
<td>PT11 Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>10</td>
<td>pt10</td>
<td>R/W</td>
<td>0</td>
<td>PT10 Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>9</td>
<td>pt9</td>
<td>R/W</td>
<td>0</td>
<td>PT9 Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
<td>Reset</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>--------</td>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>8</td>
<td>pt8</td>
<td>R/W</td>
<td>0</td>
<td><strong>PT8 Interrupt Enable</strong>&lt;br&gt;Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register.&lt;br&gt;0: Disabled.&lt;br&gt;1: Enabled.</td>
</tr>
<tr>
<td>7</td>
<td>pt7</td>
<td>R/W</td>
<td>0</td>
<td><strong>PT7 Interrupt Enable</strong>&lt;br&gt;Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register.&lt;br&gt;0: Disabled.&lt;br&gt;1: Enabled.</td>
</tr>
<tr>
<td>6</td>
<td>pt6</td>
<td>R/W</td>
<td>0</td>
<td><strong>PT6 Interrupt Enable</strong>&lt;br&gt;Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register.&lt;br&gt;0: Disabled.&lt;br&gt;1: Enabled.</td>
</tr>
<tr>
<td>5</td>
<td>pt5</td>
<td>R/W</td>
<td>0</td>
<td><strong>PT5 Interrupt Enable</strong>&lt;br&gt;Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register.&lt;br&gt;0: Disabled.&lt;br&gt;1: Enabled.</td>
</tr>
<tr>
<td>4</td>
<td>pt4</td>
<td>R/W</td>
<td>0</td>
<td><strong>PT4 Interrupt Enable</strong>&lt;br&gt;Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register.&lt;br&gt;0: Disabled.&lt;br&gt;1: Enabled.</td>
</tr>
<tr>
<td>3</td>
<td>pt3</td>
<td>R/W</td>
<td>0</td>
<td><strong>PT3 Interrupt Enable</strong>&lt;br&gt;Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register.&lt;br&gt;0: Disabled.&lt;br&gt;1: Enabled.</td>
</tr>
<tr>
<td>2</td>
<td>pt2</td>
<td>R/W</td>
<td>0</td>
<td><strong>PT2 Interrupt Enable</strong>&lt;br&gt;Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register.&lt;br&gt;0: Disabled.&lt;br&gt;1: Enabled.</td>
</tr>
<tr>
<td>1</td>
<td>pt1</td>
<td>R/W</td>
<td>0</td>
<td><strong>PT1 Interrupt Enable</strong>&lt;br&gt;Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register.&lt;br&gt;0: Disabled.&lt;br&gt;1: Enabled.</td>
</tr>
<tr>
<td>0</td>
<td>pt0</td>
<td>R/W</td>
<td>0</td>
<td><strong>PT0 Interrupt Enable</strong>&lt;br&gt;Write 1 to enable the interrupt for the corresponding PT when the flag is set in the PTG_INTFL register.&lt;br&gt;0: Disabled.&lt;br&gt;1: Enabled.</td>
</tr>
</tbody>
</table>
### 17.9.1 Pulse Train Engine Safe Enable Register

A 32-bit value written to this register performs an immediate binary OR with the contents of PTG\_ENABLE. The result is immediately stored in the PTG\_ENABLE.

#### Table 17-6: Pulse Train Engine Safe Enable Register

<table>
<thead>
<tr>
<th>Pulse Train Engine Safe Enable Register</th>
<th>PTG_SAFE_EN</th>
<th>[0x0010]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bits</strong></td>
<td><strong>Field</strong></td>
<td><strong>Access</strong></td>
</tr>
<tr>
<td>31:16</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td>15</td>
<td>safeen_pt15</td>
<td>WO</td>
</tr>
<tr>
<td>14</td>
<td>safeen_pt14</td>
<td>WO</td>
</tr>
<tr>
<td>13</td>
<td>safeen_pt13</td>
<td>WO</td>
</tr>
<tr>
<td>12</td>
<td>safeen_pt12</td>
<td>WO</td>
</tr>
<tr>
<td>11</td>
<td>safeen_pt11</td>
<td>WO</td>
</tr>
<tr>
<td>10</td>
<td>safeen_pt10</td>
<td>WO</td>
</tr>
<tr>
<td>9</td>
<td>safeen_pt9</td>
<td>WO</td>
</tr>
<tr>
<td>8</td>
<td>safeen_pt8</td>
<td>WO</td>
</tr>
<tr>
<td>7</td>
<td>safeen_pt7</td>
<td>WO</td>
</tr>
</tbody>
</table>
### Pulse Train Engine Safe Enable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>safeen_pt6</td>
<td>WO</td>
<td>-</td>
<td>Safe Enable Control for PT6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Writing a 1 sets <code>PTG_ENABLE.enable_pt6</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable corresponding pulse train</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No effect</td>
</tr>
<tr>
<td>5</td>
<td>safeen_pt5</td>
<td>WO</td>
<td>-</td>
<td>Safe Enable Control for PT5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Writing a 1 sets <code>PTG_ENABLE.enable_pt5</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable corresponding pulse train</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No effect</td>
</tr>
<tr>
<td>4</td>
<td>safeen_pt4</td>
<td>WO</td>
<td>-</td>
<td>Safe Enable Control for PT4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Writing a 1 sets <code>PTG_ENABLE.enable_pt4</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable corresponding pulse train</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No effect</td>
</tr>
<tr>
<td>3</td>
<td>safeen_pt3</td>
<td>WO</td>
<td>-</td>
<td>Safe Enable Control for PT3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Writing a 1 sets <code>PTG_ENABLE.enable_pt3</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable corresponding pulse train</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No effect</td>
</tr>
<tr>
<td>2</td>
<td>safeen_pt2</td>
<td>WO</td>
<td>-</td>
<td>Safe Enable Control for PT2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Writing a 1 sets <code>PTG_ENABLE.enable_pt2</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable corresponding pulse train</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No effect</td>
</tr>
<tr>
<td>1</td>
<td>safeen_pt1</td>
<td>WO</td>
<td>-</td>
<td>Safe Enable Control for PT1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Writing a 1 sets <code>PTG.Enable.enable_pt1</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable corresponding pulse train</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No effect</td>
</tr>
<tr>
<td>0</td>
<td>safeen_pt0</td>
<td>WO</td>
<td>-</td>
<td>Safe Enable Control for PT0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Writing a 1 sets <code>PTG_ENABLE.enable_pt0</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable corresponding pulse train</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No effect</td>
</tr>
</tbody>
</table>

### 17.9.2 Pulse Train Engine Safe Disable Register

A 32-bit value written to this register immediately disables the corresponding pulse train in the `PTG_ENABLE` register. The result is immediately stored in `PTG_ENABLE`. Setting a field to 1 disables the corresponding pulse train immediately.

<table>
<thead>
<tr>
<th>Pulse Train Engine Safe Disable Register</th>
<th>PTG_SAFE_DIS</th>
<th>[0x0014]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Access</td>
<td>Reset</td>
</tr>
<tr>
<td>31:16</td>
<td>RO</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>WO</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>WO</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
<td>--------</td>
</tr>
</tbody>
</table>
| 13   | safedis_pt13 | WO     | -     | Safe Disable Control for PT1  
|      |            |        |       | Writing a 1 clears PTG_ENABLE.enable_pt13.                                  |
|      |            |        |       | 1: Disable corresponding pulse train                                           |
|      |            |        |       | 0: No effect                                                                 |
| 12   | safedis_pt12 | WO     | -     | Safe Disable Control for PT12  
|      |            |        |       | Writing a 1 clears PTG_ENABLE.enable_pt12.                                  |
|      |            |        |       | 1: Disable corresponding pulse train                                           |
|      |            |        |       | 0: No effect                                                                 |
| 11   | safedis_pt11 | WO     | -     | Safe Disable Control for PT11  
|      |            |        |       | Writing a 1 clears PTG_ENABLE.enable_pt11.                                  |
|      |            |        |       | 1: Disable corresponding pulse train                                           |
|      |            |        |       | 0: No effect                                                                 |
| 10   | safedis_pt10 | WO     | -     | Safe Disable Control for PT10  
|      |            |        |       | Writing a 1 clears PTG_ENABLE.enable_pt10.                                  |
|      |            |        |       | 1: Disable corresponding pulse train                                           |
|      |            |        |       | 0: No effect                                                                 |
| 9    | safedis_pt9  | WO     | -     | Safe Disable Control for PT9  
|      |            |        |       | Writing a 1 clears PTG_ENABLE.enable_pt9.                                   |
|      |            |        |       | 1: Disable corresponding pulse train                                           |
|      |            |        |       | 0: No effect                                                                 |
| 8    | safedis_pt8  | WO     | -     | Safe Disable Control for PT8  
|      |            |        |       | Writing a 1 clears PTG_ENABLE.enable_pt8.                                   |
|      |            |        |       | 1: Disable corresponding pulse train                                           |
|      |            |        |       | 0: No effect                                                                 |
| 7    | safedis_pt7  | WO     | -     | Safe Disable Control for PT7  
|      |            |        |       | Writing a 1 clears PTG_ENABLE.enable_pt7.                                   |
|      |            |        |       | 1: Disable corresponding pulse train                                           |
|      |            |        |       | 0: No effect                                                                 |
| 6    | safedis_pt6  | WO     | -     | Safe Disable Control for PT6  
|      |            |        |       | Writing a 1 clears PTG_ENABLE.enable_pt6.                                   |
|      |            |        |       | 1: Disable corresponding pulse train                                           |
|      |            |        |       | 0: No effect                                                                 |
| 5    | safedis_pt5  | WO     | -     | Safe Disable Control for PT5  
|      |            |        |       | Writing a 1 clears PTG_ENABLE.enable_pt5.                                   |
|      |            |        |       | 1: Disable corresponding pulse train                                           |
|      |            |        |       | 0: No effect                                                                 |
| 4    | safedis_pt4  | WO     | -     | Safe Disable Control for PT4  
|      |            |        |       | Writing a 1 clears PTG_ENABLE.enable_pt4.                                   |
|      |            |        |       | 1: Disable corresponding pulse train                                           |
|      |            |        |       | 0: No effect                                                                 |
| 3    | safedis_pt3  | WO     | -     | Safe Disable Control for PT3  
|      |            |        |       | Writing a 1 clears PTG_ENABLE.enable_pt3.                                   |
|      |            |        |       | 1: Disable corresponding pulse train                                           |
|      |            |        |       | 0: No effect                                                                 |
### Pulse Train Engine Safe Disable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0    | safedis_pt0 | WO     | -     | **Safe Disable Control for PT0**  
Writing a 1 clears PTG_ENABLE.enable_pt0.  
1: Disable corresponding pulse train  
0: No effect |
| 1    | safedis_pt1 | WO     | -     | **Safe Disable Control for PT1**  
Writing a 1 clears PTG_ENABLE.enable_pt1.  
1: Disable corresponding pulse train  
0: No effect |
| 2    | safedis_pt2 | WO     | -     | **Safe Disable Control for PT2**  
Writing a 1 clears PTG_ENABLE.enable_pt2.  
1: Disable corresponding pulse train  
0: No effect |

### Pulse Train Engine Configuration Register

<table>
<thead>
<tr>
<th>Pulse Train Configuration Register</th>
<th>PTn_RATE_LENGTH</th>
<th>[0x0020]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>--------</td>
</tr>
</tbody>
</table>
| 31:27| mode     | R/W    | 0b00001| **Square Wave or Pulse Train Output Mode**  
Sets either pulse train mode with length, or Square Wave mode.  
0: Pulse train mode, 32-bits long  
1: Square Wave mode  
2: Pulse train mode, 2-bits long  
3: Pulse train mode, 3-bits long  
... etc ...  
31: Pulse train mode, 31-bits long  
**Note:** If this field is set to 1, Square Wave mode, the PTn_LENGTH register is not used. |
| 26:0 | rate_control | R/W | 0 | **Pulse Train Enable and Rate Control**  
Defines the rate at which the output for PTn changes state by setting the divisor of the PT Clock, where:  
\[ f_{PTn} = \frac{f_{PTE_CLK}}{rate\_control} \]  
0: Output halted  
1: \[ f_{PTn} = f_{PTE\_CLK} \]  
2: \[ f_{PTn} = \frac{f_{PTE\_CLK}}{2} \]  
3: \[ f_{PTn} = \frac{f_{PTE\_CLK}}{3} \]  
...  
0x7FFFF: \[ f_{PTn} = \frac{f_{PTE\_CLK}}{0\times7FFFF} \] |

### Pulse Train Mode Bit Pattern Register

<table>
<thead>
<tr>
<th>Pulse Train Mode Bit Pattern</th>
<th>PTn_TRAIN</th>
<th>[0x0024]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>--------</td>
</tr>
</tbody>
</table>
| 31:0 | ptn_train | R/W    | 0     | **Pulse Train Mode Bit Pattern**  
Write the repeating bit pattern that is shifted out, LSB first, when configured in pulse train mode. Set the bit pattern length with the PTn_RATE_LENGTH.mode field.  
**Note:** This register is ignored in Square Wave mode.  
**Note:** 0x0000 0000 and 0x0001 0000 are invalid values for this register. |
### Table 17-10: Pulse Train n Loop Configuration Register

<table>
<thead>
<tr>
<th>Pulse Train Loop Configuration</th>
<th>PTn_LOOP</th>
<th>[0x0028]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bits</strong></td>
<td><strong>Field</strong></td>
<td><strong>Access</strong></td>
</tr>
</tbody>
</table>
| 31:28 | - | RO | 0 | **Reserved for Future Use**  
Do not modify this field from its default value. |
| 27:16 | delay | R/W | 0 | **Pulse Train Delay Between Loops**  
Sets the delay, in number of Peripheral Clock cycles, that the output pauses between loops.  
The bitfield count is decremented after the delay.  
If firmware writes a 0 to bitfield count, this field is ignored. |
| 15:0 | count | R/W | 0 | **Pulse Train Loop Countdown**  
Sets the number of times a pulse train pattern is repeated until it automatically stops.  
Reading this field returns the number of loops remaining.  
When this field counts down to zero, the corresponding PTn_INTFL flag is set.  
Write 0 to have the pulse train pattern repeat indefinitely.  
Ignored in Square Wave mode. |

### Table 17-11: Pulse Train n Automatic Restart Configuration Register

<table>
<thead>
<tr>
<th>Pulse Train Automatic Restart Configuration</th>
<th>PTn_RESTART</th>
<th>[0x002C]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bits</strong></td>
<td><strong>Field</strong></td>
<td><strong>Access</strong></td>
</tr>
</tbody>
</table>
| 31:16 | - | RO | 0 | **Reserved for Future Use**  
Do not modify this field from its default value. |
| 15 | on_pt_y_loop_exit | R/W | 0 | **Enable Automatic Restart for This Pulse Train on PTy Stop Event**  
0: Disable automatic restart  
1: When PTy has a Stop Event, automatically restart this pulse train from the beginning of its pattern. |
| 14:11 | - | RO | 0 | **Reserved for Future Use**  
Do not modify this field from its default value. |
| 12:8 | pt_y_select | R/W | 0 | **Select PTy**  
Select the pulse train number to be associated with PTy. This engine must be in pulse train mode.  
0b00000: PT0  
0b00001: PT1  
0b00010: PT2  
0b00011: PT3  
0b00100: PT4  
0b00101: PT5  
0b00110: PT6  
0b00111: PT7  
0b01000: PT8  
0b01001: PT9  
0b01010: PT10  
0b01011: PT11  
0b01100: PT12  
0b01101: PT13  
0b01110: PT14  
0b01111: PT15  
0b1xxxx: Reserved. |
### Pulse Train Automatic Restart Configuration

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>on_pt_n_loop</td>
<td>R/W</td>
<td>0</td>
<td>Enable Automatic Restart for this Pulse Train on a PTn Stop Event</td>
</tr>
<tr>
<td></td>
<td>_loop_exit</td>
<td></td>
<td></td>
<td>0: Disable automatic restart</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: When PTn has a Stop Event, automatically restart pulse train from the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>beginning of its pattern.</td>
</tr>
<tr>
<td>6:5</td>
<td></td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field from its default value.</td>
</tr>
<tr>
<td>4:0</td>
<td>pt_n_select</td>
<td>R/W</td>
<td>0</td>
<td>Select PTn</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Select the pulse train number to be associated with PTn. This engine must</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>be in pulse train mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00000: PT0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00001: PT1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00010: PT2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00011: PT3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00100: PT4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00101: PT5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00110: PT6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00111: PT7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01000: PT8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01001: PT9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01010: PT10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01011: PT11</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01100: PT12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01101: PT13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01110: PT14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01111: PT15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b1xxxx: Reserved.</td>
</tr>
</tbody>
</table>
18. Real-Time Clock (RTC)

18.1 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 32.768 time base. It can be generated from the internal crystal oscillator driving an external 32.768kHz crystal between the 32KIN and 32KOUT pins, or a 32.768kHz square wave driven directly into the 32KIN pin. Refer to the datasheet for the required electrical characteristics of the external crystal.

A user-configurable, digital frequency trim is provided for applications requiring higher accuracy.

The 32-bit seconds register \texttt{RTC\_SEC} is incremented ever time there is a rollover of d \texttt{RTC\_SSEC}.sec sub-seconds field.

Two alarm functions are provided:

- A programmable time-of-day alarm provides a single event, alarm timer using the \texttt{RTC\_TODA} alarm register, \texttt{RTC\_SEC} register, and \texttt{RTC\_CTRL.tod_alarm_en} field.
- A programmable sub-second provides a recurring alarm using the \texttt{RTC\_SSECA} and \texttt{RTC\_CTRL.ssec_alarm_en} field.

The RTC is powered in the always-on domain or if applicable, while there is a valid voltage on the VRTC pin.

Disabling the RTC stops incrementing \texttt{RTC\_SSEC}, \texttt{RTC\_SEC}, and the internal RTC sub-second counter, but preserves their current value. The 32kHz oscillator is not affected by the \texttt{RTC\_CTRL.enable} field.

The RTC increments the \texttt{RTC\_TRIM.vrtc_tmr} field every 32 seconds while the RTC is enabled.

\textit{Figure 18-1. MAX32665—MAX32668 RTC Block Diagram (12-bit Sub-Second Counter)}
18.2 Instances

One instance of the RTC peripheral is provided.

The RTC counter and alarm registers are shown in Table 18-1. MAX32665—MAX32668 RTC Counter and Alarm Registers.

Table 18-1. MAX32665—MAX32668 RTC Counter and Alarm Registers

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
<th>Counter Increment</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC_SEC</td>
<td>32</td>
<td>1s</td>
<td>1s</td>
<td>136 yrs</td>
<td>Seconds Counter Register</td>
</tr>
<tr>
<td>RTC_SSEC</td>
<td>12</td>
<td>244µs (1/4kHz)</td>
<td>244µs</td>
<td>1s</td>
<td>Sub-Seconds Counter Register</td>
</tr>
<tr>
<td>RTC_TODA</td>
<td>20</td>
<td>1s</td>
<td>1s</td>
<td>12 days</td>
<td>Time-of-Day Alarm Register</td>
</tr>
<tr>
<td>RTC_SSECA</td>
<td>32</td>
<td>244 µs (1/4kHz)</td>
<td>244 µs</td>
<td>1s</td>
<td>Sub-Second Alarm Register</td>
</tr>
</tbody>
</table>

18.3 Register Access Control

Access protection mechanisms prevent software from accessing critical registers and fields while RTC while hardware is updating them. Monitoring the RTC_CTRL.busy and RTC_CTRL.ready fields allows software to determine when it is safe to write to registers and when registers will return valid results.

Table 18-2. RTC Register Access

<table>
<thead>
<tr>
<th>Register</th>
<th>Field</th>
<th>Read Access</th>
<th>Write Access</th>
<th>Busy = 1 during write</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC_SEC</td>
<td>All</td>
<td>RTC_CTRL.busy = 0</td>
<td>RTC_CTRL.busy = 0</td>
<td>Y</td>
<td>Seconds Counter Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC_CTRL.ready = 1</td>
<td>RTC_CTRL.ready = 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTC_SSEC</td>
<td>.ssec</td>
<td>RTC_CTRL.busy = 0</td>
<td>RTC_CTRL.busy = 0</td>
<td>Y</td>
<td>Sub-Seconds Counter Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC_CTRL.ready = 1</td>
<td>RTC_CTRL.ready = 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTC_TODA</td>
<td>All</td>
<td>Always</td>
<td>RTC_CTRL.busy = 0</td>
<td>Y</td>
<td>Time-of-Day Alarm Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC_CTRL.tod_alarm_en = 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTC_SSECA</td>
<td>All</td>
<td>Always</td>
<td>RTC_CTRL.busy = 0</td>
<td>Y</td>
<td>Sub-Second Alarm Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC_CTRL.ssec_alarm_en = 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTC_TRIM</td>
<td>All</td>
<td>Always</td>
<td>RTC_CTRL.busy = 0</td>
<td>Y</td>
<td>Trim Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC_CTRL.write_en = 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTC_OSCCTRL</td>
<td>All</td>
<td>Always</td>
<td>RTC_CTRL.busy = 0</td>
<td>Y</td>
<td>Oscillator Control Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC_CTRL.write_en = 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTC_CTRL</td>
<td>enable</td>
<td>Always</td>
<td>RTC_CTRL.busy = 0</td>
<td>Y</td>
<td>RTC Enable field</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC_CTRL.write_en = 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>tod_alarm_en</td>
<td>Always</td>
<td>RTC_CTRL.busy = 0</td>
<td>Y</td>
<td>Time-of-Day Alarm enable field</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC_CTRL.write_en = 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ssec_alarm_en</td>
<td>Always</td>
<td>RTC_CTRL.busy = 0</td>
<td>Y</td>
<td>Sub-Second Alarm enable field</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTC_CTRL.write_en = 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>All other bits</td>
<td>Always</td>
<td>RTC_CTRL.busy = 0</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

18.3.1 RTC_SEC and RTC_SSEC Read Access Control

Software reads of the RTC_SEC and RTC_SSEC registers will return invalid results if the read operation occurs on the same cycle that the register is being updated by hardware. To avoid this, hardware clears RTC_CTRL.ready to 0 during the update cycle and sets RTC_CTRL.ready to 1 again when the cycle is complete. The period of the RTC_CTRL.ready bit set/clear activity provides a large window during which the count registers are readable.
Software can use two methods to ensure valid results when reading \textit{RTC\_SEC} and \textit{RTC\_SSEC}:

- Poll \textit{RTC\_CTRL.ready} until it reads 1 before reading the registers.
- Set the \textit{RTC\_CTRL.ready\_int\_en} field to 1 to generate an RTC interrupt when the next update cycle is complete and hardware sets \textit{RTC\_CTRL.ready} to 1 again. This avoids the software overhead associated with polling to observe the state of \textit{RTC\_CTRL.ready}.

Software can clear \textit{RTC\_CTRL.ready} following the read of the registers to acknowledge the read.

\textbf{18.3.2 RTC Write Access Control}

The read-only status field \textit{RTC\_CTRL.busy} is set to 1 by hardware following a software instruction that writes to specific registers. The bit remains 1 while the software updates are being synchronized into the RTC. Software should not write to any of the registers until hardware indicates the synchronization is complete by clearing \textit{RTC\_CTRL.busy} to 0.

\textit{Figure 18.2. RTC Busy/Ready Signal Timing}

\textbf{18.4 RTC Alarm Functions}

The RTC provides time-of-day and sub-second interval alarm functions. The time-of-day alarm is implemented by matching the count values in the counter register with the value stored in the alarm register. The sub-second interval alarm provides an auto-reload timer that is driven by the trimmed RTC clock source.

\textbf{18.4.1 Time-of-Day Alarm}

Program the RTC Time-of-Day Alarm register (\textit{RTC\_TODA}) to configure the time-of-day-alarm. The alarm triggers when the value stored in \textit{HTIMER\_RAS} matches the lower 20 bits of the \textit{RTC\_SEC} seconds count register. This allows programming the time-of-day-alarm to any future value between 1 second and 12 days relative to the current time with a resolution of 1 second. You must disable the time-of-day alarm before changing the \textit{RTC\_TODA.tod} field.

When the alarm occurs, a single event sets the Time-of-Day Alarm Interrupt Flag (\textit{RTC\_CTRL.tod\_alarm\_fl}) to 1.

Setting the \textit{RTC\_CTRL.tod\_alarm\_fl} bit to 1 in software results in an interrupt request to the processor if the Alarm Time-of-Day Interrupt Enable (\textit{RTC\_CTRL.tod\_alarm\_en}) bit is set to 1, and the RTC’s system interrupt enable is set.
18.4.2 Sub-Second Alarm

The RTC_SSECA and RTC_CTRL.ssec_alarm_en 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_en) bit to 1 enables the sub-second alarm. Once enabled, an internal alarm counter begins incrementing from the RTC_SSECA value. When the counter rolls over from 0xFFFF FFFF to 0x0000 0000, hardware sets the RTC_CTRL.ssec_alarm_fl bit triggering the alarm. At the same time, hardware also reloads the counter with the value previously written to RTC_SSECA.rssa.

You must disable the sub-second interval alarm, RTC_CTRL.ssec_alarm_en, 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.
18.4.3 RTC Interrupt and Wakeup Configuration

The following are a list of conditions that, when enabled, can generate an RTC interrupt.

- Time-of-Day Alarm
- Sub-second Alarm
- Ready field asserted high, signaling write access permitted

RTC can be configured so the Time-of-day and sub-second alarms are a wakeup source for exiting the following low power modes:

- Backup
- Deepsleep

Figure 18-3. RTC Interrupt/Wakeup Diagram Wakeup Function

Use this procedure to enable the RTC as a wakeup source:

1. Software configures the RTC interrupt enable bits so one or more interrupt conditions will generate an RTC interrupt.
2. Create a RTC IRQ handler function and register the address of the RTC IRQ handler using the NVIC.
3. Software sets GCR_PMR.rtcwken to 1 to enable the system wakeup for by the RTC.
4. Software places the device into the desired low power mode. Refer to section Operating Modes for details on entering DEEPSLEEP or BACKUP mode.

18.4.4 Square Wave Output

The RTC can output a 50% duty cycle square wave signal derived from the 32kHz oscillator on a selected device pin. Refer to Table 18-3. 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 18-3. MAX32665—MAX32668 RTC Square Wave Output Configuration

<table>
<thead>
<tr>
<th>Function</th>
<th>Option</th>
<th>Control Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Pin*</td>
<td>P0.19: SQWOUT</td>
<td>(MCR_OUTEN_sqwout0en = 1)</td>
</tr>
<tr>
<td></td>
<td>P0.27: SQWOUT</td>
<td>(MCR_OUTEN_sqwout1en = 1)</td>
</tr>
<tr>
<td>Frequency Select</td>
<td>1Hz (Compensated)</td>
<td>(RTC_OSCCTRL_freq_sel = 0b00)</td>
</tr>
<tr>
<td></td>
<td>512Hz (Compensated)</td>
<td>(RTC_OSCCTRL_freq_sel = 0b01)</td>
</tr>
<tr>
<td></td>
<td>4kHz</td>
<td>(RTC_OSCCTRL_freq_sel = 0b10)</td>
</tr>
<tr>
<td></td>
<td>32kHz</td>
<td>(RTC_OSCCTRL_freq_sel = 0bxx) (RTC_OSCCTRL_32k_out = 1)</td>
</tr>
<tr>
<td>Output Enable</td>
<td>1Hz (Compensated)</td>
<td>(RTC_CTRL_sqwout_en = 1) (RTC_OSCCTRL_32k_out = 0)</td>
</tr>
<tr>
<td></td>
<td>512Hz (Compensated)</td>
<td>(RTC_CTRL_sqwout_en = 1) (RTC_OSCCTRL_32k_out = 0)</td>
</tr>
<tr>
<td></td>
<td>4kHz</td>
<td>(RTC_CTRL_sqwout_en = 1) (RTC_OSCCTRL_32k_out = 0)</td>
</tr>
<tr>
<td></td>
<td>32kHz</td>
<td>(RTC_OSCCTRL_32k_out = 1)</td>
</tr>
</tbody>
</table>

*Note: The square wave output can be enabled on more than one pin if desired.

Use the following procedure to generate the square wave:

Software configures the fields shown in Table 18-3. to select the desired frequency.

If more than one output pin is available, software configures the fields shown in Table 18-3. MAX32665—MAX32668 RTC Square Wave Output Configuration to select the output pin.

Software configures the fields shown in Table 18-3. to enable the square wave output.

18.5 RTC Calibration

A digital trim facility provides the ability to compensate for RTC inaccuracies of up to ±127ppm when compared against an external reference clock. The trimming function utilizes an independent, dedicated timer which increments the sub-second register based on a user-supplied, 2’s compliment value in the \(RTC\_TRIM\) register as shown in Figure 18-4. Internal Implementation of Digital Trim, 4kHz.
Figure 18-4. Internal Implementation of Digital Trim, 4kHz

RTC_CTRL.enable = 1

 RTC_TRIM = 0? (TRIM FUNCTION DISABLED)

 Y

 N

 TRIM_COUNT = 1,000,000

 LOOP_COUNT = 0

 TRIM_COUNT = TRIM_COUNT + RTC_TRIM
 LOOP_COUNT = LOOP_COUNT + 1 (INCREMENTS 4kHz)

 LOOP_COUNT = 16?

 Y [4kHz DOMAIN]

 TRIM_COUNT < 1,000,000?

 Y

 INCREMENT RTS_SEC TWICE

 N

 TRIM_COUNT > 2,000,000?

 Y

 DO NOT INCREMENT

 N

 INCREMENT RTS_SEC

 [256Hz DOMAIN]
Complete the following steps to perform an RTC calibration:

1. If not already, software configures and enables one of the compensated calibration frequencies as described in section Square Wave Output.
2. Measure the frequency on the square wave output pin and compute the deviation from an accurate reference clock.
3. Software clears \texttt{RTC\_CTRL\_ready} to 0.
4. Wait for \texttt{RTC\_CTRL\_ready} = 1 by:
5. Software sets \texttt{RTC\_CTRL\_ready\_int\_en} to 1 to generate an interrupt when \texttt{RTC\_CTRL\_ready} = 1, or
6. Software polls \texttt{RTC\_CTRL\_ready} until the field = 1.
7. Software polls until \texttt{RTC\_CTRL\_busy}=0 to make sure any previous operations are complete
8. Software sets \texttt{RTC\_CTRL\_write\_en} to 1 to allow access to \texttt{RTC\_TRIM}.
9. Software writes to \texttt{RTC\_TRIM} register as desired to correct for measured inaccuracy.
10. Hardware clears \texttt{RTC\_CTRL\_busy} =0.
11. Software clears \texttt{RTC\_CTRL\_write\_en} to 0.
12. Repeat the process as needed until the desired accuracy is achieved

### 18.6 Registers

Refer to Table 3-1: APB Peripheral Base Address Map for this peripheral/module's base address. If multiple instances are provided, each will have a unique base address. All fields in this peripheral are reset on POR only; refer to the field description for details.

<table>
<thead>
<tr>
<th>Register</th>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{RTC_SEC}</td>
<td>[0x0000]</td>
<td>RTC Seconds Counter Register</td>
</tr>
<tr>
<td>\texttt{RTC_SSEC}</td>
<td>[0x0004]</td>
<td>RTC Sub-Second Counter Register</td>
</tr>
<tr>
<td>\texttt{RTC_TODA}</td>
<td>[0x0008]</td>
<td>RTC Time-of-Day Alarm Register</td>
</tr>
<tr>
<td>\texttt{RTC_SSECA}</td>
<td>[0x000C]</td>
<td>RTC Sub-Second Alarm Register</td>
</tr>
<tr>
<td>\texttt{RTC_CTRL}</td>
<td>[0x0010]</td>
<td>RTC Control Register</td>
</tr>
<tr>
<td>\texttt{RTC_TRIM}</td>
<td>[0x0014]</td>
<td>RTC 32KHz Oscillator Digital Trim Register</td>
</tr>
<tr>
<td>\texttt{RTC_OSCCCTRL}</td>
<td>[0x0018]</td>
<td>RTC 32KHz Oscillator Control Register</td>
</tr>
</tbody>
</table>

### 18.7 Register Details

**Table 18-5. RTC Seconds Counter Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0  | sec   | R/W    | 0     | Seconds Counter
|       |       |        |       | This register is a binary count of seconds. |

**Table 18-6. RTC Sub-Second Counter Register (12-bit)**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:12</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td>11:0</td>
<td>ssec</td>
<td>R/W</td>
<td>0</td>
<td>Sub-Seconds Counter (12-bit)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>\texttt{RTC_SEC} increments when this field rolls from 0xFFF to 0x0000</td>
</tr>
</tbody>
</table>
### Table 18-7. RTC Time-of-Day Alarm Register

<table>
<thead>
<tr>
<th>RTC Time-of-Day Alarm</th>
<th>RTC_TODA</th>
<th>[0x0008]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:20</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19:0</td>
<td>tod_alarm</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 18-8. RTC Sub-Second Alarm Register

<table>
<thead>
<tr>
<th>RTC Sub-Second Alarm</th>
<th>RTC_SSECA</th>
<th>[0x000C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:0</td>
<td>ssec_alarm</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 18-9. RTC Control Register

<table>
<thead>
<tr>
<th>RTC Control Register</th>
<th>RTC_CTRL</th>
<th>[0x0010]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>31:16</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>write_en</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14:13</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:11</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:9</td>
<td>freq_sel</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>sqwout_en</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
<tr>
<td>------</td>
<td>---------------</td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td>RTC_CTRL [0x0010]</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>ssec_alarm_fl</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>tod_alarm_fl</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ready_int_en</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ready</td>
<td>R/WOO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>busy</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ssec_alarm_en</td>
<td>R/W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### RTC Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>tod_alarm_en</td>
<td>R/W</td>
<td>0*</td>
<td>Time-of-Day Alarm Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Check the <strong>RTC_CTRL</strong>.busy flag after writing to this field to determine when the RTC synchronization is complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*Note: Reset on POR only.</td>
</tr>
<tr>
<td>0</td>
<td>enable</td>
<td>R/W</td>
<td>0*</td>
<td>Real-Time Clock Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The <strong>RTC_CTRL</strong>.write enable (<strong>RTC_CTRL</strong>.write_en) bit must be set and <strong>RTC_CTRL</strong>.busy (RTC_CTRL.busy) must read 0 before writing to this field. After writing to this bit, check the <strong>RTC_CTRL</strong>.busy flag for 0 to determine when the RTC synchronization is complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*Note: Reset on POR only.</td>
</tr>
</tbody>
</table>

### RTC 32KHz Oscillator Digital Trim Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>vrtc_tmr</td>
<td>R/W</td>
<td>0*</td>
<td>VRTC Time Counter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Hardware increments this field every 32s while the RTC is enabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*Note: Reset on POR only.</td>
</tr>
<tr>
<td>7:0</td>
<td>trim</td>
<td>R/W</td>
<td>0*</td>
<td>RTC Trim</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*Note: Reset on POR only.</td>
</tr>
</tbody>
</table>

### RTC Oscillator Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:6</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>5</td>
<td>32kout</td>
<td>R/W</td>
<td>0</td>
<td>RTC Square Wave Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enables the 32kHz oscillator output or the external clock source is output on square wave output pin. See Table 18-3. for configuration details.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*Note: Reset on POR only.</td>
</tr>
<tr>
<td>4</td>
<td>bypass</td>
<td>R/W</td>
<td>0</td>
<td>RTC Crystal Bypass</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This field disables the RTC oscillator and allows an external clock source to be driven on the 32KIN pin.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disable bypass.RTC timebase is external 32kHz crystal.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enable bypass.RTC timebase is external square wave driven on 32KIN.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*Note: Reset on POR only.</td>
</tr>
<tr>
<td>3:0</td>
<td>-</td>
<td>R/W</td>
<td>0b1001</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Software must not modify this field from its current value.</td>
</tr>
</tbody>
</table>
19. **Watchdog Timer (WDT)**

The watchdog timer protects against corrupt or unreliable software, power faults, and other system-level problems, which may place the microcontroller into an improper operating state. When the application is executing properly, application software periodically resets the watchdog counter. If the watchdog timer interrupt is enabled and the software does not reset the counter within the interrupt time period \((WDTn_CTRL.int\_period)\), the watchdog timer generates a watchdog timer interrupt. If the watchdog timer reset is enabled and the software does not reset the counter within the reset time period \((WDTn_CTRL.rst\_period)\), the watchdog timer generates a system reset.

*Figure 19-1* shows the block diagram of the watchdog timers.
19.1 Features

- Sixteen programmable time periods for the watchdog interrupt $2^{16}$ through $2^{31}$ PCLK cycles
- Sixteen programmable time periods for the watchdog reset $2^{16}$ through $2^{31}$ PCLK cycles
- The watchdog timer counter is reset on all forms of reset

19.2 Usage

Utilizing the watchdog timer in the application software is straightforward. As early as possible in the application software, enable the watchdog timer interrupt and watchdog timer reset. Periodically the application software must write to the WDT_RST register to reset the watchdog counter. If program execution becomes lost, the watchdog timer interrupt will occur, giving the system a “last chance” to recover from whatever circumstance caused the improper code execution. The interrupt routine may either attempt to repair the situation or allow the watchdog timer reset to occur. In the event of a system software failure, the interrupt will not be executed, and the watchdog system reset will recover operation.
As soon as possible after a reset, the application software should interrogate the $WDTn\_CTRL\_rst\_flag$ to determine if the reset event resulted from a watchdog timer reset. If so, application software should assume that there was a program execution error and take whatever steps necessary to guard against a software corruption issue.

### 19.3 Interrupt and Reset Period Timeout Configuration

Each watchdog timer supports two independent timeout periods, the interrupt period timeout and reset period timeout.

- **Interrupt Period Timeout**: $WDTn\_CTRL\_int\_period$ sets the number of PCLK cycles until a watchdog timer interrupt is generated. This period must be less than the Reset Period Timeout for the watchdog timer interrupt to occur.

- **Reset Period Timeout**: $WDTn\_CTRL\_rst\_period$ sets the number of PCLK cycles until a system reset event occurs.

The interrupt and reset period timeouts are calculated using *Equation 19-1* and *Equation 19-2* respectively, where $f_{PCLK} = \frac{f_{SYS\_CLK}}{2}$. *Table 19-1* shows example interrupt period timeout values.

*Equation 19-1: Watchdog Timer Interrupt Period*

$$T_{INT\_PERIOD} = \left(\frac{1}{f_{PCLK}}\right) \times 2^{(31 - WDTn\_CTRL\_int\_period)}$$

*Equation 19-2. Watchdog Timer Reset Period*

$$T_{RST\_PERIOD} = \left(\frac{1}{f_{PCLK}}\right) \times 2^{(31 - WDTn\_CTRL\_rst\_period)}$$

*Table 19-1: Watchdog Timer Interrupt Period f\_SYS\_CLK = 96MHz and f\_PCLK = 48MHz*

<table>
<thead>
<tr>
<th>$WDTn_CTRL$ int_period</th>
<th>$T_{INT_PERIOD}$ (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0.001</td>
</tr>
<tr>
<td>14</td>
<td>0.003</td>
</tr>
<tr>
<td>13</td>
<td>0.005</td>
</tr>
<tr>
<td>12</td>
<td>0.011</td>
</tr>
<tr>
<td>11</td>
<td>0.022</td>
</tr>
<tr>
<td>10</td>
<td>0.044</td>
</tr>
<tr>
<td>9</td>
<td>0.087</td>
</tr>
<tr>
<td>8</td>
<td>0.175</td>
</tr>
<tr>
<td>7</td>
<td>0.350</td>
</tr>
<tr>
<td>6</td>
<td>0.699</td>
</tr>
<tr>
<td>5</td>
<td>1.398</td>
</tr>
<tr>
<td>4</td>
<td>2.796</td>
</tr>
<tr>
<td>3</td>
<td>5.592</td>
</tr>
<tr>
<td>2</td>
<td>11.185</td>
</tr>
<tr>
<td>1</td>
<td>22.370</td>
</tr>
<tr>
<td>0</td>
<td>Disabled</td>
</tr>
</tbody>
</table>

### 19.4 Timed Access Protection

The WDT is a critical system safeguard, and is protected against accidental accesses that would enable, disable, or reset the watchdog timer.

The timed-access protection requires software to write two specific values to the timed-access register $WDTn\_RST\_wdt\_rst$ during consecutive instruction cycles. This simultaneously resets the WDT and unlocks access to $WDTn\_CTRL\_wdt\_en$ for a
period of one PCLK cycle. Non-consecutive writes to \textit{WDTn\_RST} will not cause a reset of the WDT. Attempts to modify \textit{WDTn\_CTRL} after the one PCLK cycle window will be ignored.

1. Write \textit{WDTn\_RST}: 0x000000A5
2. Write \textit{WDTn\_RST}: 0x0000005A

### 19.5 Enabling the Watchdog Timer

The watchdog timers are free running and require a protected sequence of writes to enable the watchdog timers to prevent an unintended reset during the enable process.

#### 19.5.1 Enable sequence

1. Write \textit{WDTn\_RST}: 0x000000A5
2. Write \textit{WDTn\_RST}: 0x0000005A
3. Set \textit{WDTn\_CTRL} \textit{wdt\_en} to 1

### 19.6 Disabling the Watchdog Timer

The watchdog timers can be disabled by software manually or by the microcontroller automatically as shown below.

#### 19.6.1 Manual Disable

Setting \textit{WDTn\_CTRL} \textit{wdt\_en} to 0 disables the watchdog timer.

#### 19.6.2 Automatic Disable

A power-on-reset (POR) event automatically disables the watchdog timers by setting \textit{WDTn\_CTRL} \textit{wdt\_en} to 0.

\textit{Note: The watchdog timers remain enabled during all other types of reset.}

### 19.7 Resetting the Watchdog Timer

To prevent a WDT interrupt, reset, or both, application software must use the timed-access procedure above to reset the WDT prior to an interrupt or reset timeout occurring.

#### 19.7.1 Reset Sequence

1. Write \textit{WDTn\_RST}: 0x000000A5
2. Write \textit{WDTn\_RST}: 0x0000005A

### 19.8 Detection of a Watchdog Reset Event

There are multiple hardware and software events that can cause a system reset. If the watchdog timer is being used, software should check the \textit{WDTn\_CTRL\_rst\_flag} to determine if the reset was the result of a watchdog reset. Application software is responsible for taking appropriate actions if a watchdog reset occurred.

### 19.9 Registers

See \textit{Table 3-1: APB Peripheral Base Address Map} for the Watchdog Timer’s Peripheral Base Address.

All Watchdog Timer Registers are reset to 0 on a POR and unaffected by other resets.

\textit{Table 19-2: Watchdog Timer Register Offsets, Names and Descriptions}

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>\textit{WDTn_CTRL}</td>
<td>Watchdog Timer 0 Control Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>\textit{WDTn_RST}</td>
<td>Watchdog Timer 0 Reset Register</td>
</tr>
</tbody>
</table>
## 19.10 Register Details

### Table 19-3: Watchdog Timer Control Register

<table>
<thead>
<tr>
<th>Watchdog Timer Control Register</th>
<th>WDTn_CTRL</th>
<th>[0x0000]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
<td>Reset</td>
</tr>
<tr>
<td>31</td>
<td>rst_flag</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td>30:12</td>
<td>-</td>
<td>RO</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>rst_en</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>int_en</td>
<td>R/W</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>int_flag</td>
<td>R/W1C</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>wdt_en</td>
<td>R/W</td>
<td>0</td>
</tr>
</tbody>
</table>
### Watchdog Timer Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:4</td>
<td>rst_period</td>
<td>R/W</td>
<td>0</td>
<td><strong>WDT Reset Period</strong>&lt;br&gt;Sets the number of PCLK cycles until a system reset occurs if the watchdog timer is not reset. This field is set to 0 on a POR and is not affected by other resets.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xF: (2^{16} \times t_{PCLK})&lt;br&gt;0xE: (2^{17} \times t_{PCLK})&lt;br&gt;0xD: (2^{18} \times t_{PCLK})&lt;br&gt;0xC: (2^{19} \times t_{PCLK})&lt;br&gt;0xB: (2^{20} \times t_{PCLK})&lt;br&gt;0xA: (2^{21} \times t_{PCLK})&lt;br&gt;0x9: (2^{22} \times t_{PCLK})&lt;br&gt;0x8: (2^{23} \times t_{PCLK})&lt;br&gt;0x7: (2^{24} \times t_{PCLK})&lt;br&gt;0x6: (2^{25} \times t_{PCLK})&lt;br&gt;0x5: (2^{26} \times t_{PCLK})&lt;br&gt;0x4: (2^{27} \times t_{PCLK})&lt;br&gt;0x3: (2^{28} \times t_{PCLK})&lt;br&gt;0x2: (2^{29} \times t_{PCLK})&lt;br&gt;0x1: (2^{30} \times t_{PCLK})&lt;br&gt;0x0: (2^{31} \times t_{PCLK})</td>
</tr>
<tr>
<td>3:0</td>
<td>int_period</td>
<td>R/W</td>
<td>0</td>
<td><strong>WDT Interrupt Period</strong>&lt;br&gt;Sets the number of PCLK cycles until a watchdog timer interrupt is generated. This field is set to 0 on a POR and is not affected by other resets.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xF: (2^{16} \times t_{PCLK})&lt;br&gt;0xE: (2^{17} \times t_{PCLK})&lt;br&gt;0xD: (2^{18} \times t_{PCLK})&lt;br&gt;0xC: (2^{19} \times t_{PCLK})&lt;br&gt;0xB: (2^{20} \times t_{PCLK})&lt;br&gt;0xA: (2^{21} \times t_{PCLK})&lt;br&gt;0x9: (2^{22} \times t_{PCLK})&lt;br&gt;0x8: (2^{23} \times t_{PCLK})&lt;br&gt;0x7: (2^{24} \times t_{PCLK})&lt;br&gt;0x6: (2^{25} \times t_{PCLK})&lt;br&gt;0x5: (2^{26} \times t_{PCLK})&lt;br&gt;0x4: (2^{27} \times t_{PCLK})&lt;br&gt;0x3: (2^{28} \times t_{PCLK})&lt;br&gt;0x2: (2^{29} \times t_{PCLK})&lt;br&gt;0x1: (2^{30} \times t_{PCLK})&lt;br&gt;0x0: (2^{31} \times t_{PCLK})</td>
</tr>
<tr>
<td>Watchdog Timer Reset Register</td>
<td>WDTn_RST</td>
<td>[0x0004]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----------</td>
<td>----------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits</td>
<td>Register Field</td>
<td>Access</td>
<td>Reset</td>
<td>Description</td>
</tr>
<tr>
<td>31:8</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field from its reset value.</td>
</tr>
<tr>
<td>7:0</td>
<td>wdt_rst</td>
<td>R/W</td>
<td>-</td>
<td><strong>Reset Register</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Writing the watchdog counter reset sequence to this register resets the watchdog counter. The following is the required reset sequence to reset the watchdog and prevent a watchdog timer interrupt or watchdog system reset. This field is set to 0 on a POR and is not affected by other resets.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Write <code>WDTn_RST</code>: 0x000000A5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Write <code>WDTn_RST</code>: 0x0000005A</td>
</tr>
</tbody>
</table>
### 20. 1-Wire Master (OWM)

The device provides a 1-Wire master (OWM) that you can use to communicate with one or more external 1-Wire slave devices using a single-signal, combined clock, data protocol. The OWM is contained in the OWM module. The OWM module handles the lower-level details (including timing and drive modes) required by the 1-Wire protocol, allowing the CPU to communicate over the 1-Wire bus at a logical data level.

The OWM provides the following features:

- Flexible, 1-Wire timing generation (required 1MHz timing base) using the OWM module clock frequency, which is in turn derived from the current system clock source. You can also prescale the OWM module clock to allow proper 1-Wire timing generation using a range of base frequencies.
- Automatic generation of proper 1-Wire time slots for both standard and overdrive timing modes.
- Flexible configuration for 1-Wire line pullup modes: options for internal pullup, external fixed pullup, and optional external strong pullup are available.
- Long-line compensation and bit banging (direct firmware drive) modes.
- 1-Wire reset generation and presence-pulse detection.
- Generation of 1-Wire read and write time slots for single-bit and eight-bit byte transmissions.
- Search ROM Accelerator (SRA) mode, which simplifies the generation of multiple-bit time slots and discrepancy resolution required when completing the Search ROM function to determine the IDs of multiple, unknown 1-Wire slaves on the bus.
- Transmit data completion, received data available, presence pulse detection, and 1-Wire line-error condition interrupts.

For more information about the Maxim 1-Wire protocol and supporting devices, refer to the following resources:

- **AN937: The Book of iButton® Standards**
  - www.maximintegrated.com/AN937
- **AN1796: Overview of 1-Wire Technology and Its Use**
  - www.maximintegrated.com/AN1796
- **AN187: 1-Wire Search Algorithm**
  - www.maximintegrated.com/AN187

### 20.1 Instances

There is one instance of this peripheral.

---

*iButton is a registered trademark of Maxim Integrated Products, Inc.*
20.2 Pins and Configuration

The OWM pin mapping is shown in Table 20-1.

**Table 20-1: OWM Pin to Alternate Function Mapping**

<table>
<thead>
<tr>
<th>Alternate Function</th>
<th>Alternate Function 1</th>
<th>Alternate Function 2</th>
<th>Alternate Function 3</th>
<th>Pin Name</th>
<th>Direction</th>
<th>Signal Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWM_IO</td>
<td>SPIXF_SDIO2</td>
<td>TMR4</td>
<td>P0.4</td>
<td>I/O</td>
<td>1-Wire I/O</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPIXR_SDIO2</td>
<td>QSPI0_SDIO2</td>
<td>TMR0</td>
<td>P0.12</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AINO/AINOP</td>
<td>QSPI1_SS0</td>
<td>TMR4</td>
<td>P0.16</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PCM_LRCLK</td>
<td>QSPI2_SS0</td>
<td>TMR0</td>
<td>P0.24</td>
<td>I/O</td>
<td></td>
</tr>
<tr>
<td>OWM_PE</td>
<td>SPIXF_SDIO3</td>
<td>TMR5</td>
<td>P0.5</td>
<td>O</td>
<td>Pullup Enable Output</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPIXR_SDIO3</td>
<td>QSPI0_SDIO3</td>
<td>TMR1</td>
<td>P0.13</td>
<td>O</td>
<td>Pullup Enable Output</td>
</tr>
<tr>
<td></td>
<td>AIN1/AINOP</td>
<td>QSPI1_MOSI/SDIO0</td>
<td>TMR5</td>
<td>P0.17</td>
<td>O</td>
<td>Pullup Enable Output</td>
</tr>
<tr>
<td></td>
<td>PCM_DOUT</td>
<td>QSPI2_MOSI/SDIO0</td>
<td>TMR1</td>
<td>P0.25</td>
<td>O</td>
<td>Pullup Enable Output</td>
</tr>
</tbody>
</table>

20.2.1 Pin Configuration

Perform the following steps to configure the GPIO for OWM peripheral usage:

3. Enable the alternate function mode for pins P1.30 and P1.31 by setting GPIO1_EN[30:31] to 0.
4. Set alternate function 1 (AF1) by setting GPIO1_AF_SEL[30:31] to 0.

20.2.2 1-Wire I/O (OWM_IO)

The 1-Wire IO signal is a bidirectional I/O that is used to directly drive the external 1-Wire bus. As described in the 1-Wire interface specification, 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.

20.2.3 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.

20.2.4 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 20-1, below, where \( f_{PCLK} = \frac{f_{SYSCLK}}{2} \).

**Equation 20-1: OWM 1MHz Clock Frequency**

\[
 f_{owmclk} = 1\text{MHz} = \frac{f_{PCLK}}{\text{OWM_CLK_DIV_1US.divisor}}
\]
If the system clock is set to 120MHz, \( f_{PCLK} = 60\text{MHz} \), the \textit{OWM\_CLK\_DIV\_1US\_divisor} field should be set to 60 as shown in Equation 20-2.

\[
\text{Equation 20-2: OWM Clock Divisor for } f_{SYS\_CLK} = 120\text{MHz} \\
\text{OWM\_CLK\_DIV\_1US\_divisor} = \frac{60\text{MHz}}{1\text{MHz}} = 60
\]

### 20.3 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.

#### 20.3.1 Networking Layers

In the \textit{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 Presentation layer would correspond to higher-level application software functions (such as library layers) that implement communication protocols using the 1-Wire layers as a foundation.

#### 20.3.1.1 Bus Interface (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 20-1 and avoids bus contention if more than one device attempts to drive the 1-Wire bus at the same time.

\textit{Figure 20-1: 1-Wire Signal Interface}

![1-Wire Signal Interface Diagram]

#### 20.3.1.2 Reset, Presence Detect, and Data Transfer (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

Maxim Integrated
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.

The OWM begins each communication sequence by sending a reset pulse as shown in Figure 20-2. 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 20-2: 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 no 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.

### 20.3.1.2.1 Read and Write Time Slots

**OWM Write Time Slot**

All 1-Wire bit time slots are initiated by the 1-Wire bus master and begin with a single falling edge. There is no indication given by the beginning of a time slot whether a read bit or write bit operation is intended, as the time slots all begin in the same manner. Rather, the 1-Wire command protocol enforces agreement between the OWM and slave as to which time slots are used for bit writes and which time slots are used for bit reads.

When multiple bits of a value are transmitted (or read) in sequence, the least significant bit of the value is always sent or received first. The 1-Wire bus is a half-duplex bus, so data is transmitted in only one direction (from master to slave or from slave to master) at any given time.

As shown in Figure 20-3, 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.
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.

**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 20-3. 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 20-4 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.
20.3.1.2 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.

20.3.1.3 ROM Commands (Network Layer)

Following the initial 1-Wire reset pulse on the bus, all slave 1-Wire devices are active, which means that they are monitoring the bus for commands. Because the 1-Wire bus can have multiple slave devices present on the bus at any time, the OWM must go through a process (defined by the 1-Wire command protocol) to activate only the 1-Wire slave device that it intends to communicate with and deactivate all others. This is the purpose of the ROM commands (network layer) shown in Table 20-2.

<table>
<thead>
<tr>
<th>ROM Command</th>
<th>Hex Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read ROM</td>
<td>0x33</td>
</tr>
<tr>
<td>Match ROM</td>
<td>0x55</td>
</tr>
<tr>
<td>Search ROM</td>
<td>0xF0</td>
</tr>
<tr>
<td>Skip ROM</td>
<td>0xCC</td>
</tr>
<tr>
<td>Overdrive Skip ROM</td>
<td>0x3C</td>
</tr>
<tr>
<td>Overdrive Match ROM</td>
<td>0x69</td>
</tr>
<tr>
<td>Resume Communication</td>
<td>0xA5</td>
</tr>
</tbody>
</table>

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.

*Figure 20-5* 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.

*Table 20-3* provides a detailed description of each of the ROM ID fields.
### Table 20-3: 1-Wire Slave Device ROM ID Field

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family code</td>
<td>0-7</td>
<td>This 8-bit value is used to identify the type of a 1-Wire slave device.</td>
</tr>
<tr>
<td>Unique ID</td>
<td>8-55</td>
<td>This 48-bit value is factory-programmed to give each 1-Wire slave device (within a given family code group) a globally unique identifier.</td>
</tr>
<tr>
<td>CRC</td>
<td>56-63</td>
<td>This is the 8-bit, 1-Wire CRC as defined in the Book of iButton Standards. The CRC is generated using the polynomial (x^8 + x^5 + x^4 + 1).</td>
</tr>
</tbody>
</table>

*Note: For certain operations that consist only of writing from the OWM to the slave, it is technically possible for the master to communicate with more than one slave at a time on the same 1-Wire bus. For this to work, the exact same data must be transmitted to all slave devices, and any values read back from the slaves must either be identical as well or must be disregarded by the master device (because different slaves can attempt to transmit different values). The descriptions below assume, however, that the master is communicating with only one slave device at a time because this is the method that is normally used.*

As explained above, the ROM ID contents play an important role in addressing and selecting devices on the 1-Wire bus. All devices except one are in an idle/inactive state after the Match ROM command or the Search ROM command is executed. They return to the active state only after receiving a 1-Wire reset pulse.

Devices with overdrive capability are distinguished from others by their family code and two additional ROM commands: Overdrive Skip ROM and Overdrive Match ROM. The first transmission of the ROM command itself takes place at the normal speed that is understood by all 1-Wire devices. After a device with overdrive capability is addressed and set into overdrive mode (that is, after the appropriate ROM command is received), further communication to that device must occur at overdrive speed. Because all deselected devices remain in the idle state as long as 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.

#### 20.3.2 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 proceed to the Transport layer.

#### 20.3.3 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.
20.3.4 Match ROM and Overdrive Match ROM Commands

The Match ROM command is used by the OWM to select one and only one slave 1-Wire device when the ROM ID of the device has already been determined. When transmitting this command, the master sends the command byte (that is, 55h for standard speed and 69h for overdrive speed) and then sends the entire 64-bit ROM ID for the device selected, least significant bit first.

During the transmission of the ROM ID by the master, all slave devices monitor the bus. As each bit is transmitted, each of the slave devices compares it against the corresponding bit of their ROM ID. If the bits match, the slave device continues to monitor the bus. If the bits do not match, the slave device transitions to the inactive state (waiting for a 1-Wire reset) and stops monitoring the bus.

At the end of the transmission, at most one slave device is active, which is the slave device whose ROM ID matched the ROM ID that was transmitted. All other slave devices are inactive. Communication then proceeds to the Transport layer for the device that was selected.

The Overdrive Match ROM command operates in an identical manner except that it also causes the slave device that is selected by the command to shift communication speed from standard speed to overdrive speed. The Overdrive Match ROM command byte (69h) and the 64-bit ROM ID bits are transmitted at standard speed. All subsequent communication is sent at overdrive speed.

20.3.5 Search ROM Command

The Search ROM command allows the OWM to determine the ROM ID values of all 1-Wire slave devices connected to the bus using an iterative search process. Each execution of the Search ROM command reveals the ROM ID of one slave device on the bus.

The operation of the Search ROM command resembles a combination of the Read ROM and Match ROM commands. First, all slaves on the bus transmit the least significant bit (Bit 0) of their ROM IDs. Next, all slaves on the bus transmit a complement of the same bit. By analyzing the two bits received, the master can determine whether the Bit 0 values were 0 for all slaves, 1 for all slaves, or a combination of the two. Next, the master selects which slaves remain activated for the next step in the Search ROM process by transmitting the Bit 0 value for the slaves it selects. All slaves whose Bit 0 matches the value transmitted by the master remain active, while slaves with a different Bit 0 value go to the inactive state and do not participate in the remainder of the Search ROM command.

Next, the same process is followed for Bit 1, then Bit 2, and so on until the 63rd bit (most significant bit) of the ROM ID is transmitted. At this point only one slave device remains active, and the master can either continue with communication at the Transport layer or issue a 1-Wire reset pulse to go back for another pass at the Search ROM command.

The *Book of iButton Standards* goes into more detail about the process that is used by the master to obtain ROM IDs of all devices on the 1-Wire bus using multiple executions of the Search ROM command. The algorithm resembles a binary tree search and is used regardless of how many devices are on the bus.

There is no overdrive equivalent version of the Search ROM command.

20.3.6 Search ROM Accelerator Operation

To allow the Search ROM command to process more quickly, the OWM module provides a special accelerator mode for use with the Search ROM command. This mode is activated by setting `OWM_CTRL_STAT.sra_mode` to 1.

When this mode is active, ROM IDs being processed by the Search ROM command are broken into 4-bit nibbles where the current 64-bit ROM ID varies with each pass through the search algorithm. Each 4-bit processing step is initiated by writing the 4-bit value to `OWM_DATA.tx_rx`. This causes the generation of twelve 1-Wire time slots by the OWM as each bit in the 4-bit value (starting with the LSB) results in a read of two bits (all active slaves transmitting bit N of their ROM IDs, then all active slaves transmitting the complement of bit N of their ROM ID), and then a write of a single bit by the OWM.
After the 4-bit processing stage is complete, the return value loaded into `OWM_DATA.tx_rx` consists of 8 bits. The low nibble (bits 0 through 3) contains the four discrepancy flags: one for each ID bit processed. If the discrepancy bit is set to 1, it means that either two slaves with differing ID bits in that position both responded (the 2 bits read were both zero), or that no slaves responded (the 2 bits read were both 1). If the discrepancy bit is set to 0, then the 2 bits read were complementary (either 0, 1 or 1, 0) meaning that there was no bus conflict.

In this way, at each step in the Search ROM command, the master either follows the ID of the responding slaves or deselects some of the slaves on the bus in case of a conflict. By the time the end of the 64-bit ROM ID is reached (the sixteenth 4-bit group processing step), the combination of all bits from the high nibbles of the received data are equal to the ROM ID of one of the slaves remaining on the bus. Subsequent passes through the Search ROM algorithm are used to determine additional slave ROM ID values until all slaves are identified. Refer to the Book of iButton Standards for a detailed explanation of the search function and possible variants of the search algorithm applicable to specific circumstances.

### 20.3.7 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.
20.4 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 handles 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

20.4.1 Resetting the OWM

The first step in any 1-Wire communication sequence is to reset the 1-Wire bus. To direct the OWM module to complete a 1-Wire reset, write OWM_CTRL_STAT.start_ow_reset to 1. This generates a reset pulse and checks for a replying presence pulse from any connected slave devices.

Once the reset time slot is complete, the OWM_CTRL_STAT.start_ow_reset field is automatically cleared to zero. Then, the interrupt flag OWM_INTFL.ow_reset_done is set to 1 by hardware. This flag must be cleared by writing a 1 bit to the flag.

If a presence pulse is detected on the 1-Wire bus during the reset sequence (that should normally be the case unless no 1-Wire slave devices are present on the bus), the OWM_CTRL_STAT.presence_detect flag is also set to 1. This flag does not result in the generation of an interrupt.

20.5 1-Wire Data Reads

20.5.1 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 `OMW_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 `OMW_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 `OMW_DATA` register initiates the read of the bit on the 1-Wire bus.
3. Once the single-bit transmission is complete, hardware sets the interrupt flag `OMW_INTFL.tx_data_empty` to 1. This flag (that triggers an OWM module interrupt if `OMW_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 `OMW_INTFL.rx_data_ready` is set to 1. If `OMW_INTEN.rx_ready` is set to 1, an OWM module interrupt occurs.
5. Read `OMW_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.

### 20.5.2 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 `OMW_CFG.single_bit_mode` to 0. This setting causes the OWM to transmit/receive in the default 8-bit mode.
2. Write `OMW_DATA.tx_rx` to 0xFFh.
3. Once the 8-bit transmission completes, hardware sets the interrupt flag `OMW_INTFL.tx_data_empty` to 1. This flag (that triggers an OWM module interrupt if `OMW_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 `OMW_INTFL.rx_data_ready` is set to 1. If `OMW_INTEN.rx_ready` is set to 1, an OWM module interrupt occurs.
5. Read `OMW_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.

### 20.6 Registers

See [Table 3-1: APB Peripheral Base Address Map](#) for this peripheral/module's base address. If multiple instances are provided, each will have a unique base address. Unless specified otherwise, all fields are reset on a system reset, soft reset, POR, and the peripheral-specific reset, if applicable.

**Table 20-4: OWM Register Summary**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td>OWM_CFG</td>
<td>OWM Configuration Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td>OWM_CLK_DIV_1US</td>
<td>OWM Clock Divisor Register</td>
</tr>
<tr>
<td>[0x0008]</td>
<td>OWM_CTRL_STAT</td>
<td>OWM Control/Status Register</td>
</tr>
<tr>
<td>[0x000C]</td>
<td>OWM_DATA</td>
<td>OWM Data Buffer Register</td>
</tr>
<tr>
<td>[0x0010]</td>
<td>OWM_INTFL</td>
<td>OWM Interrupt Flag Register</td>
</tr>
<tr>
<td>[0x0014]</td>
<td>OWM_INTEN</td>
<td>OWM Interrupt Enable Register</td>
</tr>
</tbody>
</table>
## 20.7 Register Details

**Table 20-5: OWM Configuration Register**

<table>
<thead>
<tr>
<th>OWM Configuration Register</th>
<th>OWM_CFG</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Field</td>
<td>Access</td>
</tr>
</tbody>
</table>
| 31:8           | -                   | R/W    | 0     | Reserved for Future Use  
Do not modify this field. |
| 7              | int_pullup_enable   | R/W    | 0     | Internal Pullup Enable |
|                |                     |        |       | Set this field to enable the internal pullup resistor.  
0: Internal pullup disabled.  
1: Internal pullup enabled. |
| 6              | overdrive           | R/W    | 0     | Overdrive Enable |
|                |                     |        |       | Set this field to 1 to enable overdrive mode for 1-Wire communications. Clearing this field sets 1-Wire communications to standard speed.  
0: Overdrive mode disabled, standard speed mode.  
1: Overdrive mode enabled. |
| 5              | single_bit_mode     | R/W    | 0     | Bit Mode Enable |
|                |                     |        |       | When set to 1, only a single bit at a time is transmitted and received (LSB of OWM_DATA) rather than the whole byte.  
0: Byte mode enabled, single bit mode disabled.  
1: Single bit mode enabled, byte mode disabled. |
| 4              | ext_pullup_enable   | R/W    | 0     | External Pullup Enable |
|                |                     |        |       | Enables external FET pullup when the 1-Wire master is idle. FET is designed to pull the wire high regardless of its enable state (that is, high or low). Idle means the 1-Wire master is idle, and there are no 1-Wire accesses in progress.  
0: External pullup pin is not driven to high.  
1: External pullup pin is driven high when the 1-Wire bus is idle, actively pulling the 1-Wire IO high. |
| 3              | ext_pullup_mode     | R/W    | 0     | External Pullup Mode |
|                |                     |        |       | Provides an extra output to control an external pullup. For long wires, a pullup resistor strong enough to pull the wire high in a reasonable amount of time might need to be so strong that it would be difficult to drive the line low. In this case, implement an external FET to actively drive the wire high for a brief amount of time. Then, let the resistor keep the line high. |
| 2              | bit_bang_en         | R/W    | 0     | Bit-Bang Mode Enable |
|                |                     |        |       | Enable bit-bang control of the I/O pin. If this bit is set to 1, OWM_CTRL_STAT.bit_bang_oe controls the state of the I/O pin.  
0: Bit-bang mode disabled.  
1: Bit-bang mode enabled. |
| 1              | force_pres_det      | R/W    | 1     | Presence Detect Force |
|                |                     |        |       | Setting this bit to 1 drives the OWM_IO pin low during presence detection. Use this bit field to prevent a large number of 1-Wire slaves on the bus from all responding at different times, which might cause ringing. When this bit is set to 1, the OWM_CTRL_STAT.presence_detect bit is always set as the result of a 1-Wire reset even if no slave devices are present on the bus.  
0: OWM_IO pin floats during presence detection portion of 1-Wire reset.  
1: OWM_IO pin is driven low during presence detection portion of 1-Wire reset. |
### OWM Configuration Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>long_line_mode</td>
<td>R/W</td>
<td>0</td>
<td><strong>Long Line Mode Enable</strong>&lt;br&gt;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.&lt;br&gt;0: Standard operation for lines less than 40 meters.&lt;br&gt;1: Long line mode enabled, see description above.</td>
</tr>
</tbody>
</table>

### OWM Clock Divisor Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>-</td>
<td>R</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>7:0</td>
<td>divisor</td>
<td>R/W</td>
<td>0</td>
<td><strong>OWM Clock Divisor</strong>&lt;br&gt;Divisor for the OWM peripheral clock. The target is to achieve a 1MHz clock. See the Clock Configuration section for details.&lt;br&gt;0x00: OWM clock disabled.&lt;br&gt;0x01: $f_{owmclk} = \frac{f_{PCLK}}{1}$&lt;br&gt;0x02: $f_{owmclk} = \frac{f_{PCLK}}{2}$&lt;br&gt;...&lt;br&gt;0xFF: $f_{owmclk} = \frac{f_{PCLK}}{255}$</td>
</tr>
</tbody>
</table>

### OWM Control/Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:6</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>5</td>
<td>presence_detect</td>
<td>RO</td>
<td>0</td>
<td><strong>Presence Detect Flag</strong>&lt;br&gt;Set to 1 when a presence pulse is detected from one or more slaves during the 1-Wire reset sequence.&lt;br&gt;0: No presence detect pulse during previous 1-Wire reset sequence.&lt;br&gt;1: Presence detect pulse on bus during previous 1-Wire reset sequence.</td>
</tr>
<tr>
<td>4</td>
<td>od_spec_mode</td>
<td>RO</td>
<td>0</td>
<td><strong>Overdrive Spec Mode</strong>&lt;br&gt;Returns the version of the overdrive spec.</td>
</tr>
<tr>
<td>3</td>
<td>ow_input</td>
<td>RO</td>
<td>-</td>
<td><strong>OWM_IN State</strong>&lt;br&gt;Returns the current logic level on the OWM_IO pin.&lt;br&gt;0: OWM_IO pin is low.&lt;br&gt;1: OWM_IO pin is high.</td>
</tr>
</tbody>
</table>
### OWM Control/Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>bit_bang_oe</td>
<td>R/W</td>
<td>0</td>
<td><strong>OWM Bit-Bang Output</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When bit-bang mode is enabled ($OWM CFG.bit_bang_en = 1$), this bit sets the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>state of the OWM_IO pin. Setting this bit to 1 drives the OWM_IO pin low.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting this bit to 0 releases the line, allowing the OWM_IO pin to be pulled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>high by the pullup resistor or held low by a slave device.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: OWM_IO pin floating.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Drive OWM_IO pin to low state.</td>
</tr>
<tr>
<td>1</td>
<td>sra_mode</td>
<td>R/W</td>
<td>0</td>
<td><strong>Search ROM Accelerator Enable</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Enable Search ROM Accelerator mode. This mode is used to identify slaves and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>their addresses that are attached to the 1-Wire bus.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Search ROM accelerator mode disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Search ROM accelerator mode enabled.</td>
</tr>
<tr>
<td>0</td>
<td>start_ow_reset</td>
<td>R/W</td>
<td>0</td>
<td><strong>Start 1-Wire Reset Pulse</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write 1 to start a 1-Wire reset sequence. Automatically cleared by the OWM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>hardware when the reset sequence is complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: 1-Wire reset sequence complete or inactive.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Start a 1-Wire reset sequence.</td>
</tr>
</tbody>
</table>

### OWM Data Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td></td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>7:0</td>
<td>tx_rx</td>
<td>R/W</td>
<td>0</td>
<td><strong>OWM Data Field</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Writing to this field sets the transmit data and initiates a 1-Wire data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>transmit cycle. Reading from this field returns the data received by the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>master during the last 1-Wire data transmit cycle.</td>
</tr>
</tbody>
</table>

### OWM Interrupt Flag Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:5</td>
<td></td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>4</td>
<td>line_low</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>Line Low Flag</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>If this flag is set, the OWM_IO pin was in a low state. Write 1 to clear this</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>flag.</td>
</tr>
<tr>
<td>3</td>
<td>line_short</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>Line Short Flag</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The OWM hardware detected a short on the OWM_IO pin. Write 1 to clear this</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>flag.</td>
</tr>
<tr>
<td>2</td>
<td>rx_data_ready</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>RX Data Ready</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Data received from the 1-Wire bus and is available in the $\text{OWM DATA.}\text{tx_rx}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>field. Write 1 to clear this flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: RX data not available.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Data received and is available in the $\text{OWM DATA.}\text{tx_rx}$ field.</td>
</tr>
</tbody>
</table>
### OWM Interrupt Flag Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>tx_data_empty</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>TX Empty</strong>&lt;br&gt; 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 <code>OWM_DATA.tx_rx</code> field has not completed transmission. 1: Data in the <code>OWM_DATA.tx_rx</code> field was transmitted.</td>
</tr>
<tr>
<td>0</td>
<td>ow_reset_done</td>
<td>R/W1C</td>
<td>0</td>
<td><strong>Reset Complete</strong>&lt;br&gt; This flag is set when a 1-Wire reset sequence completes. To start a 1-Wire reset sequence, see <code>OWM_CTRL_STAT.start_ow_reset</code>. Write 1 to clear this flag. 0: 1-Wire reset sequence not complete or bus idle. 1: 1-Wire reset sequence complete.</td>
</tr>
</tbody>
</table>

### OWM Interrupt Enable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Field</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:5</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt; Do not modify this field.</td>
</tr>
<tr>
<td>4</td>
<td>line_low</td>
<td>R/W</td>
<td>0</td>
<td><strong>Line Low Interrupt Enable</strong>&lt;br&gt; I/O pin low detected interrupt enable.    0: Interrupt disabled. 1: Interrupt enabled.</td>
</tr>
<tr>
<td>3</td>
<td>line_short</td>
<td>R/W</td>
<td>0</td>
<td><strong>Line Short Interrupt Enable</strong>&lt;br&gt; I/O pin short detected interrupt enable. 0: Interrupt disabled. 1: Interrupt enabled.</td>
</tr>
<tr>
<td>2</td>
<td>rx_data_ready</td>
<td>R/W</td>
<td>0</td>
<td><strong>Receive Data Ready Interrupt Enable</strong>&lt;br&gt; RX data ready interrupt enable. 0: Interrupt disabled. 1: Interrupt enabled.</td>
</tr>
<tr>
<td>1</td>
<td>tx_data_empty</td>
<td>R/W</td>
<td>0</td>
<td><strong>Transmit Data Empty Interrupt Enable</strong>&lt;br&gt; TX data empty interrupt enable. 0: Interrupt disabled. 1: Interrupt enabled.</td>
</tr>
<tr>
<td>0</td>
<td>ow_reset_done</td>
<td>R/W</td>
<td>0</td>
<td><strong>1-Wire Reset Sequence Complete Interrupt Enable</strong>&lt;br&gt; 1-Wire reset sequence completed. 0: Interrupt disabled. 1: Interrupt enabled.</td>
</tr>
</tbody>
</table>
21. USB 2.0 High-Speed (USBHS) Host Interface with PHY

The microcontroller includes one Universal Serial Bus (USB) Host communications peripheral with a USB physical interface (PHY). The USB Host is USB 2.0 High-Speed (USBHS) compliant, capable of transfers at 480Mbps. It supports Host mode with 12 USB buffers called endpoints.

The following features are supported:

- USB Device Mode
- USB 2.0 Full Speed (FS) 12Mbps transfers
- USB 2.0 Hi-Speed (HS) 480Mbps transfers
- Bulk transfers
- Isochronous transfers
- 11 endpoints plus Endpoint 0, each with dedicated FIFOs
- Packet splitting and combining
- High bandwidth IN and OUT Isochronous endpoints

Each endpoint has an associated FIFO with the following sizes:

- Endpoint 0 FIFO: 64 bytes deep
- Endpoints 1 through 7 FIFOs: 512 bytes deep
- Endpoints 8 and 9 FIFOs: 2048 bytes deep
- Endpoints 10 and 11 FIFOs: 4096 bytes deep

Supported interrupts include:

- Interrupts for each IN endpoint from Endpoint 0 to Endpoint 11
- Interrupts for each OUT endpoint from Endpoint 1 to Endpoint 11
- Start of Frame (SOF)
- RESET bus state
- RESUME bus state
- SUSPEND Mode bus state
- STALL sent
- Control byte received
- Control transfer ended early
- Packet transmitted
- Packet received
- Data underrun
- Data overrun
- Invalid token received
- Empty data packet sent

This chapter includes a simplified description of USB bus states. Refer to the USB 2.0 Specification for a complete description of USB operation.

The USB device hardware behavior is controlled by the internal serial interface engine (SIE). The SIE is a small control processor that manages the USB port’s behavior. When referring to behavior of the USB hardware, it is the SIE doing the work.

21.1 Instances

There is one instance of the peripheral.
21.2 USBHS Bus Signals

A USB cable connects a USB Host, which controls the transfer, and a USB Device, which is controlled by the Host. The USBHS Peripheral is a USB Device. A USB cable has four conductors (three hardware signals plus ground). These signals can be duplicated more than once in a physical USB cable. The signals in a USB cable are as follows:

- **D+ (DPLUS):** Positive line of the differential data pair.
- **D- (DMINUS):** Negative line of the differential data pair.
- **VBUS:** Bus voltage supplied by Host.
- **Ground**

When a USB Device is attached to a USB Host, the USB Host identifies the speed of the USB Device by the presence of a pullup resistor on either D+ or D-. The Host then begins the Enumeration sequence. The Enumeration sequence allows the Host to identify the type and characteristics of the Device attached to the Host so that it can load the proper drivers for the Device. Enumeration always uses Endpoint 0. The Host requests and reads from the Device the contents of the USB Endpoint Descriptor Table that tells the Host everything it needs to know about the capabilities of the USB Device. The Host then assigns the USB Device an address, which firmware writes to the `USBHS_FADDR.faddr` bit field.

Table 21-1 shows the USB Bus states seen by the Host indicated by the differential pair.

<table>
<thead>
<tr>
<th>Bus State</th>
<th>Condition</th>
<th>D+</th>
<th>D-</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Differential 1</td>
<td>Host or Device is driving the bus</td>
<td>Hi</td>
<td>Lo</td>
<td></td>
</tr>
<tr>
<td>Differential 0</td>
<td>Host or Device is driving the bus</td>
<td>Lo</td>
<td>Hi</td>
<td></td>
</tr>
<tr>
<td>Single-Ended Zero (SE0)</td>
<td>Cable Detached</td>
<td>Lo</td>
<td>Lo</td>
<td>No Device plugged in.</td>
</tr>
<tr>
<td>Single-Ended One (SE1)</td>
<td>Illegal State</td>
<td>Hi</td>
<td>Hi</td>
<td>Illegal state. This state should never occur on a properly configured USB bus.</td>
</tr>
<tr>
<td>IDLE State, Full Speed</td>
<td>No Host or Device is driving the bus</td>
<td>Hi</td>
<td>Lo</td>
<td>USB Device is Full Speed. No activity on bus.</td>
</tr>
<tr>
<td>IDLE State, Low Speed</td>
<td>No Host or Device is driving the bus</td>
<td>Lo</td>
<td>Hi</td>
<td>USB Device is Low Speed. No activity on bus.</td>
</tr>
<tr>
<td>DISCONNECT</td>
<td>Device wants to disconnect from Host</td>
<td>Lo</td>
<td>Lo</td>
<td>Held for 2.5µs or longer.</td>
</tr>
<tr>
<td>RESET</td>
<td>Host is initiating communication with a Device</td>
<td>Lo</td>
<td>Lo</td>
<td>Held for 10ms or longer.</td>
</tr>
</tbody>
</table>

USB communication is based on the above basic conditions, which are used to generate the following states:

- **Data J State** – Same as IDLE state, but bus is actively driven by either the Host or the Device.
- **Data K State** – Opposite of J state. Bus is actively driven by the Host or the Device.
- **RESUME** – Data K State. Tells Device to exit SUSPEND mode.
- **START OF PACKET (SOP)** – Bus switches from IDLE to K state.
- **END OF PACKET (EOP)** – SE0 for two bit periods, then J state for one bit period.
- **KEEP ALIVE Signal** – EOP sent every 1 millisecond.

21.3 USBHS Device Endpoints

Each USB Device supports one or more endpoints. Endpoints serve as a source or destination for data and are supported by memory buffers. This USB controller supports 12 endpoints, each with its own set of descriptors and data buffers. These Endpoints are referenced as Endpoint 0 through Endpoint 11.
Endpoints support four different types of data transfers:

- **Control Endpoint** – Always uses Endpoint 0, this endpoint is used by the USB Host to setup the USB Device for the USB Device to receive operational status from the USB Host.
- **Interrupt Endpoints** – Used to send and receive non-time critical data to and from a USB Device. An application example is a USB keyboard or a USB mouse.
- **Bulk Endpoints** – Used to send and receive high-volume data that does not require real-time processing. An application example is a USB flash drive which transfers high volume data.
- **Isochronous Endpoints** – Used to send or receive real-time data that requires a guaranteed bandwidth to or from a Host. An application example is a video camera used for real-time video streaming.

The USBHS supports Control, Interrupt, Bulk, and Isochronous Endpoints. Per the USB 2.0 Specification, Endpoint 0 is dedicated to Control Transfers only.

Endpoint directions are always defined from a USB Host to a USB Device. OUT Endpoint 1 refers to a Device Endpoint holding data sent out from a USB Host to a USB Device. IN Endpoint 2 refers to a Device Endpoint holding data sent from a USB Device to a USB Host.

Each USBHS Data Endpoint supports the following features:

- Single or double buffered
- Programmable and flexible interrupts
- Ability to send a STALL packet to the Host to indicate an error with the data
- Ability to automatically send an ACK packet to the Host to acknowledge a successful data transfer
- Ability to send a NYET (Not Yet) packet to the Host for Hi-Speed transfers to indicate it is not yet ready to receive more data
- Configurable response to Status Stage of Control transfer

### 21.4 USBHS Reset and Clock

When a RESET state is detected on the bus, the USBHS performs the following actions:

1. Sets $USBHS\_FADDR.addr = 0$
2. Sets $USBHS\_INDEX = 0$
3. All endpoint FIFOs are flushed
4. All control and status registers are reset
5. The USB PHY is electrically disconnected from the bus
6. All endpoint interrupts are enabled
7. Generates a USB Reset IRQ

For correct operation of the USBHS interface, the system clock, $f_{SYS.CLK}$, must be no less than 32MHz.

### 21.5 USBHS SUSPEND Mode and RESUME States

When the USBHS sees no activity on the bus for 3ms, and if SUSPEND mode is allowed ($USBHS\_POWER.suspendm = 1$), then the USBHS goes into low-power SUSPEND mode. The SUSPEND status flag is set ($USBHS\_INTSIGFL.suspend = 1$), and a SUSPEND interrupt is generated if enabled ($USBHS\_INTSIGEN.suspend = 1$).

Firmware can exit SUSPEND mode by sending a RESUME state on the bus by setting the bit field $USBHS\_POWER.resume = 1$. Firmware must leave this bit set between 2ms and 15ms with 10ms being the optimal time after which firmware must clear the resume bitfield.

If the external Host generates a RESUME state on the bus, a RESUME interrupt is generated. A RESUME interrupt is not generated if the RESUME state on the bus is caused by firmware setting the $USBHS\_POWER.resume$ bit.
21.6 Packet Size

For all transfers the packet size is specified in the USBHS_INMAXP register for IN endpoints and the USBHS_OUTMAXP register for OUT endpoints. These registers specify the size of the entire transactions.

21.7 Endpoint 0 Control Transactions

Endpoint 0 (EP0) is the main control endpoint and handles all USB Standard Device Requests for control transfers. There are three types of Standard Device Requests:

1. In Zero Data Requests, all the information for the request is included in an 8-byte command.
2. In Write Requests, the command from the USBHS is followed by additional data.
3. In Read Requests, the USBHS is communicating with a USB Device that is required to send data back to the Host.

21.7.1 Endpoint 0 Error Handling

The USBHS can detect and generate interrupts for control transfers errors. It sends a STALL packet on the bus and generates an interrupt if the incorrect amount of data is transferred over the bus. This can happen under the following conditions:

1. The Host sends more data during the OUT Data phase of a write request than the amount specified in the command. This condition is detected when the Host sends an OUT token after the Data End bit USBHS_CSR0.dataend is set by firmware.
2. The Host requests more data during the IN Data phase of a read request than the amount specified in the command. This condition is detected when the Host sends an IN token after the DataEnd bit USBHS_CSR0.dataend is set by firmware.
3. The Host sends more data in an OUT data packet than the amount specified in the USBHS_OUTMAXP register.
4. The Host sends a non-zero length DATA1 packet during the STATUS phase of a read request.

An error occurs if a control transaction ends prematurely. This can happen if the USB Host enters the STATUS phase before all data has been transferred. This can also occur if a USB Host transmits a new SETUP packet before finishing the current control transaction. In both cases, the USBHS_CSR0.setupend bit is set, which generates an Endpoint 0 interrupt.

If the USBHS_CSR0.outpktrdy bit is set, this indicates that the Host has sent another SETUP packet. Firmware should then process the command in that packet.

21.8 Bulk Endpoints Operation and Options

21.8.1 Bulk IN Endpoints

A Bulk IN endpoint is used to transfer high-volume data that does not require real-time processing. Five features are available for use with a Bulk IN endpoint as shown in Table 21-2.

Table 21-2: USB Bulk IN Endpoints Options

<table>
<thead>
<tr>
<th>Bulk IN Endpoint Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double Packet Buffering</td>
<td>When the value written to the USBHS_INMAXP register is less than or equal to half the size of the FIFO allocated to the endpoint, and double packet buffering is allowed (USBHS_INCSR0.dpktbufdis = 0), double packet buffering is enabled. This allows up to two packets to be stored in the FIFO for transmission to the Host.</td>
</tr>
</tbody>
</table>
Bulk IN Endpoint Option | Description
---|---
DMA Transfers | If the DMA is enabled for a Bulk IN endpoint (USBHS_INCSRU.dmareqenab = 1), a DMA request is generated whenever the endpoint can accept another packet in its FIFO. The DMA request is terminated when the entire packet is loaded into the DMA or when the USBHS_INCSRU.inpktrdy bit field is set indicating a packet in the FIFO was transmitted. For Bulk IN endpoints, the DMA is set to DMA Request Mode 1 where a DMA request is generated each time a packet is received. The DMA completes burst transfers based on the maximum packet size for the endpoint: 512 bytes for Hi-Speed and 64 bytes for full speed. DMA burst transfers continue until the entire data block is transferred.

AutoSet | When the AutoSet feature is enabled (USBHS_INCSRU.autoset = 1) for a Bulk IN endpoint, the IN Packet Ready bit field USBHS_INCSRU.inpktrdy is automatically set when a packet of USBHS_INMAXP bytes is loaded into the FIFO.

Automatic Packet Splitting | For some USB transfers, it might be necessary to write larger amounts of data to an endpoint than you can transfer in a single USB operation. For these transfers, the USBHS supports split transactions where large data packets that are written to Bulk endpoints are split into multiple smaller packets. The necessary packet size information is set in the USBHS_INCSRU register.

Error Handling | A STALL packet is used to indicate that an endpoint has had an error. To shut down the Bulk IN endpoint transfer, set the USBHS_INCSRU.sendstall bit field. When the USBHS receives the next IN token, it then sends a STALL to the Host, sets the USBHS_INCSRU.sentstall bit field, and generates an interrupt.

21.8.2 Bulk OUT Endpoints

A Bulk OUT endpoint is used to transfer non-periodic data from the Host to the function controller. Five optional features are available for use with a Bulk OUT endpoint.

Bulk OUT Endpoint Option | Description
---|---
Double Packet Buffering | When the value written to the USBHS_OUTMAXP register is less than or equal to half the size of the FIFO allocated to the endpoint, and double packet buffering is allowed (USBHS_OUTCSRU.dpktbufdis = 0), double packet buffering is enabled. This allows storage of up to two packets in the FIFO for transmission to the Host.

DMA Transfers | DMA transfers for an OUT endpoint depend on the DMA Request Mode selected with the USBHS_OUTCSRU.dmareqmode bit field.

In DMA Request Mode 0, a DMA request is generated when a data packet is available in the OUT Endpoint FIFO. The DMA request is terminated when the last byte of the data packet is read from the OUT FIFO, or when the USBHS_OUTCSRU.outpktrdy bit is cleared indicating the OUT FIFO is empty.

In DMA Request Mode 1, the DMA request line only goes high when the packet received is of the maximum packet size set in the USBHS_OUTMAXP register. If the packet received is of some other size, a DMA request is not generated, leaving the packet in the FIFO with the USBHS_OUTCSRU.outpktrdy bit still set.

AutoClear | When the AutoClear feature is enabled (USBHS_OUTCSRU.autoclear = 1), the USBHS_OUTCSRU.outpktrdy bit is automatically cleared when a packet of USBHS_OUTMAXP bytes is unloaded from the FIFO.

Automatic Packet Combining | For some USB transfers, it might be necessary to receive larger amounts of data from an endpoint than can be received in a single USB operation. For these transfers, the USBHS supports automatically combining packets received by split transactions, where large data packets received by Bulk endpoints had been split into multiple smaller packets. The necessary packet size information is set in the USBHS_OUTMAXP register.
### Error Handling

A STALL packet is used to indicate that an endpoint has an error. To shut down the Bulk OUT endpoint transfer, set `USBHS_OUTCSR.L.sendstall = 1`. When the USBHS receives the next packet, it then sends a STALL to the Host, sets the `USBHS_OUTCSR.L.sentstall` bit, and generates an interrupt.

## 21.9 Interrupt Endpoints

### 21.9.1 Interrupt IN Endpoints

Interrupt IN endpoints use the same protocols as Bulk IN endpoints and are used the same way. Although DMA can be used, there is little benefit as Interrupt IN endpoints transfer all their data in a single packet.

One feature supported by Interrupt IN endpoints and not Bulk IN endpoints is continuous toggle of the Data-Toggle bit. This feature is enabled by setting bit `USBHS_INCSR.U.frdatatog = 1`. When continuous toggling of the Data-Toggle bit is enabled, USBHS always considers a transmitted Interrupt packet as successfully sent and toggles Data-Toggle regardless of whether an ACK was received from the Host.

### 21.9.2 Interrupt OUT Endpoints

Interrupt OUT endpoints use almost the same protocols as Bulk OUT endpoints and are used in the same way. Although DMA can be used, there is little benefit as Interrupt OUT endpoints receive all their data in a single packet.

One feature not supported by Interrupt OUT endpoints that is supported by Bulk OUT endpoints is PING flow control. Because of this, Interrupt OUT endpoints cannot respond with NYET (Not Yet) handshakes. Instead, they can only respond with ACK, NAK, or STALL.

## 21.10 Isochronous Endpoints

### 21.10.1 Isochronous IN Endpoints

An Isochronous IN endpoint is used to transfer time-sensitive but loss-tolerant data from a USB Device to a USB Host. Five optional features are available for use with an Isochronous IN endpoint as shown in Table 21-3.

#### Table 21-3: USB Isochronous IN Endpoint Options

<table>
<thead>
<tr>
<th>Isochronous IN Endpoint Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double Packet Buffering</td>
<td>When the value written to the <code>USBHS_INMAXP</code> register is less than or equal to half the size of the FIFO allocated to the endpoint, and double packet buffering is allowed (<code>USBHS_INCSR.U.dpktbufdis = 0</code>), double packet buffering is enabled. This allows the storage of up to two packets in the FIFO for transmission to the Host. This is recommended to avoid data underrun.</td>
</tr>
<tr>
<td>DMA Transfers</td>
<td>If the DMA is enabled for an endpoint (<code>USBHS_INCSR.U.dmareqenab = 1</code>), a DMA request is generated whenever the endpoint can accept another full packet in its FIFO. However, there is little benefit with Isochronous endpoints because the packets transferred are often not the maximum packet size. In this situation, the <code>USBHS_INCSR.L.underrun</code> bit would have to be checked for underrun errors after each packet.</td>
</tr>
<tr>
<td>AutoSet</td>
<td>When the AutoSet feature is enabled (<code>USBHS_INCSR.U.autoset = 1</code>), the <code>USBHS_INCSR.U.inpktrdy</code> bit is automatically set when a packet of <code>USBHS_INMAXP</code> bytes is loaded into the FIFO. However, there is little benefit with Isochronous endpoints because the packets transferred are often not the maximum packet size. In this situation, the <code>USBHS_INCSR.L.underrun</code> bit would have to be checked for underrun errors after each packet.</td>
</tr>
</tbody>
</table>
Isochronous IN Endpoint Option | Description
--- | ---
Error Handling | If an Isochronous IN endpoint receives an IN Token while the IN FIFO is empty, it creates an underrun condition. This automatically sets the `USBHS_INCSR1.underrun` bit and results in the USBHS sending a null packet to the USB Host.

If firmware is loading the IN Endpoint FIFO one packet per frame, it should check that there is room in the IN FIFO by making sure the `USBHS_INCSR1.inpktrdy` bit is cleared before loading the next packet. If this bit is set, it indicates that a data packet is still in the FIFO and has not been sent, possibly from a corrupt IN Token. This error condition must be handled by firmware, for example, firmware might flush the unsent packet, or skip the current packet.

Error Handling – High Bandwidth Isochronous IN Endpoints Only | High-bandwidth Isochronous IN endpoints can transfer three 1024-byte packets in one payload. To the USB bus, it appears to be a single packet of 3072 bytes with a data transfer rate of up to 24MBps. If a high-bandwidth isochronous data transfer is split into more than one packet but has not received enough IN tokens from the Host to send all the packets, an error condition exists. In this case, the Incomplete Split Transfer Error Status bit `USBHS_INCSR1.incomPTn`, is automatically set. This also automatically flushes the remainder of the packet from the IN FIFO. If a second packet is in the IN FIFO, it is not flushed. Because the packet was lost, the `USBHS_INCSR1.inpktrdy` bit is cleared.

21.10.2 Isochronous OUT Endpoints

An Isochronous OUT endpoint is used to transfer time-sensitive but loss-tolerant data from the Host to the function controller. Five optional features are available for use with an Isochronous OUT endpoint as shown in Table 21-4.

Table 21-4: USB Isochronous OUT Endpoint Options

<table>
<thead>
<tr>
<th>Isochronous OUT Endpoint Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double Packet Buffering</td>
<td>When the value written to the <code>USBHS_OUTMAXP</code> register is less than or equal to half the size of the FIFO allocated to the endpoint, and double packet buffering is allowed (<code>USBHS_OUTCSR1.dpktbufdis = 0</code>), double packet buffering is enabled. This allows the storage of up to two packets in the FIFO for transmission to the Host. Double packet buffering is recommended for isochronous OUT endpoints to avoid data overrun errors.</td>
</tr>
<tr>
<td>DMA Transfers</td>
<td>If the DMA is enabled for an endpoint, a DMA request is generated whenever the endpoint can accept another full packet in its FIFO. However, there is little benefit with Isochronous endpoints because the packets transferred are often not the maximum packet size. In this situation, the <code>USBHS_INCSR1.underrun</code> bit would need to be checked for underrun errors after each packet.</td>
</tr>
<tr>
<td>AutoClear</td>
<td>When the AutoClear feature is enabled (<code>USBHS_OUTCSR1.autoclear = 1</code>), the <code>USBHS_INCSR1.outpktrdy</code> bit is automatically cleared when a packet of <code>USBHS_OUTMAXP</code> bytes is unloaded from the FIFO. However, there is little benefit with Isochronous endpoints because the packets transferred are often not the maximum packet size. In this situation, the <code>USBHS_INCSR1.underrun</code> bit would need to be checked for underrun errors after each packet.</td>
</tr>
</tbody>
</table>
| Error Handling | If a packet is received from a USB Host, but the OUT FIFO is full, it creates an overrun error condition. The register bit `USBHS_OUTCSR1.overrun` is automatically set. This error condition usually means that firmware is not unloading the OUT FIFO fast enough. This error condition must be handled by firmware.

If a received packet has a CRC error the packet is stored in the OUT FIFO, and both the `USBHS_OUTCSR1.dataerror` bit and the `USBHS_OUTCSR1.outpktrdy` bit are set. This error condition must be handled by firmware.
### Isochronous OUT Endpoint Option

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Handling – High Bandwidth Isochronous OUT Endpoints Only</td>
</tr>
</tbody>
</table>

High-bandwidth Isochronous OUT endpoints can transfer three 1024-byte packets in one payload. To the USB bus, it appears to be a single packet of 3072 bytes. If a high-bandwidth Isochronous data transmission is split into more than one packet, but if less than the expected number of packets is received by the OUT endpoint, an error condition exists. In this case, the Incomplete Isochronous Packet Received Error Status bit `USBHS_OUTCSRU.incomprx` is automatically set to indicate that the data received in the OUT FIFO is incomplete. If a packet of the wrong data type is received during a high-bandwidth Isochronous OUT transaction, then the PID Error Status bit `USBHS_OUTCSRU.piderror` is automatically set.

#### 21.11 USBHS Device Registers

See Table 3-1: APB Peripheral Base Address Map for the USBHS Peripheral Base Address.

**Table 21-5: USBHS Device Register Offsets, Names, Access, and Descriptions**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register Name</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0x0000]</td>
<td><code>USBHS_FADDR</code></td>
<td>R/W</td>
<td>USBHS Device Address Register</td>
</tr>
<tr>
<td>[0x0001]</td>
<td><code>USBHS_POWER</code></td>
<td>R/W</td>
<td>USBHS Power Management Register</td>
</tr>
<tr>
<td>[0x0002]</td>
<td><code>USBHS_INTRINFL</code></td>
<td>RO</td>
<td>USBHS IN Endpoint Interrupt Status Register</td>
</tr>
<tr>
<td>[0x0004]</td>
<td><code>USBHS_INTOUTFL</code></td>
<td>RO</td>
<td>USBHS OUT Endpoint Interrupt Status Flags Register</td>
</tr>
<tr>
<td>[0x0006]</td>
<td><code>USBHS_INTRINEN</code></td>
<td>R/W</td>
<td>USBHS IN Endpoint Interrupt Enable Register</td>
</tr>
<tr>
<td>[0x0008]</td>
<td><code>USBHS_INTOUTEN</code></td>
<td>R/W</td>
<td>USBHS OUT Endpoint Interrupt Enable Register</td>
</tr>
<tr>
<td>[0x000A]</td>
<td><code>USBHS_INTSIGFL</code></td>
<td>RO</td>
<td>USBHS Signaling Interrupt Status Flags Register</td>
</tr>
<tr>
<td>[0x000B]</td>
<td><code>USBHS_INTSIGEN</code></td>
<td>R/W</td>
<td>USBHS Signaling Interrupt Enable Register</td>
</tr>
<tr>
<td>[0x000C]</td>
<td><code>USBHS_FRAME</code></td>
<td>RO</td>
<td>USBHS Frame Number Register</td>
</tr>
<tr>
<td>[0x00EE]</td>
<td><code>USBHS_INDEX</code></td>
<td>R/W</td>
<td>USBHS Endpoint and Status Register Index Register</td>
</tr>
<tr>
<td>[0x00F0]</td>
<td><code>USBHS_TESTMODE</code></td>
<td>R/W</td>
<td>USBHS Test Mode Register</td>
</tr>
<tr>
<td>[0x0100]</td>
<td><code>USBHS_INMAXP</code></td>
<td>R/W</td>
<td>USBHS IN Endpoint Maximum Packet Size Register</td>
</tr>
<tr>
<td>[0x0112]</td>
<td><code>USBHS_CSRO</code></td>
<td>R/W</td>
<td>USBHS Endpoint 0 Control Status Register (<code>USBHS_INDEX = 0</code>)</td>
</tr>
<tr>
<td>[0x0112]</td>
<td><code>USBHS_INCSRLO</code></td>
<td>R/W</td>
<td>USBHS IN Endpoint Lower Control and Status Register (<code>USBHS_INDEX = 0</code>)</td>
</tr>
<tr>
<td>[0x0132]</td>
<td><code>USBHS_INCSRUR</code></td>
<td>R/W</td>
<td>USBHS IN Endpoint Upper Control and Status Register</td>
</tr>
<tr>
<td>[0x0144]</td>
<td><code>USBHS_OUTMAXP</code></td>
<td>R/W</td>
<td>USBHS OUT Endpoint Maximum Packet Sizes Register</td>
</tr>
<tr>
<td>[0x0164]</td>
<td><code>USBHS_OUTCSRLO</code></td>
<td>R/W</td>
<td>USBHS OUT Endpoint Lower Control Status Register</td>
</tr>
<tr>
<td>[0x0174]</td>
<td><code>USBHS_OUTCSRUR</code></td>
<td>R/W</td>
<td>USBHS OUT Endpoint Upper Control Status Register</td>
</tr>
<tr>
<td>[0x0184]</td>
<td><code>USBHS_COUNTO</code></td>
<td>RO</td>
<td>USBHS Endpoint 0 IN FIFO Byte Count Register</td>
</tr>
<tr>
<td>[0x0188]</td>
<td><code>USBHS_OUTCOUNT</code></td>
<td>RO</td>
<td>USBHS Endpoint OUT FIFO Byte Count Register</td>
</tr>
<tr>
<td>[0x0200]</td>
<td><code>USBHS_FIFO0</code></td>
<td>R/W</td>
<td>USBHS FIFO for Endpoint 0 Register</td>
</tr>
<tr>
<td>[0x0202]</td>
<td><code>USBHS_FIFO1</code></td>
<td>R/W</td>
<td>USBHS FIFO for Endpoint 1 Register</td>
</tr>
<tr>
<td>[0x0204]</td>
<td><code>USBHS_FIFO2</code></td>
<td>R/W</td>
<td>USBHS FIFO for Endpoint 2 Register</td>
</tr>
<tr>
<td>[0x0206]</td>
<td><code>USBHS_FIFO3</code></td>
<td>R/W</td>
<td>USBHS FIFO for Endpoint 3 Register</td>
</tr>
<tr>
<td>[0x0208]</td>
<td><code>USBHS_FIFO4</code></td>
<td>R/W</td>
<td>USBHS FIFO for Endpoint 4 Register</td>
</tr>
<tr>
<td>[0x020A]</td>
<td><code>USBHS_FIFO5</code></td>
<td>R/W</td>
<td>USBHS FIFO for Endpoint 5 Register</td>
</tr>
<tr>
<td>[0x020C]</td>
<td><code>USBHS_FIFO6</code></td>
<td>R/W</td>
<td>USBHS FIFO for Endpoint 6 Register</td>
</tr>
<tr>
<td>[0x020E]</td>
<td><code>USBHS_FIFO7</code></td>
<td>R/W</td>
<td>USBHS FIFO for Endpoint 7 Register</td>
</tr>
<tr>
<td>[0x0210]</td>
<td><code>USBHS_FIFO8</code></td>
<td>R/W</td>
<td>USBHS FIFO for Endpoint 8 Register</td>
</tr>
<tr>
<td>[0x0212]</td>
<td><code>USBHS_FIFO9</code></td>
<td>R/W</td>
<td>USBHS FIFO for Endpoint 9 Register</td>
</tr>
<tr>
<td>[0x0214]</td>
<td><code>USBHS_FIFO10</code></td>
<td>R/W</td>
<td>USBHS FIFO for Endpoint 10 Register</td>
</tr>
<tr>
<td>[0x0216]</td>
<td><code>USBHS_FIFO11</code></td>
<td>R/W</td>
<td>USBHS FIFO for Endpoint 11 Register</td>
</tr>
<tr>
<td>[0x0218]</td>
<td><code>USBHS_EPINFO</code></td>
<td>RO</td>
<td>USBHS Endpoint Count Info Register</td>
</tr>
<tr>
<td>[0x021A]</td>
<td><code>USBHS_RAMINFO</code></td>
<td>RO</td>
<td>USBHS RAM and MAInfo Register</td>
</tr>
<tr>
<td>[0x007A]</td>
<td><code>USBHS_SOFTRESET</code></td>
<td>R/W1C</td>
<td>USBHS Soft Reset Control Register</td>
</tr>
</tbody>
</table>
### 21.12 USBHS Device Register Details

#### Table 21-6: USBHS Device Address Register

<table>
<thead>
<tr>
<th>USBHS Device Address Register</th>
<th>USBHS_FADDR</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>--------</td>
</tr>
</tbody>
</table>
| 7    | update | RO     | 0     | Read USBHS Device Update Status  
  0: The Device address in the bit field addr is presently used.  
  1: New address written to the bit field addr is pending. New address takes  
  effect at the end of the current transfer. |
| 6:0  | faddr | R/W    | 0     | USBHS Device Address  
  This is the USB Device address specified by the external USB Host during the  
  enumeration process. It must be written with the address value contained in the  
  SET_ADDRESS Device request when received during a Control Transaction. |

#### Table 21-7: USBHS Power Management Register

<table>
<thead>
<tr>
<th>USBHS Power Management</th>
<th>USBHS_POWER</th>
<th>[0x0001]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>--------</td>
</tr>
</tbody>
</table>
| 7    | iso_update | R/W     | 0     | Isochronous Update  
  1: If an SOF token is received from the Host and a packet is in the IN FIFO  
  (USBHS_INCSRL.inpktrdy= 1), then send the packet. However, if an IN token is  
  received from the Host before an SOF token, then send a zero-length data  
  packet.  
  Note: This register is only applicable in Isochronous Mode and ignored in all other  
  modes. |
| 6    | softconn | R/W     | 0     | Soft Connect/Disconnect PHY  
  0: The USB D+/D- lines of the PHY are tri-stated, and this USB is electrically  
  disconnected from the USB bus.  
  1: The USB D+/D- lines of the PHY are enabled. |
| 5    | hs_enable | R/W     | 1     | Enable Hi-Speed (HS) Mode  
  0: USB remains in Full Speed Mode even if connected to a USB HS port.  
  1: USB always negotiates for HS mode on the bus. |
| 4    | hs_mode | RO      | 0     | Read Hi-Speed Mode Status Flag  
  0 = USB in Full Speed Mode.  
  1 = USB in Hi-Speed Mode. |
| 3    | power_reset | RO      | 0     | Read RESET Mode Status Flag  
  0 = Normal operation.  
  1 = RESET state is on the bus. |
| 2    | resume | R/W     | 0     | Generate RESUME State  
  Set to generate a RESUME state on the bus. Once set, it should be left set for at  
  least 10ms and no more than 15ms, then cleared. |
| 1    | suspend | RO      | 0     | Read SUSPEND Mode Status  
  0 = Normal operation.  
  1 = USBHS is in SUSPEND Mode.  
  Note: Automatically cleared when a SUSPEND Mode interrupt occurs, or if the  
  resume bit (above) is set to 1. |
### USBHS Power Management

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>suspendm</td>
<td>R/W</td>
<td>0</td>
<td>SUSPEND Mode Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: SUSPEND Mode disabled. USB will not enter SUSPEND Mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: SUSPEND Mode allowed. If no activity is detected on the bus for more than</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.0ms, this USB enters low-power SUSPEND Mode.</td>
</tr>
</tbody>
</table>

Table 21-8: USBHS IN Endpoint Interrupt Flags Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16:12</td>
<td>-</td>
<td>ROC</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>11</td>
<td>ep11_in</td>
<td>ROC</td>
<td>0</td>
<td>IN EP11 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: IN Endpoint 11 not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: IN Endpoint 11 occurred.</td>
</tr>
<tr>
<td>10</td>
<td>ep10_in</td>
<td>ROC</td>
<td>0</td>
<td>IN EP10 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: IN Endpoint 10 not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: IN Endpoint 10 occurred.</td>
</tr>
<tr>
<td>9</td>
<td>ep9_in</td>
<td>ROC</td>
<td>0</td>
<td>IN EP9 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: IN Endpoint 9 not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: IN Endpoint 9 occurred.</td>
</tr>
<tr>
<td>8</td>
<td>ep8_in</td>
<td>ROC</td>
<td>0</td>
<td>IN EP8 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: IN Endpoint 8 not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: IN Endpoint 8 occurred.</td>
</tr>
<tr>
<td>7</td>
<td>ep7_in</td>
<td>ROC</td>
<td>0</td>
<td>IN EP7 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: IN Endpoint 7 not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: IN Endpoint 7 occurred.</td>
</tr>
<tr>
<td>6</td>
<td>ep6_in</td>
<td>ROC</td>
<td>0</td>
<td>IN EP6 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: IN Endpoint 6 not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: IN Endpoint 6 occurred.</td>
</tr>
<tr>
<td>5</td>
<td>ep5_in</td>
<td>ROC</td>
<td>0</td>
<td>IN EP5 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: IN Endpoint 5 not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: IN Endpoint 5 occurred.</td>
</tr>
<tr>
<td>4</td>
<td>ep4_in</td>
<td>ROC</td>
<td>0</td>
<td>IN EP4 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: IN Endpoint 4 not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: IN Endpoint 4 occurred.</td>
</tr>
<tr>
<td>3</td>
<td>ep3_in</td>
<td>ROC</td>
<td>0</td>
<td>IN EP3 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: IN Endpoint 3 not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: IN Endpoint 3 occurred.</td>
</tr>
<tr>
<td>2</td>
<td>ep2_in</td>
<td>ROC</td>
<td>0</td>
<td>IN EP2 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: IN Endpoint 2 not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: IN Endpoint 2 occurred.</td>
</tr>
<tr>
<td>1</td>
<td>ep1_in</td>
<td>ROC</td>
<td>0</td>
<td>IN EP1 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: IN Endpoint 1 not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: IN Endpoint 1 occurred.</td>
</tr>
<tr>
<td>0</td>
<td>ep0</td>
<td>ROC</td>
<td>0</td>
<td>EP0 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: In Endpoint 0 not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: In Endpoint 0 occurred.</td>
</tr>
</tbody>
</table>
### Table 21-9: USBHS OUT Endpoint Interrupt Flags Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16:12</td>
<td>-</td>
<td>ROC</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>11</td>
<td>ep11_out</td>
<td>ROC</td>
<td>0</td>
<td>OUT EP11 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: OUT Endpoint 11 interrupt not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: OUT Endpoint 11 interrupt active.</td>
</tr>
<tr>
<td>10</td>
<td>ep10_out</td>
<td>ROC</td>
<td>0</td>
<td>OUT EP10 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: OUT Endpoint 10 interrupt not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: OUT Endpoint 10 interrupt active.</td>
</tr>
<tr>
<td>9</td>
<td>ep9_out</td>
<td>ROC</td>
<td>0</td>
<td>OUT EP9 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: OUT Endpoint 9 interrupt not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: OUT Endpoint 9 interrupt active.</td>
</tr>
<tr>
<td>8</td>
<td>ep8_out</td>
<td>ROC</td>
<td>0</td>
<td>OUT EP8 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: OUT Endpoint 8 interrupt not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: OUT Endpoint 8 interrupt active.</td>
</tr>
<tr>
<td>7</td>
<td>ep7_out</td>
<td>ROC</td>
<td>0</td>
<td>OUT EP7 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: OUT Endpoint 7 interrupt not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: OUT Endpoint 7 interrupt active.</td>
</tr>
<tr>
<td>6</td>
<td>ep6_out</td>
<td>ROC</td>
<td>0</td>
<td>OUT EP6 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: OUT Endpoint 6 interrupt not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: OUT Endpoint 6 interrupt active.</td>
</tr>
<tr>
<td>5</td>
<td>ep5_out</td>
<td>ROC</td>
<td>0</td>
<td>OUT EP5 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: OUT Endpoint 5 interrupt not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: OUT Endpoint 5 interrupt active.</td>
</tr>
<tr>
<td>4</td>
<td>ep4_out</td>
<td>ROC</td>
<td>0</td>
<td>OUT EP4 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: OUT Endpoint 4 interrupt not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: OUT Endpoint 4 interrupt active.</td>
</tr>
<tr>
<td>3</td>
<td>ep3_out</td>
<td>ROC</td>
<td>0</td>
<td>OUT EP3 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: OUT Endpoint 3 interrupt not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: OUT Endpoint 3 interrupt active.</td>
</tr>
<tr>
<td>2</td>
<td>ep2_out</td>
<td>ROC</td>
<td>0</td>
<td>OUT EP2 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: OUT Endpoint 2 interrupt not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: OUT Endpoint 2 interrupt active.</td>
</tr>
<tr>
<td>1</td>
<td>ep1_out</td>
<td>ROC</td>
<td>0</td>
<td>OUT EP1 Interrupt Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: OUT Endpoint 1 interrupt not active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt occurred.</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
<td>ROC</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
</tbody>
</table>

### Table 21-10: USBHS IN Endpoint Interrupt Enable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16:12</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>11</td>
<td>ep11_in</td>
<td>R/W</td>
<td>0</td>
<td>IN EP11 Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set to 1 to enable the interrupt for IN Endpoint 11.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled.</td>
</tr>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
<td>Reset</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
<td>--------</td>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| 10   | ep10_in| R/W    | 0     | **IN EP10 Interrupt Enable**  
0: Interrupt disabled.  
1: Interrupt enabled.      |
| 9    | ep9_in | R/W    | 0     | **IN EP9 Interrupt Enable**  
0: Interrupt disabled.  
1: Interrupt enabled.      |
| 8    | ep8_in | R/W    | 0     | **IN EP8 Interrupt Enable**  
0: Interrupt disabled.  
1: Interrupt enabled.      |
| 7    | ep7_in | R/W    | 0     | **IN EP7 Interrupt Enable**  
0: Interrupt disabled.  
1: Interrupt enabled.      |
| 6    | ep6_in | R/W    | 0     | **IN EP6 Interrupt Enable**  
0: Interrupt disabled.  
1: Interrupt enabled.      |
| 5    | ep5_in | R/W    | 0     | **IN EP5 Interrupt Enable**  
0: Interrupt disabled.  
1: Interrupt enabled.      |
| 4    | ep4_in | R/W    | 0     | **IN EP4 Interrupt Enable**  
0: Interrupt disabled.  
1: Interrupt enabled.      |
| 3    | ep3_in | R/W    | 0     | **IN EP3 Interrupt Enable**  
0: Interrupt disabled.  
1: Interrupt enabled.      |
| 2    | ep2_in | R/W    | 0     | **IN EP2 Interrupt Enable**  
0: Interrupt disabled.  
1: Interrupt enabled.      |
| 1    | ep1_in | R/W    | 0     | **IN EP1 Interrupt Enable**  
0: Interrupt disabled.  
1: Interrupt enabled.      |
| 0    | ep0    | R/W    | 0     | **EP0 Interrupt Enable**  
0: Interrupt disabled.  
1: Interrupt enabled.      |
### Table 21-11: USBHS OUT Endpoint Interrupt Enable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>16:12</td>
<td>-</td>
<td>R/W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>ep11_out</td>
<td>R/W</td>
<td>1</td>
<td>OUT EP11 Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set to 1 to enable the interrupt for OUT Endpoint 11.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled.</td>
</tr>
<tr>
<td>10</td>
<td>ep10_out</td>
<td>R/W</td>
<td>1</td>
<td>OUT EP10 Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set to 1 to enable the interrupt for OUT Endpoint 10.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled.</td>
</tr>
<tr>
<td>9</td>
<td>ep9_out</td>
<td>R/W</td>
<td>1</td>
<td>OUT EP9 Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set to 1 to enable the interrupt for OUT Endpoint 9.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled.</td>
</tr>
<tr>
<td>8</td>
<td>ep8_out</td>
<td>R/W</td>
<td>1</td>
<td>OUT EP8 Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set to 1 to enable the interrupt for OUT Endpoint 8.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled.</td>
</tr>
<tr>
<td>7</td>
<td>ep7_out</td>
<td>R/W</td>
<td>1</td>
<td>OUT EP7 Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set to 1 to enable the interrupt for OUT Endpoint 7.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled.</td>
</tr>
<tr>
<td>6</td>
<td>ep6_out</td>
<td>R/W</td>
<td>1</td>
<td>OUT EP6 Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set to 1 to enable the interrupt for OUT Endpoint 6.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled.</td>
</tr>
<tr>
<td>5</td>
<td>ep5_out</td>
<td>R/W</td>
<td>1</td>
<td>OUT EP5 Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set to 1 to enable the interrupt for OUT Endpoint 5.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled.</td>
</tr>
<tr>
<td>4</td>
<td>ep4_out</td>
<td>R/W</td>
<td>1</td>
<td>OUT EP4 Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set to 1 to enable the interrupt for OUT Endpoint 4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled.</td>
</tr>
<tr>
<td>3</td>
<td>ep3_out</td>
<td>R/W</td>
<td>1</td>
<td>OUT EP3 Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set to 1 to enable the interrupt for OUT Endpoint 3.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled.</td>
</tr>
<tr>
<td>2</td>
<td>ep2_out</td>
<td>R/W</td>
<td>1</td>
<td>OUT EP2 Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set to 1 to enable the interrupt for OUT Endpoint 2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled.</td>
</tr>
<tr>
<td>1</td>
<td>ep1_out</td>
<td>R/W</td>
<td>1</td>
<td>OUT EP1 Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set to 1 to enable the interrupt for OUT Endpoint 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt enabled.</td>
</tr>
</tbody>
</table>
### Table 21-12: USBHS Signaling Interrupt Status Flag Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:4</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>3</td>
<td>sof</td>
<td>R</td>
<td>0</td>
<td>Start Of Frame Detected Status Flag</td>
</tr>
<tr>
<td>2</td>
<td>reset</td>
<td>R</td>
<td>0</td>
<td>RESET State Detected Status Flag</td>
</tr>
<tr>
<td>1</td>
<td>resume</td>
<td>R</td>
<td>0</td>
<td>RESUME State Detected Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set when a RESUME state is detected while in SUSPEND Mode.</td>
</tr>
<tr>
<td>0</td>
<td>suspend</td>
<td>R</td>
<td>0</td>
<td>SUSPEND Mode Status Flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reads 1 when SUSPEND mode is active.</td>
</tr>
</tbody>
</table>

### Table 21-13: USBHS Signaling Interrupt Enable Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:4</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>3</td>
<td>sof</td>
<td>R/W</td>
<td>0</td>
<td>Start Of Frame (SOF) Detected Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt event disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt event enabled.</td>
</tr>
<tr>
<td>2</td>
<td>reset</td>
<td>R/W</td>
<td>1</td>
<td>RESET State Detected Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt event disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt event enabled.</td>
</tr>
<tr>
<td>1</td>
<td>resume</td>
<td>R/W</td>
<td>1</td>
<td>RESUME State Detected Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt event disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt event enabled.</td>
</tr>
<tr>
<td>0</td>
<td>suspend</td>
<td>R/W</td>
<td>0</td>
<td>SUSPEND Mode Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt event disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt event enabled.</td>
</tr>
</tbody>
</table>

### Table 21-14: USBHS Frame Number Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:11</td>
<td>framenum</td>
<td>R</td>
<td>0</td>
<td>Frame Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Always reads 0.</td>
</tr>
<tr>
<td>10:0</td>
<td>framenum</td>
<td>R</td>
<td>0</td>
<td>Read Last Received Frame Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This is the 11-bit frame number received in the SOF packet.</td>
</tr>
</tbody>
</table>
Table 21-15: USBHS Register Index Select Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:5</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td>Reserved for Future Use</td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>4:0</td>
<td>index</td>
<td>R/W</td>
<td>0x0</td>
<td>Index Register Access Selector</td>
</tr>
<tr>
<td></td>
<td>Index Register Access Selector</td>
<td></td>
<td></td>
<td>Each IN and OUT endpoint has memory-mapped control and status registers in addresses from 0x400B 1010 to 0x400B 1018. Only one endpoint’s registers are addressable in the memory map at time. This bit field selects which endpoint’s registers are present in the memory map where: 0x0: Endpoint 0 IN/OUT status registers addressable. 0x1: Endpoint 1 IN/OUT status registers addressable. 0x2: Endpoint 2 IN/OUT status registers addressable. … 0xB: Endpoint 11 IN/OUT status registers addressable.</td>
</tr>
</tbody>
</table>

Table 21-16: USBHS Test Mode Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:6</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td>Reserved for Future Use</td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>5</td>
<td>force_fs</td>
<td>R/W</td>
<td>0</td>
<td>Force FS Mode</td>
</tr>
<tr>
<td></td>
<td>Force FS Mode</td>
<td></td>
<td></td>
<td>When the USBHS receives a RESET from the Host, the USBHS is forced into Full-Speed mode.</td>
</tr>
<tr>
<td>4</td>
<td>force_hs</td>
<td>R/W</td>
<td>0</td>
<td>Force HS Mode</td>
</tr>
<tr>
<td></td>
<td>Force HS Mode</td>
<td></td>
<td></td>
<td>When the USBHS receives a RESET from the Host, the USBHS is forced into Hi-Speed mode.</td>
</tr>
<tr>
<td>3</td>
<td>test_packet</td>
<td>R/W</td>
<td>0</td>
<td>Test Packet Mode</td>
</tr>
<tr>
<td></td>
<td>Test Packet Mode</td>
<td></td>
<td></td>
<td>To enter this test mode, firmware must write the standard 53-byte test packet to the Endpoint 0 FIFO, then set USBHS_InCSRL.inpktrdy = 1, then set this bit. The DATA0 PID is automatically added to the head of the packet and the CRC to the end of the packet. The USBHS will continue to send the test packet until this bit is cleared.</td>
</tr>
<tr>
<td>2</td>
<td>test_k</td>
<td>R/W</td>
<td>0</td>
<td>HS Data K State Test Mode</td>
</tr>
<tr>
<td></td>
<td>HS Data K State Test Mode</td>
<td></td>
<td></td>
<td>The USBHS transmits a continuous Data K State</td>
</tr>
<tr>
<td>1</td>
<td>test_j</td>
<td>R/W</td>
<td>0</td>
<td>HS Data J State Test Mode</td>
</tr>
<tr>
<td></td>
<td>HS Data J State Test Mode</td>
<td></td>
<td></td>
<td>The USBHS transmits a continuous Data J State</td>
</tr>
<tr>
<td>0</td>
<td>test_se0_nak</td>
<td>R/W</td>
<td>0</td>
<td>SE0 NAK Test Mode</td>
</tr>
<tr>
<td></td>
<td>SE0 NAK Test Mode</td>
<td></td>
<td></td>
<td>The USBHS responds to any valid IN token with a NAK</td>
</tr>
</tbody>
</table>

21.12.2 Endpoint Register Access Control

Each IN and OUT endpoint from Endpoint 0x1 to 0xB uses memory mapped access to the registers in Table 21-17. Selecting a specific endpoint, using the USBHS_INDEX register, maps each of the registers in Table 21-17 to the selected endpoint.

Table 21-17: USB Memory Mapped Register Access for Endpoints 1 to 11

| USBHS_INMAXP | USBHS_INCSRL | USBHS_INCSRU |
21.12.3 USBHS IN Endpoint Maximum Packet Size Registers

Endpoints 1 to 11 have a memory mapped version of this register selected using the `USBHS_INDEX` register.

Table 21-18: USBHS IN Endpoint Maximum Packet Size Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:11</td>
<td>numpackminus1</td>
<td>R/W</td>
<td>0</td>
<td><strong>Number of Split Packets - 1</strong>&lt;br&gt;Defines the maximum number of packets minus 1 that a USB payload can be split into. This must be an exact multiple of <code>maxpacketsize</code>.&lt;br&gt;The total number of bytes transferred in the payload is:&lt;br&gt;<code>N_{BYTES,TRANSFERED} = maxpacketsize \times (numpackminus1 + 1)</code>&lt;br&gt;This must match the <code>maxpacketsize</code> field of the Standard Endpoint Descriptor for the associated endpoint.&lt;br&gt;For HS High Bandwidth Isochronous endpoints, the multiplier can only be 2 or 3, so this field can only be 0x01 or 0x02.&lt;br&gt;For Bulk endpoints, the max multiplier is 32, so the maximum value for this register is 31 (0x1F).&lt;br&gt;Note: Only applicable for High-Speed (HS), High-Bandwidth Isochronous endpoints, and Bulk endpoints. Ignored in all other cases.</td>
</tr>
<tr>
<td>10:0</td>
<td>maxpacketsize</td>
<td>R/W</td>
<td>0</td>
<td><strong>Maximum Packet Size in a Single Transaction</strong>&lt;br&gt;This is the maximum packet size, in bytes, that is transmitted for each microframe. The maximum value is 1024, subject to the limitations for the endpoint type set in the USB 2.0 Specification, Chapter 9.&lt;br&gt;For Bulk Transfers: The USB 2.0 Specification requires this to be 8, 16, 32, or 64. HS Bulk Transfer also supports 512.</td>
</tr>
</tbody>
</table>

21.12.4 USBHS IN Endpoint Lower Control and Status Registers

Endpoints 1 to 11 have a memory mapped version of this register selected using the `USBHS_INDEX` register.

Table 21-19: USBHS IN Endpoint Lower Control and Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>incompt</td>
<td>R/WDC</td>
<td>0</td>
<td><strong>Read Incomplete Split Transfer Error Status</strong>&lt;br&gt;High-Bandwidth Isochronous transfers:&lt;br&gt;Automatically set when a payload is split into multiple packets but insufficient IN tokens were received to send all packets. The current packet is flushed from the IN FIFO. Write a 0 to clear.&lt;br&gt;Bulk and Interrupt Transfers:&lt;br&gt;Ignored.</td>
</tr>
</tbody>
</table>
### USBHS IN Endpoint Lower Control and Status

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 6    | clrdatatog | R/W1O  | 0     | Clear IN Endpoint Data Toggle
1: Clear the IN Endpoint data toggle to 0.
*Note: Automatically cleared after set.* |
| 5    | sentstall  | R/W0C  | 0     | Read STALL Handshake Sent Status
Automatically set when a STALL handshake is transmitted, at which time the IN FIFO is flushed, and USBHS_INCSRL.inpktrdy is cleared.
*Note: Write a 0 to clear.* |
| 4    | sendstall  | R/W    | 0     | Send STALL Handshake
1: Respond to an IN token with a STALL handshake.
0: Terminate STALL handshake
*Note: Ignored for Isochronous transfers.* |
| 3    | flushfifo  | R/W1O  | 0     | Flush Next Packet from IN FIFO
1: Flush the next packet to be transmitted from the IN FIFO. This also clears the bit field USBHS_INCSRL.inpktrdy. This must only be set when USBHS_INCSRL.inpktrdy = 1, or FIFO data corruption might occur.
*Note: If the IN FIFO contains two packets, set the flushfifo field twice to clear both packets.*
*Note: Automatically cleared when the packet is flushed.* |
| 2    | underrun   | R/W0C  | 0     | Read IN FIFO Underrun Error Status
Isochronous Mode: Automatically set if the IN FIFO is empty (inpktrdy = 0), an IN token has been received, and a zero-length data packet has been sent.
Bulk or Interrupt Modes: Automatically set when an IN token is received, and a NAK is sent.
*Note: Write a 0 to clear.* |
| 1    | fifonotempty | R/W0C | 0     | Read FIFO Not Empty Status
Automatically set when there is at least one packet in the IN FIFO.
*Note: Write a 0 to clear.* |
| 0    | inpktrdy   | R/W1O  | 0     | IN Packet Ready
1: Write a 1 after writing a data packet to the IN FIFO. Automatically cleared when the data packet is transmitted. If double-buffering is enabled, this bit is automatically cleared when there is space for a second packet in the FIFO.
*Note: This bit field is also controlled by USBHS_INCSR0.autoset.* |

### USBHS Endpoint 0 Control Status Register

#### Table 21-20: USBHS Endpoint 0 Control Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
</table>
| 7    | servicedsetupend | R/W1C  | 0     | Clear EP0 Setup End Bit
Write a 1 to clear the setupend bit.
*Note: Automatically cleared after being set.* |
| 6    | servicedoutpktrdy | R/W1C | 0     | Clear EP0 Out Packet Ready Bit
Write a 1 to clear the outpktrdy bit (below).
*Note: Automatically cleared after being set.* |
### USBHS Endpoint 0 Control Status

#### Table 21-1: USBHS_ENDPOINT Upper Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>sendstall</td>
<td>R/W1O</td>
<td>0</td>
<td><strong>Send EP0 STALL Handshake</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write a 1 to this bit to terminate the current Control Transaction by sending a STALL handshake.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Automatically cleared after being set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note: This behavior is different from the sendstall bits associated with IN/OUT endpoints.</strong></td>
</tr>
<tr>
<td>4</td>
<td>setupend</td>
<td>RO</td>
<td>0</td>
<td><strong>Read Setup End Status</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Automatically set when a Control Transaction ends before the dataend bit has been set by firmware.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>An interrupt is generated when this bit is set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write a 1 to servicedsetupend (above) to clear.</td>
</tr>
<tr>
<td>3</td>
<td>dataend</td>
<td>R/W1O</td>
<td>0</td>
<td><strong>Control Transaction Data End</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write a 1 to this bit after firmware completes any of the following three transactions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1) Set inpktrdy = 1 for the last data packet.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2) Set inpktrdy = 1 for a zero-length data packet.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3) Clear outpktrdy = 0 after unloading the last data packet.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note: Automatically cleared after being set.</strong></td>
</tr>
<tr>
<td>2</td>
<td>sentstall</td>
<td>R/W0C</td>
<td>0</td>
<td><strong>Read EP0 STALL Handshake Sent Status</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Automatically set when a STALL handshake is transmitted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write a 0 to clear.</td>
</tr>
<tr>
<td>1</td>
<td>inpktrdy</td>
<td>R/W1O</td>
<td>0</td>
<td><strong>EP0 IN Packet Ready</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Set this bit to indicate a packet is ready to transmit from the IN FIFO. Hardware automatically clears this bit when the packet transmit is complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Packet was transmitted or no packet transmit pending. Read only.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Write a 1 after writing a data packet to the IN FIFO to indicate the EP0 IN packet is ready.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Note: An interrupt is generated when this bit is cleared.</strong></td>
</tr>
<tr>
<td>0</td>
<td>outpktrdy</td>
<td>RO</td>
<td>0</td>
<td><strong>EP0 OUT Packet Ready Status</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Automatically set when a data packet is received in the OUT FIFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>An interrupt is generated when this bit is set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Write a 1 to the servicedoutpktrdy bit (above) to clear after the packet is unloaded from the OUT FIFO.</td>
</tr>
</tbody>
</table>

### 21.12.6 USBHS IN Endpoint Upper Control Registers

Endpoint 1 to 11 have a memory mapped version of this register selected using the **USBHS_INDEX** register.

#### Table 21-2: USBHS IN Endpoint Upper Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>autoset</td>
<td>R/W</td>
<td>0</td>
<td><strong>Auto Set inpktrdy</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: <strong>USBHS_INCSRU.inpktrdy</strong> must be set by firmware</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: <strong>USBHS_INCSRU.inpktrdy</strong> is automatically set when data that is of the maximum packet size specified in the <strong>USBHS_INMAXP</strong> register is loaded into the IN FIFO.</td>
</tr>
</tbody>
</table>
### USBHS IN Endpoint Upper Control

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>iso</td>
<td>R/W</td>
<td>0</td>
<td><strong>Isochronous Transfer Enable</strong>&lt;br&gt;0: Enable IN Bulk and IN Interrupt transfers&lt;br&gt;1: Enable IN Isochronous transfers</td>
</tr>
<tr>
<td>5</td>
<td>mode</td>
<td>R/W</td>
<td>0</td>
<td><strong>Endpoint Direction Mode</strong>&lt;br&gt;0: Endpoint direction is OUT&lt;br&gt;1: Endpoint direction is IN&lt;br&gt;Note: Ignored if endpoint is not used for both IN and OUT transactions.</td>
</tr>
<tr>
<td>4</td>
<td>dmareqenab</td>
<td>R/W</td>
<td>0</td>
<td><strong>DMA Request Enable</strong>&lt;br&gt;0: Disable DMA for this IN endpoint&lt;br&gt;1: Enable DMA for this IN endpoint</td>
</tr>
<tr>
<td>3</td>
<td>frcdatatog</td>
<td>R/W</td>
<td>0</td>
<td><strong>Force IN Data-Toggle</strong>&lt;br&gt;0: Toggle data-toggle only when an ACK is received&lt;br&gt;1: Toggle data-toggle regardless of whether an ACK is received&lt;br&gt;Note: Useful for Interrupt IN endpoints that are communicating rate feedback to Isochronous endpoints.</td>
</tr>
<tr>
<td>2</td>
<td>dmareqmode</td>
<td>R/W</td>
<td>0</td>
<td><strong>DMA Request Mode Enable</strong>&lt;br&gt;0: Enable DMA Request Mode 0. A DMA request is generated for each packet transmission. This mode can only be selected after the dmareqenab bit is cleared first.&lt;br&gt;1: Enable DMA Request Mode 1. A DMA request is generated only when a packet of size USBHS_INMAXP_maxpacketsize is received.</td>
</tr>
<tr>
<td>1</td>
<td>dpktbufdis</td>
<td>R/W</td>
<td>0</td>
<td><strong>Double Packet Buffering Disable</strong>&lt;br&gt;0: Enable double packet buffering. Firmware must also configure the FIFO and packet size.&lt;br&gt;1: Disable double packet buffering</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
</tbody>
</table>

Table 21-22: **USBHS OUT Endpoint Maximum Packet Size Register**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:11</td>
<td>numpackminus1</td>
<td>R/W</td>
<td>0x00</td>
<td><strong>Number of Split Packets</strong>&lt;br&gt;Defines the maximum number of packets minus 1 that a USB payload is combined into. The value must be an exact multiple of maxpacketsize. The total number of bytes transferred in the payload is maxpacketsize x (numpackminus1+1). This must match the maxpacketsize field of the Standard Endpoint Descriptor for the associated endpoint. Only applicable for Hi-Speed (HS), High Bandwidth Isochronous endpoints, and Bulk endpoints. Ignored in all other cases. <strong>HS High Bandwidth Isochronous Endpoints</strong>&lt;br&gt;The multiplier can only be 2 or 3, so this bit field value can only be 0x01 or 0x02. <strong>Bulk Endpoints</strong>&lt;br&gt;The max multiplier is 32, so the maximum value for this register is 31 (0x1F).</td>
</tr>
</tbody>
</table>
### USBHS OUT Endpoint Maximum Packet Size

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:0</td>
<td>maxpacketsize</td>
<td>R/W</td>
<td>0x000</td>
<td><strong>Maximum Packet Size in a Single Transaction</strong>&lt;br&gt;This is the maximum packet size, in bytes, that is transmitted for each microframe. The maximum value is 1024, subject to the limitations for the endpoint type set in the USB 2.0 Specification, Chapter 9. For all Bulk Transfers, the USB 2.0 Specification requires this to be 8, 16, 32, or 64. HS Bulk Transfer also supports 512.</td>
</tr>
</tbody>
</table>

### Table 21-23: USBHS OUT Endpoint Lower Control Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>cirdatatog</td>
<td>R/W1O</td>
<td>0</td>
<td><strong>Clear OUT Endpoint Data Toggle</strong>&lt;br&gt;1: Clear the OUT Endpoint data toggle to 0.&lt;br&gt;Note: Automatically cleared.</td>
</tr>
<tr>
<td>6</td>
<td>sentstall</td>
<td>R/WOC</td>
<td>0</td>
<td><strong>STALL Handshake Sent Status</strong>&lt;br&gt;Automatically set when a STALL handshake is transmitted.&lt;br&gt;Write a 0 to clear.</td>
</tr>
<tr>
<td>5</td>
<td>sendstall</td>
<td>R/W</td>
<td>0</td>
<td><strong>Send STALL Handshake</strong>&lt;br&gt;1: Send a STALL handshake to a data packet&lt;br&gt;0: Terminate STALL handshake&lt;br&gt;Ignored for Isochronous transfers. Write a 0 to clear.</td>
</tr>
<tr>
<td>4</td>
<td>flushfifo</td>
<td>R/W1O</td>
<td>0</td>
<td><strong>Flush OUT FIFO Packet</strong>&lt;br&gt;1: Flush the next packet to be read from the OUT FIFO. This also clears the outpktrdy bit. This must only be set when outpktrdy = 1, or data corruption in the FIFO might occur.&lt;br&gt;If the out FIFO contains two packets, flushfifo might need to be set twice to completely clear the FIFO.&lt;br&gt;Note: Automatically cleared when the packet is flushed.</td>
</tr>
<tr>
<td>3</td>
<td>dataerror</td>
<td>RO</td>
<td>0</td>
<td><strong>OUT Packet CRC Error Status</strong>&lt;br&gt;Isochronous Mode: Automatically set if a data packet is received (outpktrdy = 1), and the data packet has a CRC error. Automatically cleared when outpktrdy = 0.&lt;br&gt;Bulk or Interrupt Modes: Always returns 0.</td>
</tr>
<tr>
<td>2</td>
<td>overrun</td>
<td>R/WOC</td>
<td>0</td>
<td><strong>OUT FIFO Overrun Error Status</strong>&lt;br&gt;Isochronous Mode:&lt;br&gt;Automatically set if the OUT FIFO is full (fifofull = 1), and an OUT packet arrives. In this case, the OUT packet is lost.&lt;br&gt;Bulk or Interrupt Modes:&lt;br&gt;Always reads 0.&lt;br&gt;Note: Write a 0 to clear.</td>
</tr>
<tr>
<td>1</td>
<td>fifofull</td>
<td>RO</td>
<td>0</td>
<td><strong>FIFO Full Status</strong>&lt;br&gt;Set when the OUT FIFO is full.&lt;br&gt;Note: Automatically cleared when the FIFO is no longer full.</td>
</tr>
<tr>
<td>0</td>
<td>outpktrdy</td>
<td>R/WOC</td>
<td>0</td>
<td><strong>OUT Packet Ready Status</strong>&lt;br&gt;Automatically set when a data packet is received in the OUT FIFO.&lt;br&gt;Write a 0 to clear after the packet is unloaded from the OUT FIFO.&lt;br&gt;Note: Write 0 to clear.</td>
</tr>
</tbody>
</table>
### Table 21-24: USBHS OUT Endpoint Upper Control Status Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>autoclear</td>
<td>R/W</td>
<td>0</td>
<td><strong>Auto Clear outpktrdy</strong>&lt;br&gt;0: USBHS_OUTCSR.outpktrdy must be cleared by firmware.&lt;br&gt;1: USBHS_OUTCSR.outpktrdy is automatically cleared when data that is of the maximum packet size specified in the USBHS_OUTMAXP register is unloaded from the OUT FIFO. If packets less than the maximum packet size are unloaded, outpktrdy must be cleared by firmware.&lt;br&gt;Note: Do not set for High Bandwidth Isochronous endpoints.</td>
</tr>
<tr>
<td>6</td>
<td>iso</td>
<td>R/W</td>
<td>0</td>
<td><strong>Isochronous Transfer Enable</strong>&lt;br&gt;0: Enable OUT Bulk and OUT Interrupt transfers.&lt;br&gt;1: Enable OUT Isochronous transfers.</td>
</tr>
<tr>
<td>5</td>
<td>dmareqen</td>
<td>R/W</td>
<td>0</td>
<td><strong>DMA Request Enable</strong>&lt;br&gt;0: Disable DMA for this OUT endpoint.&lt;br&gt;1: Enable DMA for this OUT endpoint.</td>
</tr>
<tr>
<td>4</td>
<td>disnyet/piderror</td>
<td>R/W</td>
<td>0</td>
<td><strong>Disable NYET Packets (HS Bulk and HS Interrupt Modes)</strong>&lt;br&gt;0: If the OUT FIFO is full, respond to newly received OUT packets with a NYET (Not Yet) packet to indicate the FIFO is full.&lt;br&gt;1: Disable NYET packets. Respond to all received OUT packets with an ACK even when the FIFO is full.&lt;br&gt;<strong>PID Error Status (Isochronous Mode only)</strong>&lt;br&gt;Automatically set if there is a PID (Packet ID) error in the received OUT packet.&lt;br&gt;Note: Write 0 to clear.&lt;br&gt;Note: Ignored in all other modes.&lt;br&gt;Note: Bit 4 is dual-use and can be addressed by two different names depending on the endpoint mode.</td>
</tr>
<tr>
<td>3</td>
<td>dmareqmode</td>
<td>R/W</td>
<td>0</td>
<td><strong>DMA Request Mode Enable</strong>&lt;br&gt;0: Enable DMA Request Mode 0. A DMA request is generated after each OUT packet is received.&lt;br&gt;1: Enable DMA Request Mode 1. A DMA request is generated only when a packet of USBHS_OUTMAXP.maxpacketsize is received.</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>R/W</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field.</td>
</tr>
<tr>
<td>1</td>
<td>dpktbufdis</td>
<td>R/W</td>
<td>0</td>
<td><strong>Double Packet Buffering Disable</strong>&lt;br&gt;0: Enable double packet buffering. Firmware must configure the FIFO and packet size.&lt;br&gt;1: Disable double packet buffering.</td>
</tr>
<tr>
<td>0</td>
<td>incomprx</td>
<td>R</td>
<td>0</td>
<td><strong>Incomplete Isochronous Packet Received Error Status</strong>&lt;br&gt;High Bandwidth Isochronous Mode:&lt;br&gt;Automatically set if an incomplete packet is received in the OUT FIFO. Automatically cleared when USBHS_OUTCSR.outpktrdy is cleared.&lt;br&gt;Bulk or Interrupt Modes:&lt;br&gt;Always reads 0.</td>
</tr>
</tbody>
</table>

*Note: Endpoint 1 to 11 have a memory mapped version of this register selected using the USBHS_INDEX register.*
### Table 21-25: USBHS Endpoint OUT FIFO Byte Count Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:13</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>12:0</td>
<td>outcount</td>
<td>RO</td>
<td>0</td>
<td>Read Number of Data Bytes in OUT FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Returns the number of data bytes in the packet that are read next in the OUT FIFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: This value changes as the contents of the FIFO change.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: This value is only valid when a packet is in the OUT FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(USBHS_OUTCSRL.outpktrdy = 1).</td>
</tr>
</tbody>
</table>

### Table 21-26: USBHS Endpoint 0 IN FIFO Byte Count Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:7</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field.</td>
</tr>
<tr>
<td>6:0</td>
<td>count0</td>
<td>RO</td>
<td>0</td>
<td>Read Number of Data Bytes in the Endpoint 0 FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Returns the number of data bytes in the Endpoint 0 FIFO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: This field changes as the contents of the FIFO change.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: This field is only valid when USBHS_OUTCSRL.outpktrdy = 1.</td>
</tr>
</tbody>
</table>

### Table 21-27: USBHS FIFO for Endpoint n Register

<table>
<thead>
<tr>
<th>USBHS FIFO for Endpoint 0</th>
<th>USBHS_FIFO0</th>
<th>[0x0020]</th>
</tr>
</thead>
<tbody>
<tr>
<td>USBHS FIFO for Endpoint 1</td>
<td>USBHS_FIFO1</td>
<td>[0x0024]</td>
</tr>
<tr>
<td>USBHS FIFO for Endpoint 2</td>
<td>USBHS_FIFO2</td>
<td>[0x0028]</td>
</tr>
<tr>
<td>USBHS FIFO for Endpoint 3</td>
<td>USBHS_FIFO3</td>
<td>[0x002C]</td>
</tr>
<tr>
<td>USBHS FIFO for Endpoint 4</td>
<td>USBHS_FIFO4</td>
<td>[0x0030]</td>
</tr>
<tr>
<td>USBHS FIFO for Endpoint 5</td>
<td>USBHS_FIFO5</td>
<td>[0x0034]</td>
</tr>
<tr>
<td>USBHS FIFO for Endpoint 6</td>
<td>USBHS_FIFO6</td>
<td>[0x0038]</td>
</tr>
<tr>
<td>USBHS FIFO for Endpoint 7</td>
<td>USBHS_FIFO7</td>
<td>[0x003C]</td>
</tr>
<tr>
<td>USBHS FIFO for Endpoint 8</td>
<td>USBHS_FIFO8</td>
<td>[0x0040]</td>
</tr>
<tr>
<td>USBHS FIFO for Endpoint 9</td>
<td>USBHS_FIFO9</td>
<td>[0x0044]</td>
</tr>
<tr>
<td>USBHS FIFO for Endpoint 10</td>
<td>USBHS_FIFO10</td>
<td>[0x0048]</td>
</tr>
<tr>
<td>USBHS FIFO for Endpoint 11</td>
<td>USBHS_FIFO11</td>
<td>[0x004C]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>usbhs_fifo</td>
<td>R/W</td>
<td>-</td>
<td>USBHS Endpoint FIFO Read/Write Register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reads from this register unload data from the OUT FIFO for the corresponding endpoint.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Writes to this register load data into the IN FIFO for the corresponding endpoint.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FIFO reads and writes may be 8-bit, 16-bit, 24-bit or 32-bit. Any combination is allowed provided the data accessed is contiguous.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>However, all reads and writes for a packet must be of the same width so that the data is consistently byte-, word- or double-word-aligned. The last transfer can contain fewer bytes than the previous transfers when completing an odd-byte or odd-word transfer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: The value of these registers at reset is undetermined.</td>
</tr>
</tbody>
</table>
Table 21-28: USBHS Endpoint Count Info Register

<table>
<thead>
<tr>
<th>USBHS Endpoint Count Info</th>
<th>USBHS_EPINFO</th>
<th>[0x0078]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>7:4</td>
<td>outendpoints</td>
<td>RO</td>
</tr>
<tr>
<td>3:0</td>
<td>inendpoints</td>
<td>RO</td>
</tr>
</tbody>
</table>

Table 21-29: USBHS RAM Info Register

<table>
<thead>
<tr>
<th>USBHS RAM Info</th>
<th>USBHS_RAMINFO</th>
<th>[0x0079]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>7:4</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td>3:0</td>
<td>rambits</td>
<td>RO</td>
</tr>
</tbody>
</table>

Table 21-30: USBHS Soft Reset Control Register

<table>
<thead>
<tr>
<th>USBHS Soft Reset Control</th>
<th>USBHS_SOFTRESET</th>
<th>[0x007A]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>7:2</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td>1</td>
<td>rstxs</td>
<td>R/W1O</td>
</tr>
<tr>
<td>0</td>
<td>rsts</td>
<td>R/W1O</td>
</tr>
</tbody>
</table>

Table 21-31: USBHS Early DMA Register

<table>
<thead>
<tr>
<th>USBHS Early DMA</th>
<th>USBHS_EARLYDMA</th>
<th>[0x007B]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>Name</td>
<td>Access</td>
</tr>
<tr>
<td>7:2</td>
<td>-</td>
<td>R/W</td>
</tr>
<tr>
<td>1</td>
<td>edmain</td>
<td>R/W</td>
</tr>
</tbody>
</table>
### USBHS Early DMA

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>edmaout</td>
<td>R/W</td>
<td>0</td>
<td>Early DMA OUT Endpoints Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: DMA Request signal for all OUT endpoints is deasserted when</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>USBHS_INMAXP bytes have been read from an endpoint.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: DMA Request signal for all OUT endpoints is deasserted when (MAXP − 8)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>bytes have been read from an endpoint.</td>
</tr>
</tbody>
</table>

Table 21-32: USBHS Hi-Speed Chirp Timeout Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:0</td>
<td>c_tuch</td>
<td>R/W</td>
<td>0x203A</td>
<td>HS Chirp Timeout Clock Cycles</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$t_{CHIRP_TIMEOUT}(PHY_clock_cycles) = c_t_uch \times 4$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The timeout value represents the number of 30MHz PHY clock cycles (66.7ns) before the chirp timeout occurs.</td>
</tr>
</tbody>
</table>

Table 21-33: USBHS Hi-Speed RESUME Delay Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:0</td>
<td>c_t_hsrtn</td>
<td>R/W</td>
<td>0x0019</td>
<td>Hi-Speed RESUME Delay Clock Cycles</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$t_{HI_SPEED_DELAY}(PHY_clock_cycles) = c_t_hsrtn \times 4$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The delay value represents the number of 30MHz PHY clock cycles (66.7ns) from the end of the RESUME state to when normal USBHS operation begins.</td>
</tr>
</tbody>
</table>
22. **Bluetooth 5 Low Energy (LE) Radio**

Bluetooth 5 Low Energy (LE) radio is the latest version of the Bluetooth wireless communication standard. It is used for wireless headphones and other audio hardware, as well as for communication between various smart home and Internet of Things (IoT) devices.

Devices operate in the unlicensed 2.4GHz ISM (Industrial, Scientific, Medical) band. A frequency-hopping transceiver is used to combat interference and fading. The system operates in the 2.4GHz ISM band at 2400MHz–2483.5MHz. It uses 40 RF channels. These RF channels have center frequencies 2402 + k x 2MHz, where k = 0, ..., 39.

Bluetooth 5 technology provides:

- 1Mbps, 2Mbps, and Long Range coded (125kbps and 500kbps) data rates
- Increased broadcast capability
- Advertising packet up to 255 bytes
- On-chip matching network to the antenna
- Provides hardware on-the-fly encryption and decryption for lower power consumption
- Supports mesh networking
- Provides high-quality audio streaming (isochronous)
- Low-power proprietary mode that supports 20kbps, 40kbps, 500kbps-MSK/GFSK, 1Mpbs-GFSK

22.1 **Power-Efficient Design**

The provided Bluetooth Low Energy radio is optimized for low-power operation.

- Higher transmit power up to +9.5dbm
- Low transmit current of 2.5mA at 0dbm at 3.3V
- Low receive current of 1.5mA at 3.3V

22.2 **Bluetooth Hardware Accelerator**

The dedicated Bluetooth hardware accelerator eliminates the need for application software to accommodate the strict timing requirements and restrictions. It transparently increases system performance while reducing overall power consumption of the Bluetooth system.

22.3 **Arm Cordio®-B50 Software Stack**

Maxim provides the Arm Cordio®-B50 stack in library form. This provides application developers access to Bluetooth without validation and development of a software stack.

The Arm Cordio-B50 software stack interfaces to the Bluetooth link layer running on dedicated hardware. The dedicated hardware for the stack enables the ultimate in power management for IoT applications.

*Cordio is a registered trademark of Arm Limited.*

---

**MAX32665-MAX32668 User Guide**
Arm Cordio-B50 features the following:

- C library for linking directly into an application development tool
- Host selects the PHY it needs to use at any given time enabling long range or higher bandwidth only when required
- LE 1M
- LE Coded S = 2
- LE Coded S = 8
- LE 2M
- Bluetooth 5 advertising extension support for enabling next generation Bluetooth beacons
- Larger packets and advertising channel offloading
- Packets up to 255 octets long
- Advertising packet chaining
- Advertising sets
- Periodic advertising
- High-duty cycle non-connectable advertising
- Sample applications using standard profiles built on the Arm Cordio-B50 software framework

*Figure 22-1: MAX32665—MAX32668 Bluetooth Stack Overview*
22.4 Pins

The interface has two device pins to connect to the off-chip user provided antenna. The ANT device pin is the radio frequency signal pin and it should be routed through the ANT THRU device pin to the antenna. The ANT device pin is a 50Ω source impedance driver.

22.5 Configuration

The Radio and Hardware Accelerator requires a 32MHz external crystal specified in the datasheet. The CPU core hosting the Arm Cordio-B50 stack must run at a core speed greater than 32MHz.

Perform the following steps to enable the Bluetooth radio:

1. Set GCR_BTLE_LDOCR.ldorxen and set to GCR_BTLE_LDOCR.ldotxen 1 to enable the internal Bluetooth LDO.
2. Set GCR_CLK_CTRL.x32M_en to 1.

22.6 Documentation

The Arm Cordio-B50 Stack Product Sheet and Profiles can be found at:


The Maxim MAX32665–MAX32668 Low Power Arm Micro Toolchain for both iOS and Windows can be found at:


This Toolchain includes documentation of the Arm Cordio-B50 Stack and profile examples.
23. **Trust Protection Unit (TPU)**

The trust protection unit (TPU) is a collection of hardware and software mechanisms that provide advanced cryptographic security. Dedicated hardware engines greatly increase the speed of computationally intensive cryptographic algorithms.

The dedicated symmetric block cipher engine provides the following features:

- AES-128, 192, and 256 (FIPS 197).
- DES and 3DES/TDEA (NIST SP800-67).
- Support for NIST-approved block modes (SP800-38).
- Parallel calculation of block cipher and hash functions

The requirements for meeting security validations are often updated. Contact Maxim before starting any secure product design to ensure that the cryptographic features of this device are compatible with the most recent requirements.

The dedicated hash function accelerator computes SHA-1, 224, 256, 384, and 512 (FIPS 180-3) values used in CMAC and HMAC.

Hamming code generator provides the ability to calculate an error correction code (ECC) on a block of data that can detect single or two-bit errors.

The cryptographic accelerator also provides a dedicated modular arithmetic accelerator (MAA). It provides high speed calculations of asymmetrical keys used in DSA, RSA, ECDSA and other cryptographic algorithms with modulus and operands up to 2048 bits in length. The MAA has a dedicated memory space for the operands and operates independently of the CPU except when loading or unloading the operands.

Most functions are configurable for big- or little-endian operations.

The cryptographic accelerator interfaces with both the APB and AHB busses.

All cryptographic operations begin by resetting the cryptographic block. The cryptographic accelerators functions each have their own done bit, as well a global done bit for the cryptographic block. The cryptographic accelerators can generate an interrupt if enabled.
23.1 Dedicated Cryptographic DMA Engine (CDMA)

A dedicated DMA engine performs high-speed accesses between the TPU and memory on the AHB bus. This greatly improves performance during data-intensive operations such as encryption/decryption and hashing. The source, destination, and count registers are located in the cryptographic accelerator register space. The source and destination of the DMA engine can point to the same memory location to encrypt or decrypt the data in situ.

While the cryptographic accelerator is busy encrypting or hashing data, the DMA prefetches the data for the next operation and stores it in the read FIFO. Once the cipher or hash generator is done, the data for the next operation is immediately available. Data output is buffered in the write FIFO so the next cipher or hash operation can immediately begin calculating on the next block. This keeps the cipher and hash generator running continuously without having to wait for data to be written to or read from the bus.

The block cipher can operate in parallel with the hash accelerator as long as only one operation uses the DMA.
23.1.1 FIFOs

The read FIFO and write FIFO have programmable sources as shown in Table 23-1 to allow flexibility in their operation.

Table 23-1. Cryptographic Accelerator DMA Sources

<table>
<thead>
<tr>
<th>READ FIFO SOURCES</th>
<th>WRITE FIFO SOURCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read FIFO</td>
<td>Write FIFO</td>
</tr>
<tr>
<td>APB</td>
<td>None</td>
</tr>
<tr>
<td>AHB DMA</td>
<td>Cipher output</td>
</tr>
<tr>
<td>Random Number Generator</td>
<td></td>
</tr>
</tbody>
</table>

During cryptographic operations, a typical setup is to use the AHB DMA as the read FIFO source and the cipher output as the write FIFO source. Data written to the write FIFO is always written out to the AHB DMA. This setup reads data from memory and writes the encrypted or decrypted result back to memory.

A Cipher-based Message Authentication Code (CMAC) is similar to a digital signature or a Keyed-Hash Message Authentication Code (HMAC). CMACs use a cipher in a block-chaining mode to form a cryptographic checksum. In this mode, the AHB DMA is the read FIFO source, but the cipher output is not written back to memory. Only the final cipher block is of interest, so set the write FIFO source to none.

You can use DMA to copy memory, similar to the memcpy standard C function, by setting the write FIFO source to the read FIFO. If the Hamming ECC generator is enabled, you can copy flash memory pages to memory while simultaneously calculating the error correction code.

You can fill memory with a block of data similar to the memset() standard C function by pointing the write FIFO source to the read FIFO source to the random number generator and the write FIFO source to the read FIFO.

To decrypt or encrypt data, set the write FIFO source to the cipher output. To implement memcpy() or memset() functions, or to fill memory with random data, you should set the write FIFO source to the read FIFO. When calculating a hash or CMAC, disable the write FIFO.
The setting of the read and write FIFO sources are detailed in each section of specific operations.

For cipher, hash, or Galois operations most operations are finished when the DMA transfer is complete. If the CRYPTO_CTRL.rdsr is configured for DMA, both the CRYPTO_CTRL.dma_done and the associated .done flag are set after the entire DMA operation is complete. In most cases only the CRYPTO_CTRL.done flag is required. Setting the CRYPTO_CTRL.dmadnemsk field will prevent the CRYPTO_CTRL.dma_done field from setting the CRYPTO_CTRL.done bit. This reduces software overhead by only using the specific operation’s .done flag and masking the DMA from interrupting the CPU.

For cipher operations, when CRYPTO_CTRL.wrsrc is configured for cipher output, the CRYPTO_CTRL.cph_done is set only after the last cipher text has completed the DMA transfer out to memory.

After the cryptographic accelerator reset, the CRYPTO_CTRL.rdy must be polled in the software before any other actions can start.

### 23.1.2 Direct FIFO Access

The read and write FIFOS are directly accessible via the CRYPTO_DIN_[3:0] and CRYPTO_DOUT_[3:0] registers, respectively. In general, however, the CMDA is much more efficient and requires less interaction.

If direct access is required, only 32-bit accesses to the CRYPTO_DIN_0 and CRYPTO_DOUT_0 registers should be used. Do not use the registers [3:1] for direct access.

### 23.1.3 Cache Security

Cryptographic operands and results may be stored in cached memory as part of normal device operation. For increased security, invalidate the cache memory associated with memory used in cryptographic operations.

### 23.2 Block Cipher Accelerator

The block cipher accelerator is a dedicated hardware module that accelerates the computation of the following algorithms:

- AES-128
- AES-192
- AES-256
- Data Encryption Standard (DES)
- Triple Data Encryption Algorithm (TDEA/3DES)
- DES and TDEA have been deprecated by NIST are only provided for legacy support.
Figure 23-3 is a block diagram of the block cipher accelerator and its interface to the CMDA.

**Figure 23-3. Block Cipher Block Diagram**

The symmetric block ciphers encrypt or decrypt data in blocks. The block sizes for each cipher are shown in Table 23-2.

<table>
<thead>
<tr>
<th>CIPHER</th>
<th>KEY SIZE</th>
<th>USED KEY BITS</th>
<th>EFFECTIVE STRENGTH (NIST SP800-57)</th>
<th>BLOCK SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDEA</td>
<td>192-bits</td>
<td>CIPHER_KEY[167:0] (168-bits)</td>
<td>112-bits</td>
<td>64-bits</td>
</tr>
<tr>
<td>AES-128</td>
<td>128-bits</td>
<td>CIPHER_KEY[127:0] 128-bits</td>
<td>128-bits</td>
<td>128-bits</td>
</tr>
<tr>
<td>AES-192</td>
<td>192-bits</td>
<td>CIPHER_KEY[191:0] 192-bits</td>
<td>192-bits</td>
<td>128-bits</td>
</tr>
<tr>
<td>AES-256</td>
<td>256-bits</td>
<td>CIPHER_KEY[255:0] 256-bits</td>
<td>256-bits</td>
<td>128-bits</td>
</tr>
</tbody>
</table>

The accelerator supports the block cipher modes approved by NIST SP800-38A.

- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)
- Cipher Feedback (CFB)
- Output Feedback (OFB)
- Counter (CTR)

In the simplest mode Electronic Code Book (ECB), each data block is simply encrypted or decrypted using the cipher. A side effect of this is that identical data blocks will encrypt to the same ciphertext. Various modes of operation are used that chain or feedback ciphertext from the previous block to seed the next encryption operation. This causes identical, plain-text data blocks to encrypt to different ciphertexts.
For the CFB mode of operation, the mode size is equal to the block size. 128-bit CFB is supported for AES, and 64-bit CFB is supported for TDEA. 1-bit CFB and 8-bit CFB are not supported. For the CTR mode of operation, the lower 32-bits of the initial vector increment.

Block cipher operations set `CRYPTO_CTRL.cph_done` = 1 and when complete.

### 23.2.1 Cipher Key Storage and Initialization

Block cipher operations require a user-supplied cipher key to be loaded into `CIPHER_KEY[7:0]` before any algorithm can be executed. The cipher key is loaded into non-volatile memory during a secure bootloader session, and restored to the `AES_KEY[3:2]`, registers following a power-on reset.

The length of the key is dependent on the specific algorithm used.
The following procedure is required to load a key of 128 bits or less:

1. Clear \( \text{CRYPTO_CTRL}.\text{done} = 0 \) and \( \text{CRYPTO_CTRL}.\text{dma_done} = 0 \).
2. Set \( \text{CIPHER_CTRL}.\text{src} = 0b01 \) to copy the contents of AES_KEY_2 into \( \text{CIPHER_KEY}_0 \).
3. Poll until hardware sets \( \text{CRYPTO_CTRL}.\text{dma_done} = 1 \).
4. Clear \( \text{CRYPTO_CTRL}.\text{done} = 0 \) and \( \text{CRYPTO_CTRL}.\text{dma_done} = 0 \).

The following procedure is required for key lengths greater than 128 bits:

1. Clear \( \text{CRYPTO_CTRL}.\text{done} = 0 \) and \( \text{CRYPTO_CTRL}.\text{dma_done} = 0 \).
2. Set \( \text{CIPHER_CTRL}.\text{src} = 0b01 \) to copy the contents of AES_KEY_2 into \( \text{CIPHER_KEY}_0 \).
3. Poll until hardware sets \( \text{CRYPTO_CTRL}.\text{dma_done} = 1 \).
4. Clear \( \text{CRYPTO_CTRL}.\text{done} = 0 \) and \( \text{CRYPTO_CTRL}.\text{dma_done} = 0 \).
5. Set \( \text{CIPHER_CTRL}.\text{src} = 0b11 \) to copy the contents of AES_KEY_3 into \( \text{CIPHER_KEY}_0 \).
6. Poll until hardware sets \( \text{CRYPTO_CTRL}.\text{dma_done} = 1 \).
7. Clear \( \text{CRYPTO_CTRL}.\text{done} = 0 \) and \( \text{CRYPTO_CTRL}.\text{dma_done} = 0 \).

### 23.2.2 Operation

The cipher algorithm and mode of operation are set in the cipher control register. The cipher key must be loaded before starting a block cipher operation. The cipher starts operating once the FIFO is full.

1. Reset the cryptographic accelerator by setting \( \text{CRYPTO_CTRL}.\text{rst} = 1 \).
2. Poll until hardware sets \( \text{CRYPTO_CTRL}.\text{rdy} = 1 \).
3. Select the cipher algorithm operation using \( \text{CIPHER_CTRL}.\text{cipher} \).
4. Select the mode of operation using \( \text{CIPHER_CTRL}.\text{mode} \).
5. Select encryption or decryption mode \( \text{CIPHER_CTRL}.\text{enc} \).
6. Load \( \text{CIPHER_INIT}_0, \text{CIPHER_INIT}_1, \text{CIPHER_INIT}_2, \text{CIPHER_INIT}_3 \) with the initial vector if using CBC, CFB, OFB or counter modes.
7. Set \( \text{CRYPTO_CTRL}.\text{rdsrc} = 0b01 \) to select the read FIFO source as DMA
8. Set \( \text{CRYPTO_CTRL}.\text{wrsrc} \) to 0b01 to select the cipher output FIFO source as DMA
9. Load DMA source address to \( \text{DMA_SRC} \).
10. Load DMA destination address to \( \text{DMA_SRC} \).
11. Load DMA count to \( \text{DMA_CNT} \).

At the end of the DMA count:

- \( \text{CRYPTO_CTRL}.\text{done} = 1 \)
- \( \text{CRYPTO_CTRL}.\text{dma_done} = 1 \)
- \( \text{CRYPTO_CTRL}.\text{cph_done} = 1 \)

An interrupt will be generated if \( \text{CRYPTO_CTRL}.\text{int} = 1 \)

### 23.3 Hash Function Accelerator

A hash algorithm takes an input message of arbitrary length and summarizes it in a fixed-length message. The hash function accelerator executes the SHA algorithms listed in Table 23-3. Hash Functions.
Table 23-3. Hash Functions

<table>
<thead>
<tr>
<th>ALGORITHM</th>
<th>DIGEST LENGTH</th>
<th>EFFECTIVE STRENGTH (NIST SP800-57)</th>
<th>BLOCK SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1</td>
<td>160 bits</td>
<td>63 bits</td>
<td>512 bits</td>
</tr>
<tr>
<td>SHA-224</td>
<td>224 bits</td>
<td>112 bits</td>
<td>512 bits</td>
</tr>
<tr>
<td>SHA-256</td>
<td>256 bits</td>
<td>128 bits</td>
<td>512 bits</td>
</tr>
<tr>
<td>SHA-384</td>
<td>384 bits</td>
<td>192 bits</td>
<td>1024 bits</td>
</tr>
<tr>
<td>SHA-512</td>
<td>512 bits</td>
<td>256 bits</td>
<td>1024 bits</td>
</tr>
</tbody>
</table>

Figure 23-4. Block Cipher Diagram

Data is processed in 512-bit or 1024-bit blocks. After every 16 or 32 words are written to the hash block, several cycles are needed to complete the hash of that block. The number of cycles needed to compute the hash is dependent upon the number of rounds in the algorithm. The first 16 rounds are calculated as the 16 or 32 words are written to the hash block. The remaining rounds are calculated at a rate of one clock cycle per round.

The integrated DMA can fetch data for the hash accelerator. Calculation of the hash can occur in parallel with the block cipher as long as only one of the operations uses the DMA.

Setting the `HASH_CTRL.init` bit will seed the hash accelerator with the secure hash constants required by security validation requirement. Contact Maxim for specific information about the seeding process and its comparability with the latest security requirements for NIST FIPS and other certifications.

### 23.3.1 Last Message Block Padding

Once all data is written to the cryptographic data register, the final message block must be padded according to the FIPS Publication 180 standard. The standard requires that the bit length be appended to the end of the message. The length of the message in binary representation is contained in the HASH Message Size Registers `HASH_MSG_SZ_0`. Prior to hashing the last message data, set `HASH_CTRL.last` along with the `HASH_MSG_SZ` registers. The `HASH_MSG_SZ` value is automatically padded to the last message block.
For SHA-1 and SHA-256, a single bit equal to 1 is appended to the end of the message. The message is then padded using software with zeros until 64 bits remain in the last 512-bit block. If less than 64 bits remain, then zeros are appended to the message until 64 bits remain in the next 512-bit block. The HASH_CTRL.laast field is set along with HASH_MSG_SZ.[1:0]. Hardware appends values in these registers to the last 64 bits of the final message block.

For SHA-384 and SHA-512, a single bit equal to 1 is appended to the end of the message. The message is then padded with zeros until 128 bits remain in the last 1024-bit block. If less than 128 bits remain, then zeros are appended to the message until 128 bits remain in the next 1024-bit block. The HASH_CTRL.last is set along with HASH_MSG_SZ.0.

The automatic padding feature is used in terms of message bytes, not bits. Therefore, the HASH_MSG_SZ[1:0] are expressed in bytes. In addition, the feature automatically generates an additional padding-only block if the last block of message data cannot accommodate the 64- or 128-bit padding block.

As an exception, attempting to hash a 0 message-size block must include a dummy write to the HASH message digest register.

Hash operations using the CMDA set CRYPTO_CTRL.hsh_done=1 and CRYPTO_CTRL.dma_done = 1, and CRYPTO_CTRL.done=1 when complete.

## 23.4 CRC Engine (Galois Field Accelerator)

Registers pertaining to CRC functionality are included in the TPU register space, but are covered in a separate chapter. See the CRC chapter for information about using the CRC.

## 23.5 Hamming Code Accelerator

The Hamming code accelerator calculates an Error Correction Code (ECC) on a block of data. Hamming codes are capable of correcting single-bit errors. You can include an extra parity bit to detect two-bit errors. This is commonly referred to as Signal Error Correction, Double Error Detection (SEC-DED). Three errors masquerade as a correctable, single-bit error.

Multi-level Cell (MLC) Flash memories require Error Correction Codes that can correct multiple-bit errors since a corrupt cell can have multiple bit errors.

The hardware can calculate ECCs for a block of data up to 216 bits in length (8kB), but software implementations can extend this to any length. Because the Hamming code can only correct a single-bit error, increasing the block size increases the likelihood of multiple errors that are uncorrectable. The Hamming code generator in the cryptographic accelerator generates even parity on even halves of bit groups.

If you want the parity of the odd halves of the bit groups, XOR the parity of the even halves with the parity of the entire array. If the parity of the entire array is odd, the parity of the odd halves is the inverse of the parity of the even halves. If the parity of the entire array is even, the parity of the odd halves is identical to the parity of the even halves.
For a block of data 2^n bits in length, n+1 ECC bits are needed for single-bit error correction (ECC B0 through Bn = n+1 ECC bits). ECC bit n indicates parity of the entire array. If it is different than the stored ECC bit n, it indicates an error in the data array. You can determine the location of the error using the lower n ECC bits (B0 to Bn-1).

To determine if an error occurred, XOR the saved and calculated ECC bits. If the result is zero, no error occurred. If the result of the ECC XOR operation is not zero, then the location of the failing bit is given by the inverse of the ECC XOR result.

failing_bit_location = ~(ecc_saved ^ ecc_calculated)

An error in the most significant ECC bit n masquerades as an error in the most significant bit of the data. To properly detect and correct an error in the MSB of the ECC, include an extra ECC bit (n+2 ECC bits). These two most significant ECC bits should be identical if they are error free. If they are different, you can determine the failing ECC bit by Xoring the saved and calculated ECC bits.
You can save this extra ECC bit at the cost of an additional parity calculation. If the MSB of the data is set, the ECC parity bits should be XOR’d with 2n+1-1 (n 1s). This effectively swaps the MSB of the data with the MSB of the ECC for purposes of parity calculation. When examining the result, if the ECC calculation indicates the MSB of the data is corrupt (the result is 2n+1-1), then the error is really in the MSB of the ECC. If the result indicates the error is in the MSB of the ECC (the result is 2n+1), then the error is really in the MSB of the data.

ECC bits n and above are all identical. They all indicate the parity of the entire array. To use a block size of 64k bits (8k bytes), the parity bit of the entire array (bit 16) can be duplicated to obtain the higher-order ECC bits.

You can generate Hamming codes over larger blocks using software. After the Hamming code is generated for an 8k block of data, software should examine the parity of the block of data just calculated (bit 16 of the ECC register). If the parity of the block is odd (parity bit is set), software should XOR additional software-maintained ECC bits with the inverse of the block address. If the parity of the block is even (parity bit of the block is clear), software does not need to modify the additional ECC bits. You should reset the parity of the block (bit 16 of the ECC register) for each block, but the remaining ECC bits (B0-B15) should remain unchanged and accumulate their respective XORs throughout the entire array.

To achieve Double Error Detection (but not correction), include another ECC parity bit. If the result of the ECC XOR is not zero, then this extra ECC bit should also be set indicating an error has occurred. If this bit is clear after the ECC XOR operation, it means an even number of errors has occurred, and the data is not correctable. This does not mean this extra ECC bit detects an even number of bit errors. It just indicates that an even number of errors have occurred. It is possible for an even number of bit errors to masquerade as valid data. This extra ECC bit is capable of detecting two-bit errors. If an odd number of bit errors occur, they always masquerade as a single correctable bit error. This is a limitation of using parity to indicate error. Parity can only detect an odd number of errors. For stronger error detection, a Cyclic Redundancy Check (CRC) is used.

The lower n ECC bits (B0 to Bn-1) are all that is needed to determine the location of a single-bit error in the data. The next ECC bit (which is the parity of the entire array) is simply used to indicate there is an error in the data that needs to be corrected. Subsequent ECC bits are only needed to detect an error in this added ECC bit or to detect double-bit errors. If an alternative error detection scheme is used, such as a CRC, then only n ECC bits are needed to find the location of a single-bit error in the data array.

Because all CRCs with at least two terms in the generator polynomial will detect all single-bit errors, ECC bit n, which is used to determine if there is a single-bit error in the array by checking parity is unnecessary. A CRC polynomial with an even number of terms has the parity polynomial (x+1) as one of its factors and checks parity as well. A CRC of sufficient length is also capable of detecting all two-bit errors, making an ECC bit added for double-error detection unnecessary as well.

Because a single-bit error in the CRC is catastrophic, the CRC should also be protected with ECC bits. If the CRC is appended to the data for the Hamming code calculation, you can protect it with the same set of ECC bits used to protect the data by including one additional ECC bit to account for the increase in block size.

Most manufacturers of NAND Flash memories have application notes that describe the calculation of a variation of Hamming codes. These application notes calculate parity on both the odd and even halves of the bit groups. As a result, they require 2n ECC bits, almost twice as many as necessary. If the even half is XOR’d with the odd half, the result is the parity of the entire array. Instead of storing parity for both the even halves and odd halves of all bit groups (which requires 2n ECC bits), an implementation only really needs to store the parity of the entire array (n+1 ECC bits). You can determine the odd half of ECC bits by XORing the even half of ECC bits with the parity of the entire array. The parity of the entire array is just the next bit in the ECC register, so saving n+1 bits from this register saves the parity of the entire array. Storing both the odd and even parity halves offers no more error protection than the methods described above. For both methods, an odd number of errors masquerades as a correctable single-bit error, and all double-bit errors are detected.

### 23.6 Modular Arithmetic Accelerator

The Modular Arithmetic Accelerator (MAA) is a dedicated hardware module that performs high-speed calculations that are key parts of asymmetric cryptographic algorithms based on DSA, RSA, and ECDSA algorithms.
The MAA features include:

- Supports up to a 2048-bit operand size
- Performs up to 2048-bit modular exponentiation \((ae \mod m)\)
- Performs up to 2048-bit modular multiplication \((a\times b \mod m)\)
- Performs up to 2048-bit modular square \((b^2 \mod m)\)
- Performs up to 2048-bit modular square followed by modular multiply \(((b^2 \mod m) \times a \mod m)\)
- Performs up to 2048-bit modular addition \((a+b \mod m)\)
- Performs up to 2048-bit modular subtraction \((a-b \mod m)\)
- Optimized calculation mode to maximize speed
- Non-optimized calculation mode to maximize security

These operations can be combined to perform modular inversion and modular reduction.

The MAA operates independently from the processor except when the processor is reading or writing the control register, or when it is used to load/unload the data in the specified data memory segment.

Operands and parameters are stored in a dedicated 768x16 data memory segment.

The MAA does not use the CMDA engine.

### 23.6.1 Operation

The MAA control register \texttt{MAA_CTRL} provides option control on arithmetic operations, data partition, and status bits for start/busy. This starting address of most parameters is configurable within the memory instance is configurable using the Memory Assignment (AMA, BMA, RMA and TMA) fields. Per FIPS' big-endian data convention, the most significant byte or sub-word of a multi-byte word is loaded first and stored at the lowest storage location.

<table>
<thead>
<tr>
<th>SEGMENT</th>
<th>MEMORY SEGMENT ASSIGNMENT</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>MAA_CTRL_ama</td>
<td>Multiplier/Operand A</td>
</tr>
<tr>
<td>b</td>
<td>MAA_CTRL_bma</td>
<td>Multiplicand/Operand B</td>
</tr>
<tr>
<td>e</td>
<td>Fixed</td>
<td>Exponent</td>
</tr>
<tr>
<td>m</td>
<td>Fixed</td>
<td>Modulus (only required for exponentiation operations)</td>
</tr>
<tr>
<td>t</td>
<td>MAA_CTRL_tma</td>
<td>Temporary storage. The data in this segment is undefined.</td>
</tr>
<tr>
<td>r</td>
<td>MAA_CTRL_rma</td>
<td>Result value</td>
</tr>
</tbody>
</table>

The Modular Accelerator Word Size field \texttt{MAA\_MAWS.msgsz} defines the calculation size of a modular operation. The content of the register presents the number of bits for the modular operation. For example, to perform a 1007-bit (03EFh) modular operation, you would need to set this register to 03EFh prior to setting STC.

Valid word size is from 1 to 2048. The accelerator does not start if \texttt{MAA\_MAWS.msgsz} is invalid.

The \texttt{CRYPTO\_CTRL\_maa\_done} bit is set to 1 after the completion of an MAA operation or when an error occurs. An interrupt is generated if \texttt{CRYPTO\_CTRL\_int} is set. Software clears the bit by writing a 0.

### 23.6.2 MAA Memory

A dedicated, 1535-byte MAA memory begins at offset 0x0100. This memory holds the operands and intermediate values for MAA calculations. The memory space is subdivided into logical segments based on the size of the calculation.

The MAA memory makes a distinction between memory instances and memory segments when assigning parameter location. The following restrictions apply when assigning parameters in the MAA memory.

Only one parameter can be located in each memory instance.
The modulus (m) is always stored in memory instance 5.

When an exponentiation operation is selected, the exponent (e) is always stored in memory instance 4. If another operation is selected, memory instance 4 is free to hold another parameter.

Parameters m, b, t and r must be stored in different memory instances (not segments) even if MAA_MAWS[msgsz < 1024] bits. For example, if b is stored in memory instance 0, then neither t nor r is stored in memory instance 1 when word size is smaller than 1024. Each memory instance is 256 bytes.

Table 23-5. MAA Memory Segments and Locations

<table>
<thead>
<tr>
<th>MEMORY INSTANCE</th>
<th>MEMORY SEGMENT (MAWS &gt;= 1024)</th>
<th>MEMORY SEGMENT (MAWS &lt;1024)</th>
<th>DEDICATED FUNCTION</th>
<th>ADDRESS OFFSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>xMA[3:0]</td>
<td>0</td>
<td>0</td>
<td>None</td>
<td>0x0100 – 0x017F</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>None</td>
<td>0x0180 – 0x01FF</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>None</td>
<td>0x0200 – 0x027F</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>None</td>
<td>0x0280 – 0x02FF</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4</td>
<td>None</td>
<td>0x0300 – 0x037F</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5</td>
<td>None</td>
<td>0x0380 – 0x03FF</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>6</td>
<td>None</td>
<td>0x0400 – 0x047F</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>7</td>
<td>None</td>
<td>0x0480 – 0x04FF</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>8</td>
<td>Exponent</td>
<td>0x0500 – 0x057F</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>9</td>
<td>Modulus</td>
<td>0x0580 – 0x05FF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>-</td>
<td>0x0600 – 0x06FF</td>
</tr>
</tbody>
</table>

### 23.6.2.1 Memory Blinding

Memory blinding is an effective cryptographic technique that increases the difficulty of side channel attacks against cryptographic operations. In a poorly designed application, the MAA memory segments are in a fixed location, leaving them vulnerable to timing and power analysis attacks.

The memory blinding features provides the option of shifting the starting position of the a, b, e, and m parameters within their respective memory instances. Although you can alter the starting position, the entire parameter is still stored within a single memory instance. Each parameter can be independently configured with the default “unblinded” and three blinded starting positions. Memory blinding for each parameter is controlled by its corresponding Memory Select bits (AMS, BMS, EMS, MMS).

Table 23-6. MAA Memory Blinding Example (Memory Instance 0, MAWS > 1024)

<table>
<thead>
<tr>
<th>xMS[1:0]</th>
<th>SHIFT</th>
<th>ADDRESS OFFSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00x0000</td>
<td>0x0100 ... 0x01FF (no blinding)</td>
</tr>
<tr>
<td>01</td>
<td>00x0040</td>
<td>0x0140 ... 0x01FF, 0x0100 ... 0x013F</td>
</tr>
<tr>
<td>10</td>
<td>00x0080</td>
<td>0x0180 ... 0x01FF, 0x0100 ... 0x017F</td>
</tr>
<tr>
<td>11</td>
<td>00x00C0</td>
<td>0x01C0 ... 0x01FF, 0x0100 ... 0x01BF</td>
</tr>
</tbody>
</table>

### 23.6.2.2 MAA Clock Source

The MAA operates at either the LPCLK or LPCLK/2 frequency as determined by the GCR_CLKCL.ccd field and the frequency specified in the relevant data sheet.
23.6.2.3  Optimized Calculations

The optimized calculation control bit (MAA_CTRL.ocal) allows the software to optimize the speed of an exponentiation by skipping unnecessary multiply operations when the corresponding exponent bit is a 0. The bit defaults to 0, forcing the MAA to operate in a non-optimized mode. The non-optimized mode skips the leading zeros of the exponent and starts square/multiply operations when a 1 is detected. From this point on, multiply operation are completed for every exponent bit, regardless of its logical value.

Optimization is highly discouraged, as it leaves the device vulnerable to power analysis attacks.

23.6.2.4  MAA Operand Sizes

MAA operand size is specified by MAA_MAWS.msgsz. Valid values are from 1 to 2048. This value specifies valid ranges for a, b, e, and m. a, b, and e, can have any values between 0 and 2MAWS-1 inclusive as long as their number of bits are less than those of m. For exponentiation operation, however, b memory must contain the value of 1, and e cannot be 0.

The m memory, which holds the modulus, has a value from 2MAWS-1 up to 2MAWS-1. For example, for a 16-bit MAA_MAWS.msgsz =0x10, it has a value from 0x8000 (32768) up to 0xFFFF (65535).

If any parameter exceeds Word Size (MAWS), an invalid result is generated without issuing an MAA error status. It is up to application to check for invalid word sizes.

The MAA operands are stored as 64-bit blocks. For all the memories, operands must be zero padded to the 64-bit block boundary. There is no restriction on values stored in the memories beyond this boundary. For example, MAA_MAWS.msgsz =65, and operand=01 xxxx xxxx xxxx xxxxh. In this example, MAA operands are stored as two 64-bit blocks. Bit[63:0] contains the lower 64 bits. Bit[127:66] are zero padded while bit[65]=1. Data in the words above where the 0000 0000 0000 0001h is stored can have any value.

For most calculations, memory segments t and r are used to store intermediate values during round operations and can contain the same value at the final operation.

For square-multiply and exponentiation, memory segment b is an additional temporary storage area during round operations.

23.7  True Random Number Generation

The Maxim-supplied Universal Cryptographic Library (UCL), implements a random bit generator that may be compliant with the entropy requirements of commonly used data security validations. The general information in this section is provided only for completeness; Maxim will work directly with the customer’s chosen security validation laboratory and provide that all the information required for validation directly to that agency.

Entropy is continuously generated by the TRNG. It can be retrieved in groups of 32- and 128 bits through one or four consecutive reads of the 32-bit TRNG_DATA register. Hardware sets TRNG_CN.rng_is = 1 when 32 bits are available in the TRNG FIFO, and sets TRNG_CN.rng_4is = 1 when 128 bits are in the FIFO. Software must set TRNG_CN.rng_isc = 1, which clears TRNG_CN.rng_is and TRNG_CN.rng_i4s, before reading from the TRNG register.

An optional interrupt can be generated if TRNG_CN.rng_je = 1 when hardware sets TRNG_CN.rng_4is = 1.

23.8  Registers

Access to specific registers is controlled to prevent software reading from or writing to the registers while they are performing specific functions.
23.8.1 Write Access

The MAA_CTRL register can only be written to when the MAA is idle (MAA_CTRL.stc = 0). See the bit description for more details on its write access limitation.

The CRYPTO_CTRL.flag_mode field determines the method used to clear the CRYPTO_CTRL.dma_done, CRYPTO_CTRL.gls_done, CRYPTO_CTRL.hsh_done, and CRYPTO_CTRL.cph_done flags.

Writing to the MAA_MAWS or MAA_CTRL registers while MAA_CTRL.stc = 1 will generate an error and set the MAA_CTRL.maaer bit. The current operation will be terminated and the MAA_CTRL.stc bit will be cleared.

23.8.2 Read Access

Reading from the MAA memory while MAA_CTRL.stc = 0 will return invalid results, but it will not generate an error.

Table 23-7. Cryptographic Registers, Offsets and Descriptions

<table>
<thead>
<tr>
<th>Register</th>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRYPTO_CTRL</td>
<td>0x0000</td>
<td>Cryptographic Control Register</td>
</tr>
<tr>
<td>CIPHER_CTRL</td>
<td>0x0004</td>
<td>Cipher Control Register</td>
</tr>
<tr>
<td>HASH_CTRL</td>
<td>0x0008</td>
<td>Hash Control Register</td>
</tr>
<tr>
<td>CRC_CTRL</td>
<td>0x000C</td>
<td>CRC Control Register</td>
</tr>
<tr>
<td>DMA_SRC</td>
<td>0x0010</td>
<td>Cryptographic DMA Source Register</td>
</tr>
<tr>
<td>DMA_DEST</td>
<td>0x0014</td>
<td>Cryptographic DMA Destination Register</td>
</tr>
<tr>
<td>DMA_CNT</td>
<td>0x0018</td>
<td>Cryptographic DMA Count Register</td>
</tr>
<tr>
<td>MAA_CTRL</td>
<td>0x001C</td>
<td>MAA Control Register</td>
</tr>
<tr>
<td>CRYPTO_DIN_0</td>
<td>0x0020</td>
<td>Cryptographic Data In [31:0]</td>
</tr>
<tr>
<td>CRYPTO_DIN_1</td>
<td>0x0024</td>
<td>Cryptographic Data In [63:32]</td>
</tr>
<tr>
<td>CRYPTO_DIN_2</td>
<td>0x0028</td>
<td>Cryptographic Data In [95:64]</td>
</tr>
<tr>
<td>CRYPTO_DIN_3</td>
<td>0x002C</td>
<td>Cryptographic Data In [127:96]</td>
</tr>
<tr>
<td>CRYPTO_DOUT_0</td>
<td>0x0030</td>
<td>Cryptographic Data Out [31:0]</td>
</tr>
<tr>
<td>CRYPTO_DOUT_1</td>
<td>0x0034</td>
<td>Cryptographic Data Out [63:32]</td>
</tr>
<tr>
<td>CRYPTO_DOUT_2</td>
<td>0x0038</td>
<td>Cryptographic Data Out [95:64]</td>
</tr>
<tr>
<td>CRYPTO_DOUT_3</td>
<td>0x003C</td>
<td>Cryptographic Data Out [127:96]</td>
</tr>
<tr>
<td>CRC_POLY</td>
<td>0x0040</td>
<td>CRC Polynomial Register</td>
</tr>
<tr>
<td>CRC_VAL</td>
<td>0x0044</td>
<td>CRC Value Register</td>
</tr>
<tr>
<td>CRC_PRNG</td>
<td>0x0048</td>
<td>Pseudo-Random Number Generator Register</td>
</tr>
<tr>
<td>HAM_ECC</td>
<td>0x004C</td>
<td>Hamming ECC Register</td>
</tr>
<tr>
<td>CIPHER_INIT_0</td>
<td>0x0050</td>
<td>Cipher Initial Vector[31:0]</td>
</tr>
<tr>
<td>CIPHER_INIT_1</td>
<td>0x0054</td>
<td>Cipher Initial Vector[63:32]</td>
</tr>
<tr>
<td>CIPHER_INIT_2</td>
<td>0x0058</td>
<td>Cipher Initial Vector[95:64]</td>
</tr>
<tr>
<td>CIPHER_INIT_3</td>
<td>0x005C</td>
<td>Cipher Initial Vector[127:96]</td>
</tr>
<tr>
<td>CIPHER_KEY_0</td>
<td>0x0060</td>
<td>Cipher Key [31:0]</td>
</tr>
<tr>
<td>CIPHER_KEY_1</td>
<td>0x0064</td>
<td>Cipher Key [63:32]</td>
</tr>
<tr>
<td>CIPHER_KEY_2</td>
<td>0x0068</td>
<td>Cipher Key [95:64]</td>
</tr>
<tr>
<td>CIPHER_KEY_3</td>
<td>0x006C</td>
<td>Cipher Key [127:96]</td>
</tr>
<tr>
<td>CIPHER_KEY_4</td>
<td>0x0070</td>
<td>Cipher Key [159:128]</td>
</tr>
<tr>
<td>CIPHER_KEY_5</td>
<td>0x0074</td>
<td>Cipher Key [191:160]</td>
</tr>
</tbody>
</table>
### register details

**table 23-8: cryptographic control register**

<table>
<thead>
<tr>
<th>cryptographic control register</th>
<th>crypto_ctrl</th>
<th>[0x0000]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bits</strong></td>
<td>name</td>
<td>access</td>
</tr>
<tr>
<td>31</td>
<td>done</td>
<td>R/W</td>
</tr>
</tbody>
</table>

- This bit is set whenever hardware completes an MMA, cipher or hash operation and sets the corresponding “done” bit in CRYPTO_CTRL.[27:25]. This bit remains set until cleared by software. Writing 0 to one or more of the bits in CRYPTO_CTRL.[27:24] will not effect this bit.

- Setting the CRYPTO_CTRL.dmanemsk bit to 1 will cause this bit to be set to 1 when hardware sets the CRYPTO_CTRL.dma_done bit.

- 0: No cryptographic operations have completed since this bit was cleared.
- 1: One or more cryptographic operations are complete.

| 30 | rdy | RO | 1 | cryptographic block ready |

- Hardware clears this status bit to 0 when software initiates a reset of the cryptographic accelerator by setting the CRYPTO_CTRL.rst.bit. Software must poll this bit until it is set to 1 by hardware, indicating cryptographic accelerator is again ready for use.

- 0: Cryptographic accelerator reset in progress.
- 1: Cryptographic accelerator ready for use
## Cryptographic Control Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>err</td>
<td>R</td>
<td>0</td>
<td>AHB Bus Error&lt;br&gt;This bit is set if the DMA attempts to access non-existent or protected memory on the AHB bus. This bit can only be cleared by resetting the cryptographic accelerator block.&lt;br&gt;0: No error&lt;br&gt;1: Cryptographic accelerator DMA bus error</td>
</tr>
<tr>
<td>28</td>
<td>maa_done</td>
<td>R/W</td>
<td>0</td>
<td>MAA Operation Done&lt;br&gt;Clear this bit before starting a new MAA operation. This bit is read only while the MAA is in progress. This bit is the opposite of \texttt{MAA_CTRL}.stc.&lt;br&gt;0: MAA operation in progress&lt;br&gt;1: Last MAA operation done</td>
</tr>
<tr>
<td>27</td>
<td>cph_done</td>
<td>R/W</td>
<td>0</td>
<td>Cipher Done&lt;br&gt;This bit is set when a block cipher encryption/decryption operation is complete. Clear this bit before starting a cipher operation.&lt;br&gt;0: Not done&lt;br&gt;1: Last cipher operation done</td>
</tr>
<tr>
<td>26</td>
<td>hsh_done</td>
<td>R/W</td>
<td>0</td>
<td>Hash Done&lt;br&gt;This bit is set to 1 when the cryptographic accelerator has completed a SHA operation is complete. Clear this bit before starting the next hash operation.&lt;br&gt;0: Not done&lt;br&gt;1: SHA/hash operation done</td>
</tr>
<tr>
<td>25</td>
<td>gls_done</td>
<td>R/W</td>
<td>0</td>
<td>Galois Done&lt;br&gt;FIFO is full, and CRC or the Hamming Code Generator is enabled. Clear this bit before starting a CRC operation.&lt;br&gt;0: Not done&lt;br&gt;1: Operation done</td>
</tr>
<tr>
<td>24</td>
<td>dma_done</td>
<td>R/W</td>
<td>0</td>
<td>DMA Done&lt;br&gt;DMA write/read operation is complete. Clear this bit before starting a DMA operation.&lt;br&gt;0: Not done&lt;br&gt;1: Operation done</td>
</tr>
<tr>
<td>23:16</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use&lt;br&gt;Do not modify this field from its default value.</td>
</tr>
<tr>
<td>15</td>
<td>dmadnemsk</td>
<td>R/W</td>
<td>0</td>
<td>DMA Done Flag Mask&lt;br&gt;This field prevents the \texttt{CRYPTO_CTRL}.dma_done flag from setting the \texttt{CRYPTO_CTRL}.done flag. The \texttt{CRYPTO_CTRL}.dma_done flag will still be set.&lt;br&gt;0: \texttt{CRYPTO_CTRL}.dma_done flag will not set \texttt{CRYPTO_CTRL}.done&lt;br&gt;1: \texttt{CRYPTO_CTRL}.dma_done flag will also set \texttt{CRYPTO_CTRL}.done</td>
</tr>
<tr>
<td>14</td>
<td>flag_mode</td>
<td>R/W</td>
<td>0</td>
<td>Done Flag Mode&lt;br&gt;This bit provides legacy support for the access behavior of the done flags. It should not be changed from its default value.&lt;br&gt;0: (Default) Unrestricted write(0 or 1) of \texttt{CRYPTO_CTRL}[27:24]&lt;br&gt;1: Access to \texttt{CRYPTO_CTRL}[27:24] is “write 1 to clear/write 0 no effect”</td>
</tr>
<tr>
<td>13:12</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use&lt;br&gt;Do not modify this field from its default value.</td>
</tr>
<tr>
<td>11:10</td>
<td>rdsrc</td>
<td>R/W</td>
<td>0</td>
<td>Read FIFO Source Select&lt;br&gt;This field selects the source of the read FIFO data.&lt;br&gt;0b00: DMA disabled&lt;br&gt;0b01: DMA or APB&lt;br&gt;0b10: RNG&lt;br&gt;0b11: Reserved</td>
</tr>
</tbody>
</table>
### Cryptographic Control Register

<table>
<thead>
<tr>
<th>BITS</th>
<th>NAME</th>
<th>ACCESS</th>
<th>RESET</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:8</td>
<td>wsrc</td>
<td>R/W</td>
<td>0</td>
<td>Write FIFO Source Select</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This field determines the source of the write FIFO data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00: None</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01: Cipher Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b10: Read FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b11: Reserved</td>
</tr>
<tr>
<td>7</td>
<td>wait_pol</td>
<td>R/W</td>
<td>0</td>
<td>Wait Pin Polarity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This feature is not implemented in this device. Do not change this bit from its default value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Active low</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Active high</td>
</tr>
<tr>
<td>6</td>
<td>wait_en</td>
<td>R/W</td>
<td>0</td>
<td>Wait Pin Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This feature is not implemented in this device. Do not change this bit from its default value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Enabled. CMDA will be halted when the pin is in its active state.</td>
</tr>
<tr>
<td>5</td>
<td>bsi</td>
<td>R/W</td>
<td>0</td>
<td>Byte Swap Input</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: No byte swap occurs if there is not a full word.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No effect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Byte swap input.</td>
</tr>
<tr>
<td>4</td>
<td>bso</td>
<td>R/W</td>
<td>0</td>
<td>Byte Swap Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Note: No byte swap occurs if there is not a full word.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No effect.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Byte swap output.</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field from its default value.</td>
</tr>
<tr>
<td>2</td>
<td>src</td>
<td>R/W</td>
<td>0</td>
<td>Source Select</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit selects the hash function and CRC generator input source.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Input FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Output FIFO</td>
</tr>
<tr>
<td>1</td>
<td>int</td>
<td>R/W</td>
<td>0</td>
<td>Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Generates an interrupt when done or error set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Interrupt disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Interrupt asserted when CRYPTO_CTRL.done is set.</td>
</tr>
<tr>
<td>0</td>
<td>rst</td>
<td>R/W</td>
<td>0</td>
<td>Reset Cryptographic Accelerator</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting this bit initiates an internal reset of the cryptographic accelerator. Software must poll the CRYPTO_CTRL.rdy bit to determine when the reset process is complete.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>All cryptographic internal states and related registers are reset to their default reset values. Control register such as CRYPTO_CTRL, CIPHER_CTRL, HASH_CTRL,CRC_CTRL,, MAA_CTRL (with the exception of the STC bit), HASH_MSG_SZ_3:0, and MAA_MAWS retain their values. This bit automatically clears itself after one cycle.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: No effect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Reset cryptographic accelerator</td>
</tr>
</tbody>
</table>

### Table 23-9: Cipher Control Register

<table>
<thead>
<tr>
<th>Cipher Control Register</th>
<th>CIPHER_CTRL</th>
<th>[0x0004]</th>
</tr>
</thead>
<tbody>
<tr>
<td>BITS</td>
<td>NAME</td>
<td>ACCESS</td>
</tr>
<tr>
<td>31:11</td>
<td>-</td>
<td>RO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Cipher Control Register

<table>
<thead>
<tr>
<th>BITS</th>
<th>NAME</th>
<th>ACCESS</th>
<th>RESET</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| 10:8 | mode | R/W    | 0     | Mode Select  
Operating mode of block cipher or memory operation.  
- 0b000: ECB  
- 0b001: CBC  
- 0b010: CFB  
- 0b011: OFB  
- 0b100: CTR  
- Others: Reserved |
| 7    | -    | RO     | 0     | Reserved for Future Use  
Do not modify this field from its default value. |
| 6:4  | cipher | R/W | 0 | Block Cipher Operation Select  
Symmetric Block Cipher algorithm selection or memory operation. Clear these bits before starting any other operation of the cryptographic accelerator.  
- 0b000: Disabled  
- 0b001: AES-128  
- 0b010: AES-192  
- 0b011: AES-256  
- 0b100: DES  
- 0b101: TDEA  
- 0b110: Reserved  
- 0b111: Reserved |
| 3:2  | src  | R/W    | 0     | Source of Cipher Key  
This indicates the source of the key data to be loaded into CIPHER_KEY[7:0] registers when the CIPHER_CTRL.key bit is set. Each field setting loads 128 bits. The load cipher key operation must be performed twice (once with 0b10 and again with 0b11) to load a full 256-bit key.  
- 0b00: CIPHER_KEY[7:0]  
- 0b01: CIPHER_KEY[7]  
- 0b10: AES_KEY2  
- 0b11: AES_KEY3 |
| 1    | key  | R/W1O  | 0     | Load Cipher Key Using CMDA  
Setting this bit initiates loading of the CIPHER_KEY[7:0] registers with the contents pointed to by CIPHER_CTRL.src. The bit must be cleared and the CRYPTO_CTRL.dma_done bit after the DMA completes loading the key.  
- 0: NOP  
- 1: Initiate key loading from DMA |
| 0    | enc  | R/W    | 0     | Encryption Mode  
Select encryption or decryption of block cipher data.  
- 0: Encrypt  
- 1: Decrypt |

### Hash Control Register

<table>
<thead>
<tr>
<th>BITS</th>
<th>NAME</th>
<th>ACCESS</th>
<th>RESET</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| 31:6 | -    | RO     | 0     | Reserved for Future Use  
Do not modify this field from its default value. |
### Hash Control Register

<table>
<thead>
<tr>
<th>BITS</th>
<th>NAME</th>
<th>ACCESS</th>
<th>RESET</th>
<th>DESCRIPTION</th>
</tr>
</thead>
</table>
| 5    | last  | R/W    | 0     | **Last Message Bit**  
This bit is set along with the `HASH_MSG_SZ_3:0` register prior to hashing the last 512 or 1024-bit block of the message data. It allows automatic preprocessing of the last message padding, which includes the trailing bit 1 followed by the respective number of zero bits for the last block size and the message length represented in bytes. The bit is automatically cleared at the same time the `CRYPTO_CTRL.hsh_done` is set, designating the completion of the last message.  
0: No effect  
1: Last message data |

| 4:2  | hash  | R      | 0     | **Hash Function Selection.**  
Select the hash mode algorithm. Clear these bits before starting any other operation of the cryptographic accelerator.  
0b000: Hash disabled  
0b001: SHA-1  
0b010: SHA-224  
0b011: SHA-256  
0b100: SHA-384  
0b101: SHA-512  
0b110: Reserved  
0b111: Reserved |

| 1    | xor   | R/W    | -     | **XOR IV and Cipher Block**  
Useful when calculating HMAC to XOR the input pad and output pad. Use the feature to Load Key from CMDA.  
This bit is automatically cleared by hardware after the DMA completes loading the key. When the DMA operation is done, it sets the appropriate CMDA Done flag.  
0: No XOR  
1: XOR input with IV |

| 0    | init  | R/W    | 0     | **Initialize**  
Load `HASH_DIGEST_0 HASH_DIGEST_[15:0]` with the initial hash values based on the message digest size the initial hash values corresponding to the selected hash function.  
0: NOP  
1: Initialize hash values |

### CRC Control Register

<table>
<thead>
<tr>
<th>CRC Control Register</th>
<th>CRC_CTRL</th>
<th>[0x000C]</th>
</tr>
</thead>
<tbody>
<tr>
<td>BITS:31:6 -</td>
<td>RO</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Do not modify this field from its default value.</td>
</tr>
</tbody>
</table>
| 5        | hrst    | R/W    | 0     | **Hamming Reset**  
Reset the Hamming code ECC generator for the next block.  
0: NOP  
1: Reset Hamming Register |

| 4        | ham     | R/W    | 0     | **Hamming Code Enable**  
Enable Hamming code calculation.  
0: Hamming disabled  
1: Hamming enabled |

| 3        | ent     | R/W    | 0     | **Entropy Enable**  
This feature is not implemented in this device. Do not change this bit from its default value. |
### MAX32665-MAX32668 User Guide

#### Table 23-12: Cryptographic DMA Source Register

<table>
<thead>
<tr>
<th>bits</th>
<th>name</th>
<th>access</th>
<th>reset</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addr</td>
<td>DMA_SRC</td>
<td>R/W</td>
<td>0</td>
<td>DMA Source Address</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DMA source address for cryptographic operations.</td>
</tr>
</tbody>
</table>

#### Table 23-13: Cryptographic DMA Destination Register

<table>
<thead>
<tr>
<th>bits</th>
<th>name</th>
<th>access</th>
<th>reset</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addr</td>
<td>DMA_DEST</td>
<td>R/W</td>
<td>0</td>
<td>DMA Destination Address</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DMA destination address for cryptographic operations.</td>
</tr>
</tbody>
</table>

#### Table 23-14: Cryptographic DMA Count Register

<table>
<thead>
<tr>
<th>bits</th>
<th>name</th>
<th>access</th>
<th>reset</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>addr</td>
<td>DMA_CNT</td>
<td>R/W</td>
<td>0</td>
<td>DMA Byte Counter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DMA counter for cryptographic operations. Writing a non-zero value to this register initiates DMA-based operations.</td>
</tr>
</tbody>
</table>

#### Table 23-15: MAA Control Register

<table>
<thead>
<tr>
<th>bits</th>
<th>name</th>
<th>access</th>
<th>reset</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tma</td>
<td>MAA_CTRL</td>
<td>R/W</td>
<td>0</td>
<td>Temporary MAA Memory Assignment</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>These bits select the logical cryptographic RAM segment for parameter t.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>settings</th>
<th>MAWS &lt; 1024</th>
<th>MAWS ≥ 1024</th>
<th>segment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0b0000</td>
<td>0b0000</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0b0001</td>
<td>0b0010</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0b0010</td>
<td>0b0100</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>0b0011</td>
<td>0b0110</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>0b0100</td>
<td>0b1000</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>0b0101</td>
<td>N/A</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>0b0111</td>
<td>N/A</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

---

**CRC Control Register**

<table>
<thead>
<tr>
<th>bits</th>
<th>NAME</th>
<th>access</th>
<th>reset</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>prng</td>
<td>R/W</td>
<td>0</td>
<td>PRNG Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This feature is not implemented in this device. Do not change this bit from its default value.</td>
</tr>
<tr>
<td>1</td>
<td>msb</td>
<td>R/W</td>
<td>0</td>
<td>CRC MSB select</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit selects the order of calculating CRC on data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: LSB data first</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: MSB data first</td>
</tr>
<tr>
<td>0</td>
<td>crc</td>
<td>R/W</td>
<td>0</td>
<td>Cyclic Redundancy Check Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This feature is not implemented in this device. Do not change this bit from its default value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: CRC disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: CRC enabled</td>
</tr>
</tbody>
</table>
## MAA Control Register

<table>
<thead>
<tr>
<th>BITS</th>
<th>NAME</th>
<th>ACCESS</th>
<th>RESET</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>0b1000</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0b1001</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0b1010</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### Result Memory Assignment

These bits select the logical cryptographic RAM segment for parameter r.

<table>
<thead>
<tr>
<th>SETTINGS</th>
<th>SEGMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAWS &lt; 1024</td>
<td></td>
</tr>
<tr>
<td>0b0000</td>
<td>0b0000</td>
</tr>
<tr>
<td>0b0001</td>
<td>0b0010</td>
</tr>
<tr>
<td>0b0010</td>
<td>0b0100</td>
</tr>
<tr>
<td>0b0011</td>
<td>0b0110</td>
</tr>
<tr>
<td>0b0100</td>
<td>0b1000</td>
</tr>
<tr>
<td>0b0101</td>
<td>N/A</td>
</tr>
<tr>
<td>0b0111</td>
<td>N/A</td>
</tr>
<tr>
<td>0b1000</td>
<td>N/A</td>
</tr>
<tr>
<td>0b1001</td>
<td>N/A</td>
</tr>
<tr>
<td>0b1010</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### Multiplicand / Operand B Memory Assignment

These bits select the logical cryptographic RAM segment for parameter b.

<table>
<thead>
<tr>
<th>SETTINGS</th>
<th>SEGMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAWS &lt; 1024</td>
<td></td>
</tr>
<tr>
<td>0b0000</td>
<td>0b0000</td>
</tr>
<tr>
<td>0b0001</td>
<td>0b0010</td>
</tr>
<tr>
<td>0b0010</td>
<td>0b0100</td>
</tr>
<tr>
<td>0b0011</td>
<td>0b0110</td>
</tr>
<tr>
<td>0b0100</td>
<td>0b1000</td>
</tr>
<tr>
<td>0b0101</td>
<td>N/A</td>
</tr>
<tr>
<td>0b0111</td>
<td>N/A</td>
</tr>
<tr>
<td>0b1000</td>
<td>N/A</td>
</tr>
<tr>
<td>0b1001</td>
<td>N/A</td>
</tr>
<tr>
<td>0b1010</td>
<td>N/A</td>
</tr>
</tbody>
</table>

### Multiplier / Operand A Memory Assignment

These bits select the logical cryptographic RAM segment for parameter a.

<table>
<thead>
<tr>
<th>SETTINGS</th>
<th>SEGMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAWS &lt; 1024</td>
<td></td>
</tr>
<tr>
<td>0b0000</td>
<td>0b0000</td>
</tr>
<tr>
<td>0b0001</td>
<td>0b0010</td>
</tr>
<tr>
<td>0b0010</td>
<td>0b0100</td>
</tr>
<tr>
<td>0b0011</td>
<td>0b0110</td>
</tr>
<tr>
<td>0b0100</td>
<td>0b1000</td>
</tr>
<tr>
<td>0b0101</td>
<td>N/A</td>
</tr>
<tr>
<td>0b0111</td>
<td>N/A</td>
</tr>
<tr>
<td>0b1000</td>
<td>N/A</td>
</tr>
<tr>
<td>0b1001</td>
<td>N/A</td>
</tr>
<tr>
<td>0b1010</td>
<td>N/A</td>
</tr>
</tbody>
</table>
### MAA Control Register

<table>
<thead>
<tr>
<th>BITS</th>
<th>NAME</th>
<th>ACCESS</th>
<th>RESET</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:14</td>
<td>mms</td>
<td>R/W</td>
<td>0</td>
<td><strong>Modulus Memory Select</strong>&lt;br&gt;These bits select the starting position of parameter (m) within logical segment 5.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>SETTING</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>OFFSET WITHIN LOGICAL SEGMENT</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b11</td>
</tr>
<tr>
<td>13:12</td>
<td>ems</td>
<td>R/W</td>
<td>0</td>
<td><strong>Exponent Memory Select</strong>&lt;br&gt;These bits select the starting position of parameter (e) within the logical segment specified by (ema).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>SETTING</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>OFFSET WITHIN LOGICAL SEGMENT</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b11</td>
</tr>
<tr>
<td>11:10</td>
<td>bms</td>
<td>R/W</td>
<td>0</td>
<td><strong>Multiplicand B Memory Select</strong>&lt;br&gt;These bits select the starting position of parameter (b) within the logical segment specified by (bma).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>SETTING</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>OFFSET WITHIN LOGICAL SEGMENT</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b11</td>
</tr>
<tr>
<td>9:8</td>
<td>ams</td>
<td>R/W</td>
<td>0</td>
<td><strong>Multiplier A Memory Select</strong>&lt;br&gt;These bits select the starting position of parameter (a) within the logical segment specified by (ama).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>SETTING</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>OFFSET WITHIN LOGICAL SEGMENT</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b01</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0b11</td>
</tr>
<tr>
<td>7</td>
<td>maaer</td>
<td>R/WOC</td>
<td>0</td>
<td><strong>MAA Error</strong>&lt;br&gt;This bit is set by hardware if software writes to the MAA_CTRL or MAA_MAWS when MAA is in progress. This also clears STC and terminates the MAA operation. Once set, it must be cleared by software otherwise no new operation is initiated.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>0</strong>: No error&lt;br&gt;<strong>1</strong>: Error occurs</td>
</tr>
<tr>
<td>6:5</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td><strong>Reserved for Future Use</strong>&lt;br&gt;Do not modify this field from its default value.</td>
</tr>
<tr>
<td>4</td>
<td>ocalc</td>
<td>R/W</td>
<td>0</td>
<td><strong>Optimized Calculation Control</strong>&lt;br&gt;Setting this bit skips unnecessary multiply operations after normalizing the exponent are skipped.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>0</strong>: No optimization&lt;br&gt;<strong>1</strong>: Optimize calculation</td>
</tr>
</tbody>
</table>
### MAA Control Register

<table>
<thead>
<tr>
<th>BITS</th>
<th>NAME</th>
<th>ACCESS</th>
<th>RESET</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>3:1</td>
<td>clc</td>
<td>R/W</td>
<td>0</td>
<td><strong>Calculation Configuration</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>These bits select the MAA calculation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>0b000</strong>: Modular exponentiation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>0b001</strong>: Square operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>0b010</strong>: Multiplication</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>0b011</strong>: Square followed by a multiplication</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>0b100</strong>: Addition</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>0b101</strong>: Subtraction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>0b110</strong>: Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>0b111</strong>: Reserved</td>
</tr>
</tbody>
</table>

| 0    | stc   | R/W    | 0     | **Start Calculation**                            |
|      |       |        |       | This bit functions as both the control and the status of the MAA. Setting this bit initiates a calculation. It remains set while the calculation is in progress, and is cleared by hardware when the calculation is finished. The bit is cleared by hardware if the MAA_MAWS.msgsz value is invalid or the MAAER bit is set. Clearing this bit in software resets the controller to its default state. |
|      |       |        |       | **0**: No operation                               |
|      |       |        |       | **1**: Start calculation specified by CLC         |

---

**Table 23-16: Cryptographic Data Input Register**

| Cryptographic Data In Register 0 [31:0] | CRYPTO_DIN_0 | [0x0020] |
| Cryptographic Data In Register 1 [63:32] | CRYPTO_DIN_1 | [0x0024] |
| Cryptographic Data In Register 2 [95:64] | CRYPTO_DIN_2 | [0x0028] |
| Cryptographic Data In Register 3 [127:96] | CRYPTO_DIN_3 | [0x002C] |

<table>
<thead>
<tr>
<th>BITS</th>
<th>NAME</th>
<th>ACCESS</th>
<th>RESET</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>DATA</td>
<td>W</td>
<td>0</td>
<td><strong>Cryptographic Data Input</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>These registers form the read FIFO for the CMDA. The endian swap input control bit (CRYPTO_CTRL.BSI) affects this register.</td>
</tr>
</tbody>
</table>

---

**Table 23-17: Cryptographic Data Output Register**

| Cryptographic Data Out Register 0 [31:0] | CRYPTO_DOUT_0 | [0x0030] |
| Cryptographic Data Out Register 1 [63:32] | CRYPTO_DOUT_1 | [0x0034] |
| Cryptographic Data Out Register 2 [95:64] | CRYPTO_DOUT_2 | [0x0038] |
| Cryptographic Data Out Register 3 [127:96] | CRYPTO_DOUT_3 | [0x003C] |

<table>
<thead>
<tr>
<th>BITS</th>
<th>NAME</th>
<th>ACCESS</th>
<th>RESET</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>data</td>
<td>W</td>
<td>0</td>
<td><strong>Cryptographic Data Output</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>These registers form the write FIFO for the CMDA. Data is placed in the lower words of these four registers depending on the algorithm. For block cipher modes, this register holds the result of most recent encryption or decryption operation. These registers are affected by the endian swap bit (CRYPTO_CTRL.BSO).</td>
</tr>
</tbody>
</table>
Table 23-18: CRC Polynomial Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>data</td>
<td>W</td>
<td>0xEDB88320</td>
<td>CRC Polynomial</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This register holds the polynomial used for CRC calculations. The reset value of this register is the CRC-32 Ethernet polynomial. This register is affected by the MSB control bit.</td>
</tr>
</tbody>
</table>

Table 23-19: CRC Value Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>val</td>
<td>R/W</td>
<td>0x0000</td>
<td>CRC Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This register holds the result of a CRC calculation. The register will immediately be set to 0x0001 if the invalid value of 0x0000 is detected. This register is affected by the MSB control bit.</td>
</tr>
</tbody>
</table>

Table 23-20: CRC PRNG Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>prng</td>
<td>R/W</td>
<td>0</td>
<td>Pseudo Random Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Output of the Galois Field shift register. This holds the resulting pseudo-random number if entropy is disabled or the true random number if entropy is enabled.</td>
</tr>
</tbody>
</table>

Table 23-21: Hamming Error Correction Code Register

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Access</th>
<th>Reset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:17</td>
<td>-</td>
<td>RO</td>
<td>-</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field from its default value.</td>
</tr>
<tr>
<td>16</td>
<td>par</td>
<td>R/W</td>
<td>0</td>
<td>Parity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This is the parity of the entire array.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0: Even parity</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1: Odd parity</td>
</tr>
<tr>
<td>15:0</td>
<td>ecc</td>
<td>R/W</td>
<td>0</td>
<td>Hamming ECC Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>These bits are the even parity of their corresponding bit groups</td>
</tr>
</tbody>
</table>

Table 23-22: Cipher Initial Vector Register [3:0]

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[31:0]</td>
<td>Block Cipher Initial Vector</td>
</tr>
<tr>
<td></td>
<td>These registers hold the initial value for cipher operations that use CBC, CFB, OFB, or CNTR modes. This register is updated with each encryption or decryption operation. This register is affected by the endian swap bits.</td>
</tr>
</tbody>
</table>
### Table 23-23: Cipher Key Register [7:0]

<table>
<thead>
<tr>
<th>Cipher Key Register</th>
<th>NAME</th>
<th>ACCESS</th>
<th>RESET</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 [31:0]</td>
<td>CIPHER_KEY_0</td>
<td>R/W</td>
<td>0</td>
<td>Cipher Key</td>
</tr>
<tr>
<td>1 [63:32]</td>
<td>CIPHER_KEY_1</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2 [95:64]</td>
<td>CIPHER_KEY_2</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3 [127:96]</td>
<td>CIPHER_KEY_3</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4 [159:128]</td>
<td>CIPHER_KEY_4</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5 [191:160]</td>
<td>CIPHER_KEY_5</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6 [223:192]</td>
<td>CIPHER_KEY_6</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7 [255:224]</td>
<td>CIPHER_KEY_7</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

### Table 23-24: HASH Message Digest Register [15:0]

<table>
<thead>
<tr>
<th>Hash Message Digest Register</th>
<th>NAME</th>
<th>ACCESS</th>
<th>RESET</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 [31:0]</td>
<td>HASH_DIGEST_0</td>
<td>R/W</td>
<td>0</td>
<td>Calculated Hash Value</td>
</tr>
<tr>
<td>1 [63:32]</td>
<td>HASH_DIGEST_1</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2 [95:64]</td>
<td>HASH_DIGEST_2</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3 [127:96]</td>
<td>HASH_DIGEST_3</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4 [159:128]</td>
<td>HASH_DIGEST_4</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5 [191:160]</td>
<td>HASH_DIGEST_5</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6 [223:192]</td>
<td>HASH_DIGEST_6</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7 [255:224]</td>
<td>HASH_DIGEST_7</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8 [287:256]</td>
<td>HASH_DIGEST_8</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>9 [319:288]</td>
<td>HASH_DIGEST_9</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10 [351:320]</td>
<td>HASH_DIGEST_10</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>11 [383:352]</td>
<td>HASH_DIGEST_11</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>12 [415:384]</td>
<td>HASH_DIGEST_12</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>13 [447:416]</td>
<td>HASH_DIGEST_13</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>14 [479:448]</td>
<td>HASH_DIGEST_14</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>15 [511:480]</td>
<td>HASH_DIGEST_15</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

### Table 23-25: Hash Message Size Registers

<table>
<thead>
<tr>
<th>Hash Message Size Register</th>
<th>NAME</th>
<th>ACCESS</th>
<th>RESET</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 [31:0]</td>
<td>HASH_MSG_SZ_0</td>
<td>R/W</td>
<td>0</td>
<td>Hash Message Size</td>
</tr>
<tr>
<td>1 [63:32]</td>
<td>HASH_MSG_SZ_1</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2 [95:64]</td>
<td>HASH_MSG_SZ_2</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3 [127:96]</td>
<td>HASH_MSG_SZ_3</td>
<td>R/W</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

These registers hold the cipher key used for block cipher operations. The number of bits used depends on the specific operation. See the CIPHER_CTRL.key field for information on loading this register. This register is affected by the endian swap input control bits.

These 16 registers hold the 512-bit calculated hash value. This register is affected by the endian swap bits.

These four registers hold the 128-bit message size in bytes.
Table 23-26: MAA Word Size Register

<table>
<thead>
<tr>
<th>MAA Word Size Register</th>
<th>MAA_MAWS</th>
<th>[0x00D0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>BITS</td>
<td>NAME</td>
<td>ACCESS</td>
</tr>
</tbody>
</table>
| 31:12  | -     | RO     | 0     | Reserved for Future Use  
Do not modify this field from its default value. |
| 11:0   | msgsz | R/W    | 0     | **MAA Word Size**  
This register defines the number of bits for a modular operation. Valid values are from 1 to 2048. Invalid values are ignored and do not initiate a MAA operation. You can only write to this register when MAA_CTRL.stc = 0 (MAA idle). |
### Table 23-27: TRNG Control Register (Base address 0x400B_5000)

<table>
<thead>
<tr>
<th>BITs</th>
<th>NAME</th>
<th>ACCESS</th>
<th>RESET</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:7</td>
<td>-</td>
<td>RO</td>
<td>1</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field from its default value.</td>
</tr>
<tr>
<td>6</td>
<td>aeskg</td>
<td>R/W</td>
<td>0</td>
<td>AES Key Generate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>When enabled, the key for securing NVSRAM is generated and transferred to</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>the secure key register automatically without user visibility or intervention.</td>
</tr>
<tr>
<td>5</td>
<td>rng_is</td>
<td>RO</td>
<td>0</td>
<td>Random Number Ready Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit is set when a new 32 bit random number is available in ( TRNG_DATA ). This bit is cleared by hardware if all the random words have been read. It is needed to poll this bit before reading the TRNG Data Register.</td>
</tr>
<tr>
<td>4</td>
<td>rng_i4s</td>
<td>RO</td>
<td>0</td>
<td>128-bit Random Number Ready Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit is set when a new 128 bit random number is ready to be read (using 4 consecutive reads of ( TRNG_DATA ). When set, an interrupt will be generated if ( TRNG_CN.rng_ie = 1 ). This bit is cleared by setting ( TRNG_CN.rng_isc ).</td>
</tr>
<tr>
<td>3</td>
<td>rng_isc</td>
<td>W</td>
<td>0</td>
<td>Random Number Interrupt Status Clear</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Setting this bit to 1 clears ( TRNG_CN.rng_i4s ) and acknowledges the interrupt, if enabled. This it is a write only bit and always reads as zero.</td>
</tr>
<tr>
<td>2</td>
<td>rng_ie</td>
<td>R/W</td>
<td>0</td>
<td>Random Number Interrupt Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This bit enables an interrupt to be generated when ( TRNG_CN.rng_i4s = 1 ).</td>
</tr>
<tr>
<td>1:0</td>
<td>-</td>
<td>RO</td>
<td>0</td>
<td>Reserved for Future Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Do not modify this field from its default value.</td>
</tr>
</tbody>
</table>

### Table 23-28: TRNG Data Register (Base address 0x400B_5000)

<table>
<thead>
<tr>
<th>BITs</th>
<th>NAME</th>
<th>ACCESS</th>
<th>RESET</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>data</td>
<td>RO</td>
<td>0</td>
<td>TRNG Data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The function of this register is dependent on the rng_is and rng_i4s bits</td>
</tr>
</tbody>
</table>
24. Revision History

<table>
<thead>
<tr>
<th>REVISION NUMBER</th>
<th>REVISION DATE</th>
<th>DESCRIPTION</th>
<th>PAGES CHANGED</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6/19</td>
<td>Initial Release</td>
<td>-</td>
</tr>
</tbody>
</table>

©2019 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.