



**LUMINARY** MICRO<sup>®</sup>

---

# LM3S1607 Microcontroller

**DATA SHEET**

---

## Legal Disclaimers and Trademark Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH LUMINARY MICRO PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN LUMINARY MICRO'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, LUMINARY MICRO ASSUMES NO LIABILITY WHATSOEVER, AND LUMINARY MICRO DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF LUMINARY MICRO'S PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. LUMINARY MICRO'S PRODUCTS ARE NOT INTENDED FOR USE IN MEDICAL, LIFE SAVING, OR LIFE-SUSTAINING APPLICATIONS.

Luminary Micro may make changes to specifications and product descriptions at any time, without notice. Contact your local Luminary Micro sales office or your distributor to obtain the latest specifications before placing your product order.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Luminary Micro reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

Copyright © 2007-2008 Luminary Micro, Inc. All rights reserved. Stellaris, Luminary Micro, and the Luminary Micro logo are registered trademarks of Luminary Micro, Inc. or its subsidiaries in the United States and other countries. ARM and Thumb are registered trademarks and Cortex is a trademark of ARM Limited. Other names and brands may be claimed as the property of others.

Luminary Micro, Inc.  
108 Wild Basin, Suite 350  
Austin, TX 78746  
Main: +1-512-279-8800  
Fax: +1-512-279-8879  
<http://www.luminarymicro.com>



**LUMINARY** MICRO



# Table of Contents

<b>About This Document .....</b>	<b>19</b>
Audience .....	19
About This Manual .....	19
Related Documents .....	19
Documentation Conventions .....	19
<b>1 Architectural Overview .....</b>	<b>22</b>
1.1 Product Features .....	22
1.2 Target Applications .....	27
1.3 High-Level Block Diagram .....	28
1.4 Functional Overview .....	29
1.4.1 ARM Cortex™-M3 .....	30
1.4.2 Motor Control Peripherals .....	31
1.4.3 Analog Peripherals .....	31
1.4.4 Serial Communications Peripherals .....	31
1.4.5 System Peripherals .....	32
1.4.6 Memory Peripherals .....	33
1.4.7 Additional Features .....	34
1.4.8 Hardware Details .....	35
<b>2 ARM Cortex-M3 Processor Core .....</b>	<b>36</b>
2.1 Block Diagram .....	37
2.2 Functional Description .....	37
2.2.1 Serial Wire and JTAG Debug .....	38
2.2.2 Embedded Trace Macrocell (ETM) .....	38
2.2.3 Trace Port Interface Unit (TPIU) .....	38
2.2.4 ROM Table .....	38
2.2.5 Memory Protection Unit (MPU) .....	38
2.2.6 Nested Vectored Interrupt Controller (NVIC) .....	39
<b>3 Memory Map .....</b>	<b>42</b>
<b>4 Interrupts .....</b>	<b>45</b>
<b>5 JTAG Interface .....</b>	<b>48</b>
5.1 Block Diagram .....	49
5.2 Functional Description .....	49
5.2.1 JTAG Interface Pins .....	50
5.2.2 JTAG TAP Controller .....	51
5.2.3 Shift Registers .....	52
5.2.4 Operational Considerations .....	52
5.3 Initialization and Configuration .....	55
5.4 Register Descriptions .....	55
5.4.1 Instruction Register (IR) .....	55
5.4.2 Data Registers .....	57
<b>6 System Control .....</b>	<b>60</b>
6.1 Functional Description .....	60
6.1.1 Device Identification .....	60
6.1.2 Reset Control .....	60

6.1.3	Non-Maskable Interrupt .....	63
6.1.4	Power Control .....	63
6.1.5	Clock Control .....	63
6.1.6	System Control .....	67
6.2	Initialization and Configuration .....	68
6.3	Register Map .....	68
6.4	Register Descriptions .....	70
<b>7</b>	<b>Hibernation Module .....</b>	<b>124</b>
7.1	Block Diagram .....	125
7.2	Functional Description .....	125
7.2.1	Register Access Timing .....	125
7.2.2	Clock Source .....	126
7.2.3	Battery Management .....	128
7.2.4	Real-Time Clock .....	129
7.2.5	Non-Volatile Memory .....	129
7.2.6	Power Control .....	129
7.2.7	Interrupts and Status .....	130
7.3	Initialization and Configuration .....	130
7.3.1	Initialization .....	130
7.3.2	RTC Match Functionality (No Hibernation) .....	131
7.3.3	RTC Match/Wake-Up from Hibernation .....	131
7.3.4	External Wake-Up from Hibernation .....	131
7.3.5	RTC/External Wake-Up from Hibernation .....	131
7.3.6	Register Reset .....	131
7.4	Register Map .....	132
7.5	Register Descriptions .....	133
<b>8</b>	<b>Internal Memory .....</b>	<b>147</b>
8.1	Block Diagram .....	147
8.2	Functional Description .....	147
8.2.1	SRAM Memory .....	147
8.2.2	ROM Memory .....	148
8.2.3	Flash Memory .....	148
8.3	Flash Memory Initialization and Configuration .....	149
8.3.1	Flash Programming .....	149
8.3.2	Nonvolatile Register Programming .....	150
8.4	Register Map .....	151
8.5	ROM Register Descriptions (System Control Offset) .....	152
8.6	Flash Register Descriptions (Flash Control Offset) .....	153
8.7	Flash Register Descriptions (System Control Offset) .....	160
<b>9</b>	<b>Micro Direct Memory Access (μDMA) .....</b>	<b>176</b>
9.1	Block Diagram .....	177
9.2	Functional Description .....	177
9.2.1	Channel Assignments .....	178
9.2.2	Priority .....	178
9.2.3	Arbitration Size .....	178
9.2.4	Request Types .....	178
9.2.5	Channel Configuration .....	179
9.2.6	Transfer Modes .....	181

9.2.7	Transfer Size and Increment .....	189
9.2.8	Peripheral Interface .....	189
9.2.9	Software Request .....	189
9.2.10	Interrupts and Errors .....	190
9.3	Initialization and Configuration .....	190
9.3.1	Module Initialization .....	190
9.3.2	Configuring a Memory-to-Memory Transfer .....	190
9.3.3	Configuring a Peripheral for Simple Transmit .....	192
9.3.4	Configuring a Peripheral for Ping-Pong Receive .....	193
9.4	Register Map .....	196
9.5	μDMA Channel Control Structure .....	197
9.6	μDMA Register Descriptions .....	203
<b>10</b>	<b>General-Purpose Input/Outputs (GPIOs) .....</b>	<b>237</b>
10.1	Functional Description .....	237
10.1.1	Data Control .....	239
10.1.2	Interrupt Control .....	240
10.1.3	Mode Control .....	241
10.1.4	Commit Control .....	241
10.1.5	Pad Control .....	241
10.1.6	Identification .....	242
10.2	Initialization and Configuration .....	242
10.3	Register Map .....	243
10.4	Register Descriptions .....	245
<b>11</b>	<b>General-Purpose Timers .....</b>	<b>282</b>
11.1	Block Diagram .....	282
11.2	Functional Description .....	283
11.2.1	GPTM Reset Conditions .....	284
11.2.2	32-Bit Timer Operating Modes .....	284
11.2.3	16-Bit Timer Operating Modes .....	285
11.3	Initialization and Configuration .....	289
11.3.1	32-Bit One-Shot/Periodic Timer Mode .....	289
11.3.2	32-Bit Real-Time Clock (RTC) Mode .....	290
11.3.3	16-Bit One-Shot/Periodic Timer Mode .....	290
11.3.4	16-Bit Input Edge Count Mode .....	291
11.3.5	16-Bit Input Edge Timing Mode .....	291
11.3.6	16-Bit PWM Mode .....	292
11.4	Register Map .....	292
11.5	Register Descriptions .....	293
<b>12</b>	<b>Watchdog Timer .....</b>	<b>316</b>
12.1	Block Diagram .....	316
12.2	Functional Description .....	316
12.3	Initialization and Configuration .....	317
12.4	Register Map .....	317
12.5	Register Descriptions .....	318
<b>13</b>	<b>Analog-to-Digital Converter (ADC) .....</b>	<b>339</b>
13.1	Block Diagram .....	340
13.2	Functional Description .....	340

13.2.1	Sample Sequencers .....	340
13.2.2	Module Control .....	341
13.2.3	Hardware Sample Averaging Circuit .....	342
13.2.4	Analog-to-Digital Converter .....	342
13.2.5	Differential Sampling .....	342
13.2.6	Internal Temperature Sensor .....	344
13.3	Initialization and Configuration .....	344
13.3.1	Module Initialization .....	345
13.3.2	Sample Sequencer Configuration .....	345
13.4	Register Map .....	345
13.5	Register Descriptions .....	346
<b>14</b>	<b>Universal Asynchronous Receivers/Transmitters (UARTs) .....</b>	<b>371</b>
14.1	Block Diagram .....	372
14.2	Functional Description .....	372
14.2.1	Transmit/Receive Logic .....	372
14.2.2	Baud-Rate Generation .....	373
14.2.3	Data Transmission .....	373
14.2.4	Serial IR (SIR) .....	374
14.2.5	FIFO Operation .....	375
14.2.6	Interrupts .....	375
14.2.7	Loopback Operation .....	376
14.2.8	DMA Operation .....	376
14.2.9	IrDA SIR block .....	377
14.3	Initialization and Configuration .....	377
14.4	Register Map .....	378
14.5	Register Descriptions .....	379
<b>15</b>	<b>Synchronous Serial Interface (SSI) .....</b>	<b>414</b>
15.1	Block Diagram .....	414
15.2	Functional Description .....	415
15.2.1	Bit Rate Generation .....	415
15.2.2	FIFO Operation .....	415
15.2.3	Interrupts .....	415
15.2.4	Frame Formats .....	416
15.2.5	DMA Operation .....	423
15.3	Initialization and Configuration .....	424
15.4	Register Map .....	425
15.5	Register Descriptions .....	426
<b>16</b>	<b>Inter-Integrated Circuit (I<sup>2</sup>C) Interface .....</b>	<b>453</b>
16.1	Block Diagram .....	453
16.2	Functional Description .....	453
16.2.1	I <sup>2</sup> C Bus Functional Overview .....	454
16.2.2	Available Speed Modes .....	456
16.2.3	Interrupts .....	457
16.2.4	Loopback Operation .....	458
16.2.5	Command Sequence Flow Charts .....	458
16.3	Initialization and Configuration .....	464
16.4	I <sup>2</sup> C Register Map .....	465

16.5	Register Descriptions (I <sup>2</sup> C Master) .....	466
16.6	Register Descriptions (I2C Slave) .....	479
<b>17</b>	<b>Pin Diagram .....</b>	<b>488</b>
<b>18</b>	<b>Signal Tables .....</b>	<b>489</b>
<b>19</b>	<b>Operating Characteristics .....</b>	<b>499</b>
<b>20</b>	<b>Electrical Characteristics .....</b>	<b>500</b>
20.1	DC Characteristics .....	500
20.1.1	Maximum Ratings .....	500
20.1.2	Recommended DC Operating Conditions .....	500
20.1.3	On-Chip Low Drop-Out (LDO) Regulator Characteristics .....	501
20.1.4	Power Specifications .....	501
20.1.5	Flash Memory Characteristics .....	503
20.1.6	Hibernation .....	503
20.2	AC Characteristics .....	503
20.2.1	Load Conditions .....	503
20.2.2	Clocks .....	503
20.2.3	Analog-to-Digital Converter .....	504
20.2.4	I <sup>2</sup> C .....	505
20.2.5	Hibernation Module .....	505
20.2.6	Synchronous Serial Interface (SSI) .....	506
20.2.7	JTAG and Boundary Scan .....	508
20.2.8	General-Purpose I/O .....	509
20.2.9	Reset .....	510
<b>21</b>	<b>Package Information .....</b>	<b>512</b>
<b>A</b>	<b>Boot Loader .....</b>	<b>514</b>
A.1	Boot Loader .....	514
A.2	Interfaces .....	514
A.2.1	UART .....	514
A.2.2	SSI .....	514
A.2.3	I <sup>2</sup> C .....	515
A.3	Packet Handling .....	515
A.3.1	Packet Format .....	515
A.3.2	Sending Packets .....	515
A.3.3	Receiving Packets .....	516
A.4	Commands .....	516
A.4.1	COMMAND_PING (0X20) .....	516
A.4.2	COMMAND_GET_STATUS (0x23) .....	516
A.4.3	COMMAND_DOWNLOAD (0x21) .....	516
A.4.4	COMMAND_SEND_DATA (0x24) .....	517
A.4.5	COMMAND_RUN (0x22) .....	517
A.4.6	COMMAND_RESET (0x25) .....	518
<b>B</b>	<b>ROM DriverLib Functions .....</b>	<b>519</b>
B.1	DriverLib Functions Included in the Integrated ROM .....	519
<b>C</b>	<b>Register Quick Reference .....</b>	<b>530</b>
<b>D</b>	<b>Ordering and Contact Information .....</b>	<b>547</b>
D.1	Ordering Information .....	547

D.2 Kits ..... 547  
D.3 Company Information ..... 547  
D.4 Support Information ..... 548



## List of Figures

Figure 1-1.	Stellaris® 1000 Series High-Level Block Diagram .....	29
Figure 2-1.	CPU Block Diagram .....	37
Figure 2-2.	TPIU Block Diagram .....	38
Figure 5-1.	JTAG Module Block Diagram .....	49
Figure 5-2.	Test Access Port State Machine .....	52
Figure 5-3.	IDCODE Register Format .....	57
Figure 5-4.	BYPASS Register Format .....	58
Figure 5-5.	Boundary Scan Register Format .....	58
Figure 6-1.	External Circuitry to Extend Reset .....	61
Figure 6-2.	Main Clock Tree .....	65
Figure 7-1.	Hibernation Module Block Diagram .....	125
Figure 7-2.	Clock Source Using Crystal .....	127
Figure 7-3.	Clock Source Using Dedicated Oscillator .....	128
Figure 8-1.	Flash Block Diagram .....	147
Figure 9-1.	μDMA Block Diagram .....	177
Figure 9-2.	Example of Ping-Pong DMA Transaction .....	182
Figure 9-3.	Memory Scatter-Gather, Setup and Configuration .....	184
Figure 9-4.	Memory Scatter-Gather, μDMA Copy Sequence .....	185
Figure 9-5.	Peripheral Scatter-Gather, Setup and Configuration .....	187
Figure 9-6.	Peripheral Scatter-Gather, μDMA Copy Sequence .....	188
Figure 10-1.	Digital I/O Pads .....	238
Figure 10-2.	Analog/Digital I/O Pads .....	239
Figure 10-3.	GPIO DATA Write Example .....	240
Figure 10-4.	GPIO DATA Read Example .....	240
Figure 11-1.	GPTM Module Block Diagram .....	283
Figure 11-2.	16-Bit Input Edge Count Mode Example .....	287
Figure 11-3.	16-Bit Input Edge Time Mode Example .....	288
Figure 11-4.	16-Bit PWM Mode Example .....	289
Figure 12-1.	WDT Module Block Diagram .....	316
Figure 13-1.	ADC Module Block Diagram .....	340
Figure 13-2.	Differential Sampling Range, $V_{IN\_ODD} = 1.5\text{ V}$ .....	343
Figure 13-3.	Differential Sampling Range, $V_{IN\_ODD} = 0.75\text{ V}$ .....	343
Figure 13-4.	Differential Sampling Range, $V_{IN\_ODD} = 2.25\text{ V}$ .....	344
Figure 13-5.	Internal Temperature Sensor Characteristic .....	344
Figure 14-1.	UART Module Block Diagram .....	372
Figure 14-2.	UART Character Frame .....	373
Figure 14-3.	IrDA Data Modulation .....	375
Figure 15-1.	SSI Module Block Diagram .....	414
Figure 15-2.	TI Synchronous Serial Frame Format (Single Transfer) .....	417
Figure 15-3.	TI Synchronous Serial Frame Format (Continuous Transfer) .....	417
Figure 15-4.	Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0 .....	418
Figure 15-5.	Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0 .....	418
Figure 15-6.	Freescale SPI Frame Format with SPO=0 and SPH=1 .....	419
Figure 15-7.	Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0 .....	420
Figure 15-8.	Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0 .....	420

Figure 15-9. Freescale SPI Frame Format with SPO=1 and SPH=1 .....	421
Figure 15-10. MICROWIRE Frame Format (Single Frame) .....	422
Figure 15-11. MICROWIRE Frame Format (Continuous Transfer) .....	423
Figure 15-12. MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements .....	423
Figure 16-1. I <sup>2</sup> C Block Diagram .....	453
Figure 16-2. I <sup>2</sup> C Bus Configuration .....	454
Figure 16-3. START and STOP Conditions .....	454
Figure 16-4. Complete Data Transfer with a 7-Bit Address .....	455
Figure 16-5. R/S Bit in First Byte .....	455
Figure 16-6. Data Validity During Bit Transfer on the I <sup>2</sup> C Bus .....	455
Figure 16-7. Master Single SEND .....	458
Figure 16-8. Master Single RECEIVE .....	459
Figure 16-9. Master Burst SEND .....	460
Figure 16-10. Master Burst RECEIVE .....	461
Figure 16-11. Master Burst RECEIVE after Burst SEND .....	462
Figure 16-12. Master Burst SEND after Burst RECEIVE .....	463
Figure 16-13. Slave Command Sequence .....	464
Figure 17-1. 64-Pin LQFP Package Pin Diagram .....	488
Figure 20-1. Load Conditions .....	503
Figure 20-2. I <sup>2</sup> C Timing .....	505
Figure 20-3. Hibernation Module Timing .....	506
Figure 20-4. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement .....	507
Figure 20-5. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer .....	507
Figure 20-6. SSI Timing for SPI Frame Format (FRF=00), with SPH=1 .....	508
Figure 20-7. JTAG Test Clock Input Timing .....	509
Figure 20-8. JTAG Test Access Port (TAP) Timing .....	509
Figure 20-9. External Reset Timing ( $\overline{\text{RST}}$ ) .....	510
Figure 20-10. Power-On Reset Timing .....	510
Figure 20-11. Brown-Out Reset Timing .....	511
Figure 20-12. Software Reset Timing .....	511
Figure 20-13. Watchdog Reset Timing .....	511
Figure 21-1. 64-Pin LQFP Package .....	512

## List of Tables

Table 1.	Documentation Conventions .....	19
Table 3-1.	Memory Map .....	42
Table 4-1.	Exception Types .....	45
Table 4-2.	Interrupts .....	46
Table 5-1.	JTAG Port Pins Reset State .....	50
Table 5-2.	JTAG Instruction Register Commands .....	55
Table 6-1.	System Control Register Map .....	69
Table 7-1.	Hibernation Module Register Map .....	132
Table 8-1.	Flash Protection Policy Combinations .....	149
Table 8-2.	Flash Resident Registers .....	150
Table 8-3.	Flash Register Map .....	151
Table 9-1.	DMA Channel Assignments .....	178
Table 9-2.	Request Type Support .....	179
Table 9-3.	Control Structure Memory Map .....	180
Table 9-4.	Channel Control Structure .....	180
Table 9-5.	$\mu$ DMA Read Example: 8-Bit Peripheral .....	189
Table 9-6.	$\mu$ DMA Interrupt Assignments .....	190
Table 9-7.	Channel Control Structure Offsets for Channel 30 .....	191
Table 9-8.	Channel Control Word Configuration for Memory Transfer Example .....	191
Table 9-9.	Channel Control Structure Offsets for Channel 7 .....	192
Table 9-10.	Channel Control Word Configuration for Peripheral Transmit Example .....	193
Table 9-11.	Primary and Alternate Channel Control Structure Offsets for Channel 8 .....	194
Table 9-12.	Channel Control Word Configuration for Peripheral Ping-Pong Receive Example .....	195
Table 9-13.	$\mu$ DMA Register Map .....	196
Table 10-1.	GPIO Pad Configuration Examples .....	242
Table 10-2.	GPIO Interrupt Configuration Example .....	243
Table 10-3.	GPIO Register Map .....	244
Table 11-1.	Available CCP Pins .....	283
Table 11-2.	16-Bit Timer With Prescaler Configurations .....	286
Table 11-3.	Timers Register Map .....	292
Table 12-1.	Watchdog Timer Register Map .....	317
Table 13-1.	Samples and FIFO Depth of Sequencers .....	340
Table 13-2.	Differential Sampling Pairs .....	342
Table 13-3.	ADC Register Map .....	345
Table 14-1.	UART Register Map .....	378
Table 15-1.	SSI Register Map .....	425
Table 16-1.	Examples of I <sup>2</sup> C Master Timer Period versus Speed Mode .....	456
Table 16-2.	Inter-Integrated Circuit (I <sup>2</sup> C) Interface Register Map .....	465
Table 16-3.	Write Field Decoding for I2CMCS[3:0] Field (Sheet 1 of 3) .....	470
Table 18-1.	Signals by Pin Number .....	489
Table 18-2.	Signals by Signal Name .....	492
Table 18-3.	Signals by Function, Except for GPIO .....	495
Table 18-4.	GPIO Pins and Alternate Functions .....	497
Table 19-1.	Temperature Characteristics .....	499
Table 19-2.	Thermal Characteristics .....	499
Table 20-1.	Maximum Ratings .....	500

Table 20-2.	Recommended DC Operating Conditions .....	500
Table 20-3.	LDO Regulator Characteristics .....	501
Table 20-4.	Detailed Power Specifications .....	502
Table 20-5.	Flash Memory Characteristics .....	503
Table 20-6.	Hibernation Module DC Characteristics .....	503
Table 20-7.	Phase Locked Loop (PLL) Characteristics .....	503
Table 20-8.	Clock Characteristics .....	504
Table 20-9.	Crystal Characteristics .....	504
Table 20-10.	ADC Characteristics .....	504
Table 20-11.	I <sup>2</sup> C Characteristics .....	505
Table 20-12.	Hibernation Module AC Characteristics .....	506
Table 20-13.	SSI Characteristics .....	506
Table 20-14.	JTAG Characteristics .....	508
Table 20-15.	GPIO Characteristics .....	509
Table 20-16.	Reset Characteristics .....	510
Table D-1.	Part Ordering Information .....	547

## List of Registers

<b>System Control .....</b>	<b>60</b>
Register 1: Device Identification 0 (DID0), offset 0x000 .....	71
Register 2: Brown-Out Reset Control (PBORCTL), offset 0x030 .....	73
Register 3: LDO Power Control (LDOPCTL), offset 0x034 .....	74
Register 4: Raw Interrupt Status (RIS), offset 0x050 .....	75
Register 5: Interrupt Mask Control (IMC), offset 0x054 .....	76
Register 6: Masked Interrupt Status and Clear (MISC), offset 0x058 .....	77
Register 7: Reset Cause (RESC), offset 0x05C .....	78
Register 8: Run-Mode Clock Configuration (RCC), offset 0x060 .....	79
Register 9: XTAL to PLL Translation (PLLCFG), offset 0x064 .....	83
Register 10: GPIO High Speed Control (GPIOHSCTL), offset 0x06C .....	84
Register 11: Run-Mode Clock Configuration 2 (RCC2), offset 0x070 .....	85
Register 12: Main Oscillator Control (MOSCCTL), offset 0x07C .....	87
Register 13: Deep Sleep Clock Configuration (DSLPCCLKCFG), offset 0x144 .....	88
Register 14: Device Identification 1 (DID1), offset 0x004 .....	89
Register 15: Device Capabilities 0 (DC0), offset 0x008 .....	91
Register 16: Device Capabilities 1 (DC1), offset 0x010 .....	92
Register 17: Device Capabilities 2 (DC2), offset 0x014 .....	94
Register 18: Device Capabilities 3 (DC3), offset 0x018 .....	96
Register 19: Device Capabilities 4 (DC4), offset 0x01C .....	98
Register 20: Device Capabilities 5 (DC5), offset 0x020 .....	99
Register 21: Device Capabilities 6 (DC6), offset 0x024 .....	100
Register 22: Device Capabilities 7 (DC7), offset 0x028 .....	101
Register 23: Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100 .....	102
Register 24: Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110 .....	104
Register 25: Deep Sleep Mode Clock Gating Control Register 0 (DCGC0), offset 0x120 .....	106
Register 26: Run Mode Clock Gating Control Register 1 (RCGC1), offset 0x104 .....	108
Register 27: Sleep Mode Clock Gating Control Register 1 (SCGC1), offset 0x114 .....	110
Register 28: Deep Sleep Mode Clock Gating Control Register 1 (DCGC1), offset 0x124 .....	112
Register 29: Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108 .....	114
Register 30: Sleep Mode Clock Gating Control Register 2 (SCGC2), offset 0x118 .....	116
Register 31: Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128 .....	118
Register 32: Software Reset Control 0 (SRCR0), offset 0x040 .....	120
Register 33: Software Reset Control 1 (SRCR1), offset 0x044 .....	121
Register 34: Software Reset Control 2 (SRCR2), offset 0x048 .....	123
<b>Hibernation Module .....</b>	<b>124</b>
Register 1: Hibernation RTC Counter (HIBRTCC), offset 0x000 .....	134
Register 2: Hibernation RTC Match 0 (HIBRTCM0), offset 0x004 .....	135
Register 3: Hibernation RTC Match 1 (HIBRTCM1), offset 0x008 .....	136
Register 4: Hibernation RTC Load (HIBRTCLD), offset 0x00C .....	137
Register 5: Hibernation Control (HIBCTL), offset 0x010 .....	138
Register 6: Hibernation Interrupt Mask (HIBIM), offset 0x014 .....	141
Register 7: Hibernation Raw Interrupt Status (HIBRIS), offset 0x018 .....	142
Register 8: Hibernation Masked Interrupt Status (HIBMIS), offset 0x01C .....	143
Register 9: Hibernation Interrupt Clear (HIBIC), offset 0x020 .....	144

Register 10:	Hibernation RTC Trim (HIBRTCT), offset 0x024 .....	145
Register 11:	Hibernation Data (HIBDATA), offset 0x030-0x12C .....	146
<b>Internal Memory</b>	<b>.....</b>	<b>147</b>
Register 1:	ROM Control (RMCTL), offset 0x0F0 .....	153
Register 2:	Flash Memory Address (FMA), offset 0x000 .....	154
Register 3:	Flash Memory Data (FMD), offset 0x004 .....	155
Register 4:	Flash Memory Control (FMC), offset 0x008 .....	156
Register 5:	Flash Controller Raw Interrupt Status (FCRIS), offset 0x00C .....	158
Register 6:	Flash Controller Interrupt Mask (FCIM), offset 0x010 .....	159
Register 7:	Flash Controller Masked Interrupt Status and Clear (FCMISC), offset 0x014 .....	160
Register 8:	USec Reload (USECRL), offset 0x140 .....	161
Register 9:	ROM Version Register (RMVER), offset 0x0F4 .....	162
Register 10:	Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x130 and 0x200 .....	163
Register 11:	Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x134 and 0x400 .....	164
Register 12:	User Debug (USER_DBG), offset 0x1D0 .....	165
Register 13:	User Register 0 (USER_REG0), offset 0x1E0 .....	166
Register 14:	User Register 1 (USER_REG1), offset 0x1E4 .....	167
Register 15:	User Register 2 (USER_REG2), offset 0x1E8 .....	168
Register 16:	User Register 3 (USER_REG3), offset 0x1EC .....	169
Register 17:	Flash Memory Protection Read Enable 1 (FMPRE1), offset 0x204 .....	170
Register 18:	Flash Memory Protection Read Enable 2 (FMPRE2), offset 0x208 .....	171
Register 19:	Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C .....	172
Register 20:	Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404 .....	173
Register 21:	Flash Memory Protection Program Enable 2 (FMPPE2), offset 0x408 .....	174
Register 22:	Flash Memory Protection Program Enable 3 (FMPPE3), offset 0x40C .....	175
<b>Micro Direct Memory Access (μDMA)</b>	<b>.....</b>	<b>176</b>
Register 1:	DMA Channel Source Address End Pointer (DMASRCENDP), offset 0x000 .....	198
Register 2:	DMA Channel Destination Address End Pointer (DMADSTENDP), offset 0x004 .....	199
Register 3:	DMA Channel Control Word (DMACHCTL), offset 0x008 .....	200
Register 4:	DMA Status (DMASTAT), offset 0x000 .....	204
Register 5:	DMA Configuration (DMACFG), offset 0x004 .....	206
Register 6:	DMA Channel Control Base Pointer (DMACTLBASE), offset 0x008 .....	207
Register 7:	DMA Alternate Channel Control Base Pointer (DMAALTBASE), offset 0x00C .....	208
Register 8:	DMA Channel Wait on Request Status (DMAWAITSTAT), offset 0x010 .....	209
Register 9:	DMA Channel Software Request (DMASWREQ), offset 0x014 .....	210
Register 10:	DMA Channel Useburst Set (DMAUSEBURSTSET), offset 0x018 .....	211
Register 11:	DMA Channel Useburst Clear (DMAUSEBURSTCLR), offset 0x01C .....	213
Register 12:	DMA Channel Request Mask Set (DMAREQMASKSET), offset 0x020 .....	214
Register 13:	DMA Channel Request Mask Clear (DMAREQMASKCLR), offset 0x024 .....	216
Register 14:	DMA Channel Enable Set (DMAENASET), offset 0x028 .....	217
Register 15:	DMA Channel Enable Clear (DMAENACLAR), offset 0x02C .....	219
Register 16:	DMA Channel Primary Alternate Set (DMAALTSET), offset 0x030 .....	220
Register 17:	DMA Channel Primary Alternate Clear (DMAALTCLR), offset 0x034 .....	222
Register 18:	DMA Channel Priority Set (DMAPRIOSET), offset 0x038 .....	223
Register 19:	DMA Channel Priority Clear (DMAPRIOCLR), offset 0x03C .....	225
Register 20:	DMA Bus Error Clear (DMAERRCLR), offset 0x04C .....	226
Register 21:	DMA Peripheral Identification 0 (DMAPeriphID0), offset 0xFE0 .....	228
Register 22:	DMA Peripheral Identification 1 (DMAPeriphID1), offset 0xFE4 .....	229

Register 23:	DMA Peripheral Identification 2 (DMAPeriphID2), offset 0xFE8 .....	230
Register 24:	DMA Peripheral Identification 3 (DMAPeriphID3), offset 0xFEC .....	231
Register 25:	DMA Peripheral Identification 4 (DMAPeriphID4), offset 0xFD0 .....	232
Register 26:	DMA PrimeCell Identification 0 (DMAPCellID0), offset 0xFF0 .....	233
Register 27:	DMA PrimeCell Identification 1 (DMAPCellID1), offset 0xFF4 .....	234
Register 28:	DMA PrimeCell Identification 2 (DMAPCellID2), offset 0xFF8 .....	235
Register 29:	DMA PrimeCell Identification 3 (DMAPCellID3), offset 0xFFC .....	236
<b>General-Purpose Input/Outputs (GPIOs) .....</b>		<b>237</b>
Register 1:	GPIO Data (GPIODATA), offset 0x000 .....	246
Register 2:	GPIO Direction (GPIODIR), offset 0x400 .....	247
Register 3:	GPIO Interrupt Sense (GPIOIS), offset 0x404 .....	248
Register 4:	GPIO Interrupt Both Edges (GPIOIBE), offset 0x408 .....	249
Register 5:	GPIO Interrupt Event (GPIOIEV), offset 0x40C .....	250
Register 6:	GPIO Interrupt Mask (GPIOIM), offset 0x410 .....	251
Register 7:	GPIO Raw Interrupt Status (GPIORIS), offset 0x414 .....	252
Register 8:	GPIO Masked Interrupt Status (GPIOMIS), offset 0x418 .....	253
Register 9:	GPIO Interrupt Clear (GPIOICR), offset 0x41C .....	254
Register 10:	GPIO Alternate Function Select (GPIOAFSEL), offset 0x420 .....	255
Register 11:	GPIO 2-mA Drive Select (GPIODR2R), offset 0x500 .....	257
Register 12:	GPIO 4-mA Drive Select (GPIODR4R), offset 0x504 .....	258
Register 13:	GPIO 8-mA Drive Select (GPIODR8R), offset 0x508 .....	259
Register 14:	GPIO Open Drain Select (GPIOODR), offset 0x50C .....	260
Register 15:	GPIO Pull-Up Select (GPIOPUR), offset 0x510 .....	261
Register 16:	GPIO Pull-Down Select (GPIOPDR), offset 0x514 .....	262
Register 17:	GPIO Slew Rate Control Select (GPIOSLR), offset 0x518 .....	263
Register 18:	GPIO Digital Enable (GPIODEN), offset 0x51C .....	264
Register 19:	GPIO Lock (GPIOLOCK), offset 0x520 .....	266
Register 20:	GPIO Commit (GPIOCR), offset 0x524 .....	267
Register 21:	GPIO Analog Mode Select (GPIOAMSEL), offset 0x528 .....	269
Register 22:	GPIO Peripheral Identification 4 (GPIOPeriphID4), offset 0xFD0 .....	270
Register 23:	GPIO Peripheral Identification 5 (GPIOPeriphID5), offset 0xFD4 .....	271
Register 24:	GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8 .....	272
Register 25:	GPIO Peripheral Identification 7 (GPIOPeriphID7), offset 0xFDC .....	273
Register 26:	GPIO Peripheral Identification 0 (GPIOPeriphID0), offset 0xFE0 .....	274
Register 27:	GPIO Peripheral Identification 1 (GPIOPeriphID1), offset 0xFE4 .....	275
Register 28:	GPIO Peripheral Identification 2 (GPIOPeriphID2), offset 0xFE8 .....	276
Register 29:	GPIO Peripheral Identification 3 (GPIOPeriphID3), offset 0xFEC .....	277
Register 30:	GPIO PrimeCell Identification 0 (GPIOPCellID0), offset 0xFF0 .....	278
Register 31:	GPIO PrimeCell Identification 1 (GPIOPCellID1), offset 0xFF4 .....	279
Register 32:	GPIO PrimeCell Identification 2 (GPIOPCellID2), offset 0xFF8 .....	280
Register 33:	GPIO PrimeCell Identification 3 (GPIOPCellID3), offset 0xFFC .....	281
<b>General-Purpose Timers .....</b>		<b>282</b>
Register 1:	GPTM Configuration (GPTMCFG), offset 0x000 .....	294
Register 2:	GPTM TimerA Mode (GPTMTAMR), offset 0x004 .....	295
Register 3:	GPTM TimerB Mode (GPTMTBMR), offset 0x008 .....	297
Register 4:	GPTM Control (GPTMCTL), offset 0x00C .....	299
Register 5:	GPTM Interrupt Mask (GPTMIMR), offset 0x018 .....	302
Register 6:	GPTM Raw Interrupt Status (GPTMRIS), offset 0x01C .....	304

Register 7:	GPTM Masked Interrupt Status (GPTMMIS), offset 0x020 .....	305
Register 8:	GPTM Interrupt Clear (GPTMICR), offset 0x024 .....	306
Register 9:	GPTM TimerA Interval Load (GPTMTAILR), offset 0x028 .....	308
Register 10:	GPTM TimerB Interval Load (GPTMTBILR), offset 0x02C .....	309
Register 11:	GPTM TimerA Match (GPTMTAMATCHR), offset 0x030 .....	310
Register 12:	GPTM TimerB Match (GPTMTBMATCHR), offset 0x034 .....	311
Register 13:	GPTM TimerA Prescale (GPTMTAPR), offset 0x038 .....	312
Register 14:	GPTM TimerB Prescale (GPTMTBPR), offset 0x03C .....	313
Register 15:	GPTM TimerA (GPTMTAR), offset 0x048 .....	314
Register 16:	GPTM TimerB (GPTMTBR), offset 0x04C .....	315
<b>Watchdog Timer .....</b>	<b>316</b>	
Register 1:	Watchdog Load (WDTLOAD), offset 0x000 .....	319
Register 2:	Watchdog Value (WDTVALUE), offset 0x004 .....	320
Register 3:	Watchdog Control (WDTCTL), offset 0x008 .....	321
Register 4:	Watchdog Interrupt Clear (WDTICR), offset 0x00C .....	322
Register 5:	Watchdog Raw Interrupt Status (WDTRIS), offset 0x010 .....	323
Register 6:	Watchdog Masked Interrupt Status (WDTMIS), offset 0x014 .....	324
Register 7:	Watchdog Test (WDTTEST), offset 0x418 .....	325
Register 8:	Watchdog Lock (WDTLOCK), offset 0xC00 .....	326
Register 9:	Watchdog Peripheral Identification 4 (WDTPeriphID4), offset 0xFD0 .....	327
Register 10:	Watchdog Peripheral Identification 5 (WDTPeriphID5), offset 0xFD4 .....	328
Register 11:	Watchdog Peripheral Identification 6 (WDTPeriphID6), offset 0xFD8 .....	329
Register 12:	Watchdog Peripheral Identification 7 (WDTPeriphID7), offset 0xFDC .....	330
Register 13:	Watchdog Peripheral Identification 0 (WDTPeriphID0), offset 0xFE0 .....	331
Register 14:	Watchdog Peripheral Identification 1 (WDTPeriphID1), offset 0xFE4 .....	332
Register 15:	Watchdog Peripheral Identification 2 (WDTPeriphID2), offset 0xFE8 .....	333
Register 16:	Watchdog Peripheral Identification 3 (WDTPeriphID3), offset 0xFEC .....	334
Register 17:	Watchdog PrimeCell Identification 0 (WDTPCellID0), offset 0xFF0 .....	335
Register 18:	Watchdog PrimeCell Identification 1 (WDTPCellID1), offset 0xFF4 .....	336
Register 19:	Watchdog PrimeCell Identification 2 (WDTPCellID2), offset 0xFF8 .....	337
Register 20:	Watchdog PrimeCell Identification 3 (WDTPCellID3), offset 0xFFC .....	338
<b>Analog-to-Digital Converter (ADC) .....</b>	<b>339</b>	
Register 1:	ADC Active Sample Sequencer (ADCACTSS), offset 0x000 .....	347
Register 2:	ADC Raw Interrupt Status (ADCRIS), offset 0x004 .....	348
Register 3:	ADC Interrupt Mask (ADCIM), offset 0x008 .....	349
Register 4:	ADC Interrupt Status and Clear (ADCISC), offset 0x00C .....	350
Register 5:	ADC Overflow Status (ADCOSTAT), offset 0x010 .....	351
Register 6:	ADC Event Multiplexer Select (ADCEMUX), offset 0x014 .....	352
Register 7:	ADC Underflow Status (ADCUSTAT), offset 0x018 .....	355
Register 8:	ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020 .....	356
Register 9:	ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028 .....	357
Register 10:	ADC Sample Averaging Control (ADCSAC), offset 0x030 .....	358
Register 11:	ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040 .....	359
Register 12:	ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044 .....	361
Register 13:	ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048 .....	364
Register 14:	ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068 .....	364
Register 15:	ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088 .....	364
Register 16:	ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8 .....	364



Register 17:	ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C .....	365
Register 18:	ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C .....	365
Register 19:	ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C .....	365
Register 20:	ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC .....	365
Register 21:	ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060 .....	366
Register 22:	ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080 .....	366
Register 23:	ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064 .....	367
Register 24:	ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084 .....	367
Register 25:	ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0 .....	369
Register 26:	ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4 .....	370
<b>Universal Asynchronous Receivers/Transmitters (UARTs) .....</b>		<b>371</b>
Register 1:	UART Data (UARTDR), offset 0x000 .....	380
Register 2:	UART Receive Status/Error Clear (UARTRSR/UARTECR), offset 0x004 .....	382
Register 3:	UART Flag (UARTFR), offset 0x018 .....	384
Register 4:	UART IrDA Low-Power Register (UARTILPR), offset 0x020 .....	386
Register 5:	UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024 .....	387
Register 6:	UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028 .....	388
Register 7:	UART Line Control (UARTLCRH), offset 0x02C .....	389
Register 8:	UART Control (UARTCTL), offset 0x030 .....	391
Register 9:	UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034 .....	393
Register 10:	UART Interrupt Mask (UARTIM), offset 0x038 .....	395
Register 11:	UART Raw Interrupt Status (UARTRIS), offset 0x03C .....	397
Register 12:	UART Masked Interrupt Status (UARTMIS), offset 0x040 .....	398
Register 13:	UART Interrupt Clear (UARTICR), offset 0x044 .....	399
Register 14:	UART DMA Control (UARTDMACTL), offset 0x048 .....	401
Register 15:	UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0 .....	402
Register 16:	UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4 .....	403
Register 17:	UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8 .....	404
Register 18:	UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC .....	405
Register 19:	UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0 .....	406
Register 20:	UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4 .....	407
Register 21:	UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8 .....	408
Register 22:	UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC .....	409
Register 23:	UART PrimeCell Identification 0 (UARTPCIID0), offset 0xFF0 .....	410
Register 24:	UART PrimeCell Identification 1 (UARTPCIID1), offset 0xFF4 .....	411
Register 25:	UART PrimeCell Identification 2 (UARTPCIID2), offset 0xFF8 .....	412
Register 26:	UART PrimeCell Identification 3 (UARTPCIID3), offset 0xFFC .....	413
<b>Synchronous Serial Interface (SSI) .....</b>		<b>414</b>
Register 1:	SSI Control 0 (SSICR0), offset 0x000 .....	427
Register 2:	SSI Control 1 (SSICR1), offset 0x004 .....	429
Register 3:	SSI Data (SSIDR), offset 0x008 .....	431
Register 4:	SSI Status (SSISR), offset 0x00C .....	432
Register 5:	SSI Clock Prescale (SSICPSR), offset 0x010 .....	434
Register 6:	SSI Interrupt Mask (SSIIM), offset 0x014 .....	435
Register 7:	SSI Raw Interrupt Status (SSIRIS), offset 0x018 .....	437
Register 8:	SSI Masked Interrupt Status (SSIMIS), offset 0x01C .....	438
Register 9:	SSI Interrupt Clear (SSIICR), offset 0x020 .....	439
Register 10:	SSI DMA Control (SSIDMACTL), offset 0x024 .....	440

Register 11:	SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0 .....	441
Register 12:	SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4 .....	442
Register 13:	SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8 .....	443
Register 14:	SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC .....	444
Register 15:	SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0 .....	445
Register 16:	SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4 .....	446
Register 17:	SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8 .....	447
Register 18:	SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC .....	448
Register 19:	SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0 .....	449
Register 20:	SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4 .....	450
Register 21:	SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8 .....	451
Register 22:	SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC .....	452
<b>Inter-Integrated Circuit (I<sup>2</sup>C) Interface .....</b>		<b>453</b>
Register 1:	I <sup>2</sup> C Master Slave Address (I2CMSA), offset 0x000 .....	467
Register 2:	I <sup>2</sup> C Master Control/Status (I2CMCS), offset 0x004 .....	468
Register 3:	I <sup>2</sup> C Master Data (I2CMDR), offset 0x008 .....	472
Register 4:	I <sup>2</sup> C Master Timer Period (I2CMTPR), offset 0x00C .....	473
Register 5:	I <sup>2</sup> C Master Interrupt Mask (I2CMIMR), offset 0x010 .....	474
Register 6:	I <sup>2</sup> C Master Raw Interrupt Status (I2CMRIS), offset 0x014 .....	475
Register 7:	I <sup>2</sup> C Master Masked Interrupt Status (I2CMMIS), offset 0x018 .....	476
Register 8:	I <sup>2</sup> C Master Interrupt Clear (I2CMICR), offset 0x01C .....	477
Register 9:	I <sup>2</sup> C Master Configuration (I2CMCR), offset 0x020 .....	478
Register 10:	I <sup>2</sup> C Slave Own Address (I2CSOAR), offset 0x000 .....	480
Register 11:	I <sup>2</sup> C Slave Control/Status (I2CSCSR), offset 0x004 .....	481
Register 12:	I <sup>2</sup> C Slave Data (I2CSDR), offset 0x008 .....	483
Register 13:	I <sup>2</sup> C Slave Interrupt Mask (I2CSIMR), offset 0x00C .....	484
Register 14:	I <sup>2</sup> C Slave Raw Interrupt Status (I2CSRIS), offset 0x010 .....	485
Register 15:	I <sup>2</sup> C Slave Masked Interrupt Status (I2CSMIS), offset 0x014 .....	486
Register 16:	I <sup>2</sup> C Slave Interrupt Clear (I2CSICR), offset 0x018 .....	487

## About This Document

This data sheet provides reference information for the LM3S1607 microcontroller, describing the functional blocks of the system-on-chip (SoC) device designed around the ARM® Cortex™-M3 core.

### Audience

This manual is intended for system software developers, hardware designers, and application developers.

### About This Manual

This document is organized into sections that correspond to each major feature.

### Related Documents

The following documents are referenced by the data sheet, and available on the documentation CD or from the Luminary Micro web site at [www.luminarymicro.com](http://www.luminarymicro.com):

- *ARM® Cortex™-M3 Technical Reference Manual*
- *ARM® CoreSight Technical Reference Manual*
- *ARM® v7-M Architecture Application Level Reference Manual*
- *Stellaris® Peripheral Driver Library User's Guide*
- *Stellaris® ROM User's Guide*

The following related documents are also referenced:

- *IEEE Standard 1149.1-Test Access Port and Boundary-Scan Architecture*

This documentation list was current as of publication date. Please check the Luminary Micro web site for additional documentation, including application notes and white papers.

### Documentation Conventions

This document uses the conventions shown in Table 1 on page 19.

**Table 1. Documentation Conventions**

Notation	Meaning
<b>General Register Notation</b>	
<b>REGISTER</b>	APB registers are indicated in uppercase bold. For example, <b>PBORCTL</b> is the Power-On and Brown-Out Reset Control register. If a register name contains a lowercase n, it represents more than one register. For example, <b>SRCRn</b> represents any (or all) of the three Software Reset Control registers: <b>SRCR0</b> , <b>SRCR1</b> , and <b>SRCR2</b> .
bit	A single bit in a register.
bit field	Two or more consecutive and related bits.
offset 0xnnn	A hexadecimal increment to a register's address, relative to that module's base address as specified in "Memory Map" on page 42.

Notation	Meaning
Register <i>N</i>	Registers are numbered consecutively throughout the document to aid in referencing them. The register number has no meaning to software.
reserved	Register bits marked <i>reserved</i> are reserved for future use. In most cases, reserved bits are set to 0; however, user software should not rely on the value of a reserved bit. To provide software compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
yy:xx	The range of register bits inclusive from xx to yy. For example, 31:15 means bits 15 through 31 in that register.
<b>Register Bit/Field Types</b>	This value in the register bit diagram indicates whether software running on the controller can change the value of the bit field.
RC	Software can read this field. The bit or field is cleared by hardware after reading the bit/field.
RO	Software can read this field. Always write the chip reset value.
R/W	Software can read or write this field.
R/W1C	Software can read or write this field. A write of a 0 to a W1C bit does not affect the bit value in the register. A write of a 1 clears the value of the bit in the register; the remaining bits remain unchanged.  This register type is primarily used for clearing interrupt status bits where the read operation provides the interrupt status and the write of the read value clears only the interrupts being reported at the time the register was read.
R/W1S	Software can read or write a 1 to this field. A write of a 0 to a R/W1S bit does not affect the bit value in the register.
W1C	Software can write this field. A write of a 0 to a W1C bit does not affect the bit value in the register. A write of a 1 clears the value of the bit in the register; the remaining bits remain unchanged. A read of the register returns no meaningful data.  This register is typically used to clear the corresponding bit in an interrupt register.
WO	Only a write by software is valid; a read of the register returns no meaningful data.
<b>Register Bit/Field Reset Value</b>	This value in the register bit diagram shows the bit/field value after any reset, unless noted.
0	Bit cleared to 0 on chip reset.
1	Bit set to 1 on chip reset.
-	Nondeterministic.
<b>Pin/Signal Notation</b>	
[ ]	Pin alternate function; a pin defaults to the signal without the brackets.
pin	Refers to the physical connection on the package.
signal	Refers to the electrical signal encoding of a pin.
assert a signal	Change the value of the signal from the logically False state to the logically True state. For active High signals, the asserted signal value is 1 (High); for active Low signals, the asserted signal value is 0 (Low). The active polarity (High or Low) is defined by the signal name (see <code>SIGNAL</code> and <code>̄SIGNAL</code> below).
deassert a signal	Change the value of the signal from the logically True state to the logically False state.
<code>̄SIGNAL</code>	Signal names are in uppercase and in the Courier font. An overbar on a signal name indicates that it is active Low. To assert <code>̄SIGNAL</code> is to drive it Low; to deassert <code>̄SIGNAL</code> is to drive it High.
<code>SIGNAL</code>	Signal names are in uppercase and in the Courier font. An active High signal has no overbar. To assert <code>SIGNAL</code> is to drive it High; to deassert <code>SIGNAL</code> is to drive it Low.
<b>Numbers</b>	
X	An uppercase X indicates any of several values is allowed, where X can be any legal pattern. For example, a binary value of 0X00 can be either 0100 or 0000, a hex value of 0xX is 0x0 or 0x1, and so on.

Notation	Meaning
0x	Hexadecimal numbers have a prefix of 0x. For example, 0x00FF is the hexadecimal number FF.  All other numbers within register tables are assumed to be binary. Within conceptual information, binary numbers are indicated with a b suffix, for example, 1011b, and decimal numbers are written without a prefix or suffix.

# 1 Architectural Overview

The Luminary Micro Stellaris® family of microcontrollers—the first ARM® Cortex™-M3 based controllers—brings high-performance 32-bit computing to cost-sensitive embedded microcontroller applications. These pioneering parts deliver customers 32-bit performance at a cost equivalent to legacy 8- and 16-bit devices, all in a package with a small footprint.

The Stellaris® family offers efficient performance and extensive integration, favorably positioning the device into cost-conscious applications requiring significant control-processing and connectivity capabilities. The Stellaris® LM3S1000 series extends the Stellaris® family with larger on-chip memories, enhanced power management, and expanded I/O and control capabilities.

The LM3S1607 microcontroller is targeted for industrial applications, including remote monitoring, electronic point-of-sale machines, test and measurement equipment, network appliances and switches, factory automation, HVAC and building control, gaming equipment, motion control, medical instrumentation, and fire and security.

For applications requiring extreme conservation of power, the LM3S1607 microcontroller features a Battery-backed Hibernation module to efficiently power down the LM3S1607 to a low-power state during extended periods of inactivity. With a power-up/power-down sequencer, a continuous time counter (RTC), a pair of match registers, an APB interface to the system bus, and dedicated non-volatile memory, the Hibernation module positions the LM3S1607 microcontroller perfectly for battery applications.

In addition, the LM3S1607 microcontroller offers the advantages of ARM's widely available development tools, System-on-Chip (SoC) infrastructure IP applications, and a large user community. Additionally, the microcontroller uses ARM's Thumb®-compatible Thumb-2 instruction set to reduce memory requirements and, thereby, cost. Finally, the LM3S1607 microcontroller is code-compatible to all members of the extensive Stellaris® family; providing flexibility to fit our customers' precise needs.

Luminary Micro offers a complete solution to get to market quickly, with evaluation and development boards, white papers and application notes, an easy-to-use peripheral driver library, and a strong support, sales, and distributor network. See "Ordering and Contact Information" on page 547 for ordering information for Stellaris® family devices.

## 1.1 Product Features

The LM3S1607 microcontroller includes the following product features:

- 32-Bit RISC Performance
  - 32-bit ARM® Cortex™-M3 v7M architecture optimized for small-footprint embedded applications
  - System timer (SysTick), providing a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism
  - Thumb®-compatible Thumb-2-only instruction set processor core for high code density
  - 50-MHz operation
  - Hardware-division and single-cycle-multiplication

- Integrated Nested Vectored Interrupt Controller (NVIC) providing deterministic interrupt handling
- 27 interrupts with eight priority levels
- Memory protection unit (MPU), providing a privileged mode for protected operating system functionality
- Unaligned data access, enabling data to be efficiently packed into memory
- Atomic bit manipulation (bit-banding), delivering maximum memory utilization and streamlined peripheral control
- Internal Memory
  - 128 KB single-cycle flash
    - User-managed flash block protection on a 2-KB block basis
    - User-managed flash data programming
    - User-defined and managed flash-protection block
  - 32 KB single-cycle SRAM
  - Pre-programmed ROM containing the Stellaris<sup>®</sup> family peripheral driver library (DriverLib) and Stellaris<sup>®</sup> boot loader
- DMA Controller
  - ARM PrimeCell<sup>®</sup> 32-channel configurable  $\mu$ DMA controller
  - Support for multiple transfer modes:
    - Basic, for simple transfer scenarios
    - Ping-pong, for continuous data flow to/from peripherals
    - Scatter-gather, from a programmable list of arbitrary transfers initiated from a single request
  - Dedicated channels for supported peripherals
  - One channel each for receive and transmit path for bidirectional peripherals
  - Dedicated channel for software-initiated transfers
  - Independently configured and operated channels
  - Per-channel configurable bus arbitration scheme
  - Two levels of priority
  - Design optimizations for improved bus access performance between  $\mu$ DMA controller and the processor core:
    - $\mu$ DMA controller access is subordinate to core access

- RAM striping
- Peripheral bus segmentation
- Data sizes of 8, 16, and 32 bits
- Source and destination address increment size of byte, half-word, word, or no increment
- Maskable device requests
- Optional software initiated requests for any channel
- Interrupt on transfer completion, with a separate interrupt per channel
- General-Purpose Timers
  - Four General-Purpose Timer Modules (GPTM), each of which provides two 16-bit timers. Each GPTM can be configured to operate independently:
    - As a single 32-bit timer
    - As one 32-bit Real-Time Clock (RTC) to event capture
    - For Pulse Width Modulation (PWM)
    - To trigger analog-to-digital conversions
  - 32-bit Timer modes
    - Programmable one-shot timer
    - Programmable periodic timer
    - Real-Time Clock when using an external 32.768-KHz clock as the input
    - User-enabled stalling in periodic and one-shot mode when the controller asserts the CPU Halt flag during debug
    - ADC event trigger
  - 16-bit Timer modes
    - General-purpose timer function with an 8-bit prescaler
    - Programmable one-shot timer
    - Programmable periodic timer
    - User-enabled stalling when the controller asserts CPU Halt flag during debug
    - ADC event trigger
  - 16-bit Input Capture modes
    - Input edge count capture
    - Input edge time capture



- 16-bit PWM mode
  - Simple PWM mode with software-programmable output inversion of the PWM signal
- ARM FiRM-compliant Watchdog Timer
  - 32-bit down counter with a programmable load register
  - Separate watchdog clock with an enable
  - Programmable interrupt generation logic with interrupt masking
  - Lock register protection from runaway software
  - Reset generation logic with an enable/disable
  - User-enabled stalling when the controller asserts the CPU Halt flag during debug
- Synchronous Serial Interface (SSI)
  - Master or slave operation
  - Programmable clock bit rate and prescale
  - Separate transmit and receive FIFOs, 16 bits wide, 8 locations deep
  - Programmable interface operation for Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces
  - Programmable data frame size from 4 to 16 bits
  - Internal loopback test mode for diagnostic/debug testing
  - Direct memory access (DMA)
- UART
  - Three fully programmable 16C550-type UARTs with IrDA support
  - Separate 16x8 transmit (TX) and 16x12 receive (RX) FIFOs to reduce CPU interrupt service loading
  - Programmable baud-rate generator allowing speeds up to 3.125 Mbps
  - Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
  - FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
  - Standard asynchronous communication bits for start, stop, and parity
  - False-start-bit detection
  - Line-break generation and detection
  - Direct memory access (DMA)

- ADC
  - Single- and differential-input configurations
  - Eight 10-bit channels (inputs) when used as single-ended inputs
  - Sample rate of 500 thousand samples/second
  - Flexible, configurable analog-to-digital conversion
  - Four programmable sample conversion sequences from one to eight entries long, with corresponding conversion result FIFOs
  - Each sequence triggered by software or internal event (timers, or GPIO)
  - On-chip temperature sensor
- I<sup>2</sup>C
  - Two I<sup>2</sup>C modules
  - Master and slave receive and transmit operation with transmission speed up to 100 Kbps in Standard mode and 400 Kbps in Fast mode
  - Interrupt generation
  - Master with arbitration and clock synchronization, multimaster support, and 7-bit addressing mode
- GPIOs
  - 0-33 GPIOs, depending on configuration
  - 5-V-tolerant input/outputs
  - Programmable interrupt generation as either edge-triggered or level-sensitive
  - Low interrupt latency; as low as 6 cycles and never more than 12 cycles
  - Bit masking in both read and write operations through address lines
  - Can initiate an ADC sample sequence
  - Pins configured as digital inputs are Schmitt-triggered.
  - Programmable control for GPIO pad configuration:
    - Weak pull-up or pull-down resistors
    - 2-mA, 4-mA, and 8-mA pad drive for digital communication; up to four pads can be configured with an 18-mA pad drive for high-current applications
    - Slew rate control for the 8-mA drive
    - Open drain enables

- Digital input enables
- Power
  - On-chip Low Drop-Out (LDO) voltage regulator, with programmable output user-adjustable from 2.25 V to 2.75 V
  - Hibernation module handles the power-up/down 3.3 V sequencing and control for the core digital logic and analog circuits
  - Low-power options on controller: Sleep and Deep-sleep modes
  - Low-power options for peripherals: software controls shutdown of individual peripherals
  - User-enabled LDO unregulated voltage detection and automatic reset
  - 3.3-V supply brown-out detection and reporting via interrupt or reset
- Flexible Reset Sources
  - Power-on reset (POR)
  - Reset pin assertion
  - Brown-out (BOR) detector alerts to system power drops
  - Software reset
  - Watchdog timer reset
  - Internal low drop-out (LDO) regulator output goes unregulated
- Additional Features
  - Six reset sources
  - Programmable clock source control
  - Clock gating to individual peripherals for power savings
  - IEEE 1149.1-1990 compliant Test Access Port (TAP) controller
  - Debug access via JTAG and Serial Wire interfaces
  - Full JTAG boundary scan
- Industrial-range 64-pin RoHS-compliant LQFP package

## 1.2 Target Applications

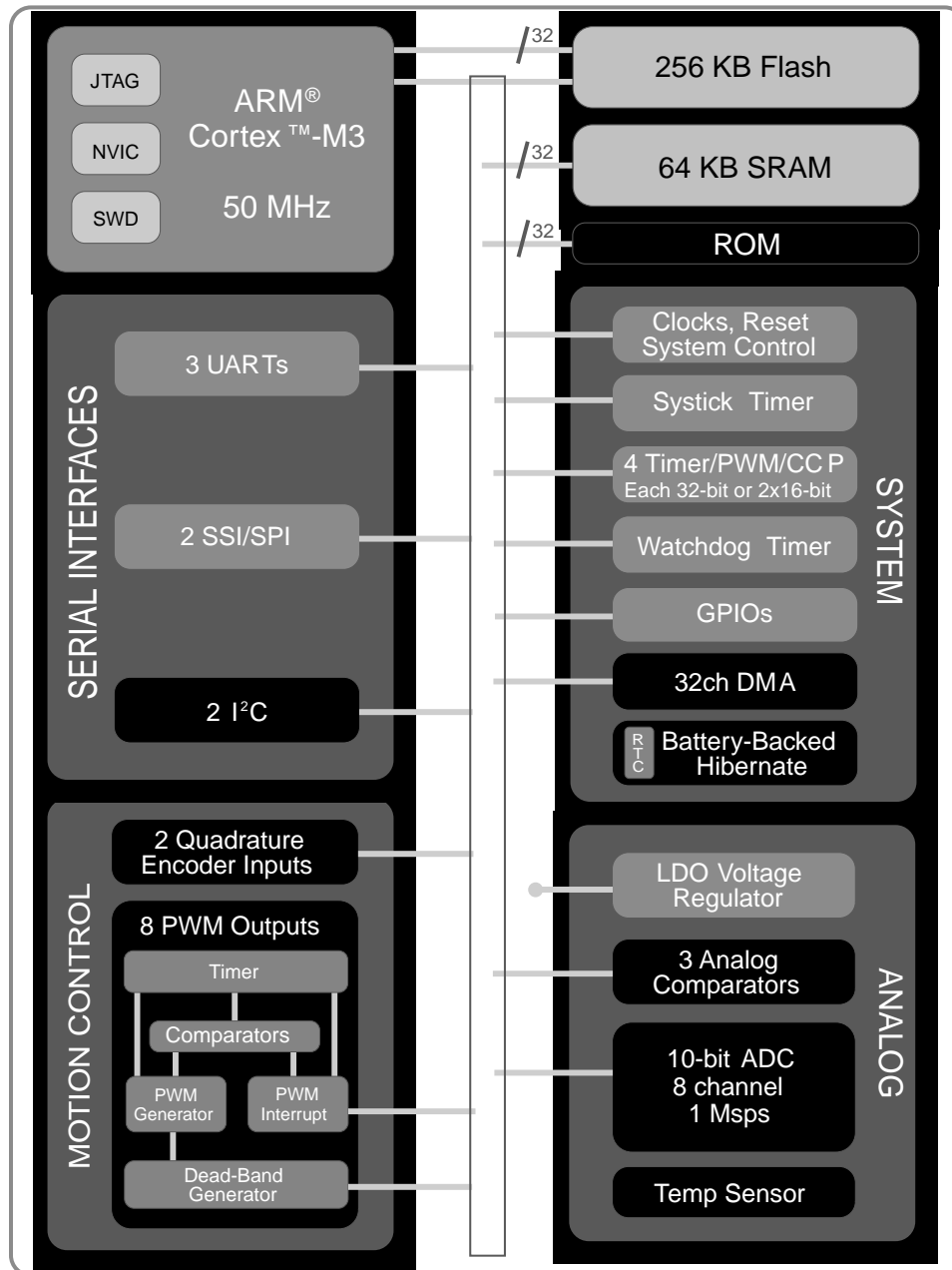
- Remote monitoring
- Electronic point-of-sale (POS) machines
- Test and measurement equipment

- Network appliances and switches
- Factory automation
- HVAC and building control
- Gaming equipment
- Motion control
- Medical instrumentation
- Fire and security
- Power and energy
- Transportation

### 1.3 High-Level Block Diagram

Figure 1-1 on page 29 represents the full set of features in the Stellaris<sup>®</sup> 1000 series of devices; not all features may be available on the LM3S1607 microcontroller.

Figure 1-1. Stellaris® 1000 Series High-Level Block Diagram



## 1.4 Functional Overview

The following sections provide an overview of the features of the LM3S1607 microcontroller. The page number in parenthesis indicates where that feature is discussed in detail. Ordering and support information can be found in “Ordering and Contact Information” on page 547.

## 1.4.1 ARM Cortex™-M3

### 1.4.1.1 Processor Core (see page 36)

All members of the Stellaris® product family, including the LM3S1607 microcontroller, are designed around an ARM Cortex™-M3 processor core. The ARM Cortex-M3 processor provides the core for a high-performance, low-cost platform that meets the needs of minimal memory implementation, reduced pin count, and low-power consumption, while delivering outstanding computational performance and exceptional system response to interrupts.

“ARM Cortex-M3 Processor Core” on page 36 provides an overview of the ARM core; the core is detailed in the *ARM® Cortex™-M3 Technical Reference Manual*.

### 1.4.1.2 System Timer (SysTick)

Cortex-M3 includes an integrated system timer, SysTick. SysTick provides a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism. The counter can be used in several different ways, for example:

- An RTOS tick timer which fires at a programmable rate (for example, 100 Hz) and invokes a SysTick routine.
- A high-speed alarm timer using the system clock.
- A variable rate alarm or signal timer—the duration is range-dependent on the reference clock used and the dynamic range of the counter.
- A simple counter. Software can use this to measure time to completion and time used.
- An internal clock source control based on missing/meeting durations. The COUNTFLAG bit-field in the control and status register can be used to determine if an action completed within a set duration, as part of a dynamic clock management control loop.

### 1.4.1.3 Nested Vectored Interrupt Controller (NVIC)

The LM3S1607 controller includes the ARM Nested Vectored Interrupt Controller (NVIC) on the ARM® Cortex™-M3 core. The NVIC and Cortex-M3 prioritize and handle all exceptions. All exceptions are handled in Handler Mode. The processor state is automatically stored to the stack on an exception, and automatically restored from the stack at the end of the Interrupt Service Routine (ISR). The vector is fetched in parallel to the state saving, which enables efficient interrupt entry. The processor supports tail-chaining, which enables back-to-back interrupts to be performed without the overhead of state saving and restoration. Software can set eight priority levels on 7 exceptions (system handlers) and 27 interrupts.

“Interrupts” on page 45 provides an overview of the NVIC controller and the interrupt map. Exceptions and interrupts are detailed in the *ARM® Cortex™-M3 Technical Reference Manual*.

### 1.4.1.4 Direct Memory Access (see page 176)

The LM3S1607 microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA (μDMA). The μDMA controller provides a way to offload data transfer tasks from the Cortex-M3 processor, allowing for more efficient use of the processor and the expanded available bus bandwidth. The μDMA controller can perform transfers between memory and peripherals. It has dedicated channels for each supported peripheral and can be programmed to automatically perform transfers between peripherals and memory as the peripheral is ready to transfer more data. The μDMA controller also supports sophisticated transfer modes such as ping-pong and scatter-gather, which allows the processor to set up a list of transfer tasks for the controller.

## 1.4.2 Motor Control Peripherals

To enhance motor control, the LM3S1607 controller features Pulse Width Modulation (PWM) outputs.

### 1.4.2.1 PWM

Pulse width modulation (PWM) is a powerful technique for digitally encoding analog signal levels. High-resolution counters are used to generate a square wave, and the duty cycle of the square wave is modulated to encode an analog signal. Typical applications include switching power supplies and motor control.

On the LM3S1607, PWM motion control functionality can be achieved through:

- The motion control features of the general-purpose timers using the CCP pins

#### **CCP Pins (see page 288)**

The General-Purpose Timer Module's CCP (Capture Compare PWM) pins are software programmable to support a simple PWM mode with a software-programmable output inversion of the PWM signal.

## 1.4.3 Analog Peripherals

To handle analog signals, the LM3S1607 microcontroller offers an Analog-to-Digital Converter (ADC).

### 1.4.3.1 ADC (see page 339)

An analog-to-digital converter (ADC) is a peripheral that converts a continuous analog voltage to a discrete digital number.

The LM3S1607 ADC module features 10-bit conversion resolution and supports eight input channels, plus an internal temperature sensor. Four buffered sample sequences allow rapid sampling of up to eight analog input sources without controller intervention. Each sample sequence provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequence priority.

## 1.4.4 Serial Communications Peripherals

The LM3S1607 controller supports both asynchronous and synchronous serial communications with:

- Three fully programmable 16C550-type UARTs
- One SSI module
- Two I<sup>2</sup>C modules

### 1.4.4.1 UART (see page 371)

A Universal Asynchronous Receiver/Transmitter (UART) is an integrated circuit used for RS-232C serial communications, containing a transmitter (parallel-to-serial converter) and a receiver (serial-to-parallel converter), each clocked separately.

The LM3S1607 controller includes three fully programmable 16C550-type UARTs that support data transfer speeds up to 3.125 Mbps. (Although similar in functionality to a 16C550 UART, it is not register-compatible.) In addition, each UART is capable of supporting IrDA.

Separate 16x8 transmit (TX) and 16x12 receive (RX) FIFOs reduce CPU interrupt service loading. The UART can generate individually masked interrupts from the RX, TX, modem status, and error

conditions. The module provides a single combined interrupt when any of the interrupts are asserted and are unmasked.

#### 1.4.4.2 SSI (see page 414)

Synchronous Serial Interface (SSI) is a four-wire bi-directional communications interface.

The LM3S1607 controller includes one SSI module that provides the functionality for synchronous serial communications with peripheral devices, and can be configured to use the Freescale SPI, MICROWIRE, or TI synchronous serial interface frame formats. The size of the data frame is also configurable, and can be set between 4 and 16 bits, inclusive.

The SSI module performs serial-to-parallel conversion on data received from a peripheral device, and parallel-to-serial conversion on data transmitted to a peripheral device. The TX and RX paths are buffered with internal FIFOs, allowing up to eight 16-bit values to be stored independently.

The SSI module can be configured as either a master or slave device. As a slave device, the SSI module can also be configured to disable its output, which allows a master device to be coupled with multiple slave devices.

The SSI module also includes a programmable bit rate clock divider and prescaler to generate the output serial clock derived from the SSI module's input clock. Bit rates are generated based on the input clock and the maximum bit rate is determined by the connected peripheral.

#### 1.4.4.3 I<sup>2</sup>C (see page 453)

The Inter-Integrated Circuit (I<sup>2</sup>C) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL).

The I<sup>2</sup>C bus interfaces to external I<sup>2</sup>C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The I<sup>2</sup>C bus may also be used for system testing and diagnostic purposes in product development and manufacture.

The LM3S1607 controller includes two I<sup>2</sup>C modules that provide the ability to communicate to other IC devices over an I<sup>2</sup>C bus. The I<sup>2</sup>C bus supports devices that can both transmit and receive (write and read) data.

Devices on the I<sup>2</sup>C bus can be designated as either a master or a slave. Each I<sup>2</sup>C module supports both sending and receiving data as either a master or a slave, and also supports the simultaneous operation as both a master and a slave. The four I<sup>2</sup>C modes are: Master Transmit, Master Receive, Slave Transmit, and Slave Receive.

A Stellaris<sup>®</sup> I<sup>2</sup>C module can operate at two speeds: Standard (100 Kbps) and Fast (400 Kbps).

Both the I<sup>2</sup>C master and slave can generate interrupts. The I<sup>2</sup>C master generates interrupts when a transmit or receive operation completes (or aborts due to an error). The I<sup>2</sup>C slave generates interrupts when data has been sent or requested by a master.

### 1.4.5 System Peripherals

#### 1.4.5.1 Programmable GPIOs (see page 237)

General-purpose input/output (GPIO) pins offer flexibility for a variety of connections.

The Stellaris<sup>®</sup> GPIO module is comprised of five physical GPIO blocks, each corresponding to an individual GPIO port. The GPIO module is FiRM-compliant (compliant to the ARM Foundation IP for Real-Time Microcontrollers specification) and supports 0-33 programmable input/output pins. The number of GPIOs available depends on the peripherals being used (see "Signal Tables" on page 489 for the signals available to each GPIO pin).



The GPIO module features programmable interrupt generation as either edge-triggered or level-sensitive on all pins, programmable control for GPIO pad configuration, and bit masking in both read and write operations through address lines. Pins configured as digital inputs are Schmitt-triggered.

#### 1.4.5.2 Four Programmable Timers (see page 282)

Programmable timers can be used to count or time external events that drive the Timer input pins.

The Stellaris<sup>®</sup> General-Purpose Timer Module (GPTM) contains four GPTM blocks. Each GPTM block provides two 16-bit timers/counters that can be configured to operate independently as timers or event counters, or configured to operate as one 32-bit timer or one 32-bit Real-Time Clock (RTC). Timers can also be used to trigger analog-to-digital (ADC) conversions.

When configured in 32-bit mode, a timer can run as a Real-Time Clock (RTC), one-shot timer or periodic timer. When in 16-bit mode, a timer can run as a one-shot timer or periodic timer, and can extend its precision by using an 8-bit prescaler. A 16-bit timer can also be configured for event capture or Pulse Width Modulation (PWM) generation.

#### 1.4.5.3 Watchdog Timer (see page 316)

A watchdog timer can generate nonmaskable interrupts (NMIs) or a reset when a time-out value is reached. The watchdog timer is used to regain control when a system has failed due to a software error or to the failure of an external device to respond in the expected way.

The Stellaris<sup>®</sup> Watchdog Timer module consists of a 32-bit down counter, a programmable load register, interrupt generation logic, and a locking register.

The Watchdog Timer can be configured to generate an interrupt to the controller on its first time-out, and to generate a reset signal on its second time-out. Once the Watchdog Timer has been configured, the lock register can be written to prevent the timer configuration from being inadvertently altered.

### 1.4.6 Memory Peripherals

The LM3S1607 controller offers both single-cycle SRAM and single-cycle Flash memory.

#### 1.4.6.1 SRAM (see page 147)

The LM3S1607 static random access memory (SRAM) controller supports 32 KB SRAM. The internal SRAM of the Stellaris<sup>®</sup> devices is located at offset 0x0000.0000 of the device memory map. To reduce the number of time-consuming read-modify-write (RMW) operations, ARM has introduced *bit-banding* technology in the new Cortex-M3 processor. With a bit-band-enabled processor, certain regions in the memory map (SRAM and peripheral space) can use address aliases to access individual bits in a single, atomic operation.

#### 1.4.6.2 Flash (see page 148)

The LM3S1607 Flash controller supports 128 KB of flash memory. The flash is organized as a set of 1-KB blocks that can be individually erased. Erasing a block causes the entire contents of the block to be reset to all 1s. These blocks are paired into a set of 2-KB blocks that can be individually protected. The blocks can be marked as read-only or execute-only, providing different levels of code protection. Read-only blocks cannot be erased or programmed, protecting the contents of those blocks from being modified. Execute-only blocks cannot be erased or programmed, and can only be read by the controller instruction fetch mechanism, protecting the contents of those blocks from being read by either the controller or by a debugger.

### 1.4.6.3 ROM

The LM3S1607 microcontroller ships with the Stellaris<sup>®</sup> family Peripheral Driver Library conveniently preprogrammed in read-only memory (ROM). The Stellaris<sup>®</sup> Peripheral Driver Library is a royalty-free software library for controlling on-chip peripherals, and includes a boot-loader capability. The library performs both peripheral initialization and peripheral control functions, with a choice of polled or interrupt-driven peripheral support, and takes full advantage of the stellar interrupt performance of the ARM<sup>®</sup> Cortex<sup>™</sup>-M3 core. No special pragmas or custom assembly code prologue/epilogue functions are required. For applications that require in-field programmability, the royalty-free Stellaris<sup>®</sup> boot loader included in the Stellaris<sup>®</sup> Peripheral Driver Library can act as an application loader and support in-field firmware updates.

## 1.4.7 Additional Features

### 1.4.7.1 Memory Map (see page 42)

A memory map lists the location of instructions and data in memory. The memory map for the LM3S1607 controller can be found in “Memory Map” on page 42. Register addresses are given as a hexadecimal increment, relative to the module’s base address as shown in the memory map.

The *ARM<sup>®</sup> Cortex<sup>™</sup>-M3 Technical Reference Manual* provides further information on the memory map.

### 1.4.7.2 JTAG TAP Controller (see page 48)

The Joint Test Action Group (JTAG) port is an IEEE standard that defines a Test Access Port and Boundary Scan Architecture for digital integrated circuits and provides a standardized serial interface for controlling the associated test logic. The TAP, Instruction Register (IR), and Data Registers (DR) can be used to test the interconnections of assembled printed circuit boards and obtain manufacturing information on the components. The JTAG Port also provides a means of accessing and controlling design-for-test features such as I/O pin observation and control, scan testing, and debugging.

The JTAG port is composed of the standard four pins: TCK, TMS, TDI, and TDO. Data is transmitted serially into the controller on TDI and out of the controller on TDO. The interpretation of this data is dependent on the current state of the TAP controller. For detailed information on the operation of the JTAG port and TAP controller, please refer to the *IEEE Standard 1149.1-Test Access Port and Boundary-Scan Architecture*.

The Luminary Micro JTAG controller works with the ARM JTAG controller built into the Cortex-M3 core. This is implemented by multiplexing the TDO outputs from both JTAG controllers. ARM JTAG instructions select the ARM TDO output while Luminary Micro JTAG instructions select the Luminary Micro TDO outputs. The multiplexer is controlled by the Luminary Micro JTAG controller, which has comprehensive programming for the ARM, Luminary Micro, and unimplemented JTAG instructions.

### 1.4.7.3 System Control and Clocks (see page 60)

System control determines the overall operation of the device. It provides information about the device, controls the clocking of the device and individual peripherals, and handles reset detection and reporting.

### 1.4.7.4 Hibernation Module (see page 124)

The Hibernation module provides logic to switch power off to the main processor and peripherals, and to wake on external or time-based events. The Hibernation module includes power-sequencing logic, a real-time clock with a pair of match registers, low-battery detection circuitry, and interrupt signalling to the processor. It also includes 64 32-bit words of non-volatile memory that can be used for saving state during hibernation.

## 1.4.8 Hardware Details

Details on the pins and package can be found in the following sections:

- “Pin Diagram” on page 488
- “Signal Tables” on page 489
- “Operating Characteristics” on page 499
- “Electrical Characteristics” on page 500
- “Package Information” on page 512

## 2 ARM Cortex-M3 Processor Core

The ARM Cortex-M3 processor provides the core for a high-performance, low-cost platform that meets the needs of minimal memory implementation, reduced pin count, and low power consumption, while delivering outstanding computational performance and exceptional system response to interrupts. Features include:

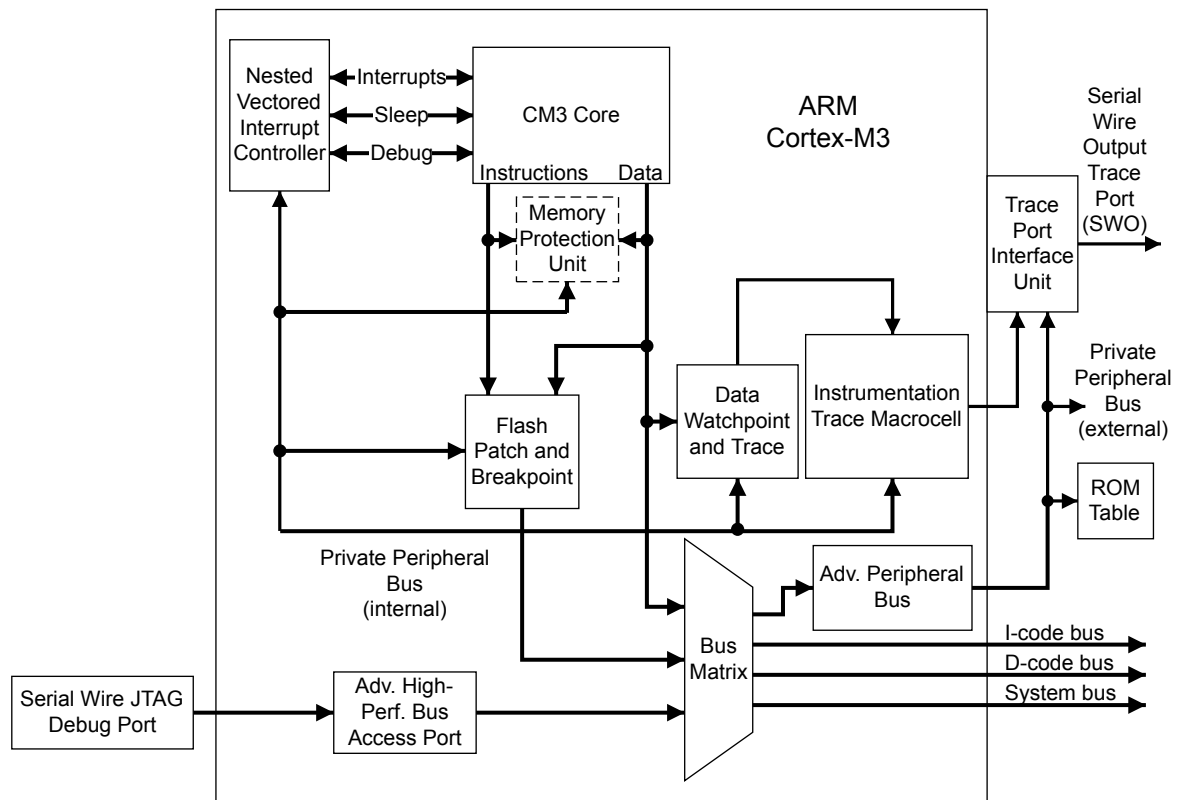
- Compact core.
- Thumb-2 instruction set, delivering the high-performance expected of an ARM core in the memory size usually associated with 8- and 16-bit devices; typically in the range of a few kilobytes of memory for microcontroller class applications.
- Rapid application execution through Harvard architecture characterized by separate buses for instruction and data.
- Exceptional interrupt handling, by implementing the register manipulations required for handling an interrupt in hardware.
- Deterministic, fast interrupt processing: always 12 cycles, or just 6 cycles with tail-chaining
- External non-maskable interrupt signal (NMI) available for immediate execution of NMI handler for safety critical applications.
- Memory protection unit (MPU) to provide a privileged mode of operation for complex applications.
- Migration from the ARM7™ processor family for better performance and power efficiency.
- Full-featured debug solution with a:
  - Serial Wire JTAG Debug Port (SWJ-DP)
  - Flash Patch and Breakpoint (FPB) unit for implementing breakpoints
  - Data Watchpoint and Trigger (DWT) unit for implementing watchpoints, trigger resources, and system profiling
  - Instrumentation Trace Macrocell (ITM) for support of printf style debugging
  - Trace Port Interface Unit (TPIU) for bridging to a Trace Port Analyzer
- Optimized for single-cycle flash usage
- Three sleep modes with clock gating for low power
- Single-cycle multiply instruction and hardware divide
- Atomic operations
- ARM Thumb2 mixed 16-/32-bit instruction set
- 1.25 DMIPS/MHz

The Stellaris<sup>®</sup> family of microcontrollers builds on this core to bring high-performance 32-bit computing to cost-sensitive embedded microcontroller applications, such as factory automation and control, industrial control power devices, building and home automation, and stepper motors.

For more information on the ARM Cortex-M3 processor core, see the *ARM<sup>®</sup> Cortex<sup>™</sup>-M3 Technical Reference Manual*. For information on SWJ-DP, see the *ARM<sup>®</sup> CoreSight Technical Reference Manual*.

## 2.1 Block Diagram

Figure 2-1. CPU Block Diagram



## 2.2 Functional Description

**Important:** The *ARM<sup>®</sup> Cortex<sup>™</sup>-M3 Technical Reference Manual* describes all the features of an ARM Cortex-M3 in detail. However, these features differ based on the implementation. This section describes the Stellaris<sup>®</sup> implementation.

Luminary Micro has implemented the ARM Cortex-M3 core as shown in Figure 2-1 on page 37. As noted in the *ARM<sup>®</sup> Cortex<sup>™</sup>-M3 Technical Reference Manual*, several Cortex-M3 components are flexible in their implementation: SW/JTAG-DP, ETM, TPIU, the ROM table, the MPU, and the Nested Vectored Interrupt Controller (NVIC). Each of these is addressed in the sections that follow.

### 2.2.1 Serial Wire and JTAG Debug

Luminary Micro has replaced the ARM SW-DP and JTAG-DP with the ARM CoreSight™-compliant Serial Wire JTAG Debug Port (SWJ-DP) interface. This means Chapter 12, “Debug Port,” of the *ARM® Cortex™-M3 Technical Reference Manual* does not apply to Stellaris® devices.

The SWJ-DP interface combines the SWD and JTAG debug ports into one module. See the *CoreSight™ Design Kit Technical Reference Manual* for details on SWJ-DP.

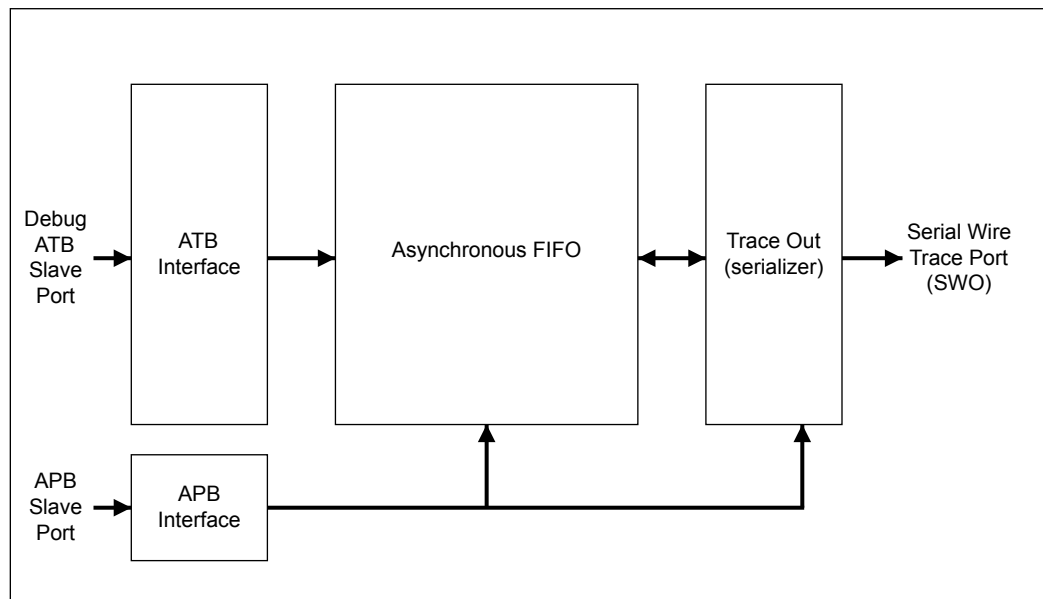
### 2.2.2 Embedded Trace Macrocell (ETM)

ETM was not implemented in the Stellaris® devices. This means Chapters 15 and 16 of the *ARM® Cortex™-M3 Technical Reference Manual* can be ignored.

### 2.2.3 Trace Port Interface Unit (TPIU)

The TPIU acts as a bridge between the Cortex-M3 trace data from the ITM, and an off-chip Trace Port Analyzer. The Stellaris® devices have implemented TPIU as shown in Figure 2-2 on page 38. This is similar to the non-ETM version described in the *ARM® Cortex™-M3 Technical Reference Manual*, however, SWJ-DP only provides SWV output for the TPIU.

**Figure 2-2. TPIU Block Diagram**



### 2.2.4 ROM Table

The default ROM table was implemented as described in the *ARM® Cortex™-M3 Technical Reference Manual*.

### 2.2.5 Memory Protection Unit (MPU)

The Memory Protection Unit (MPU) is included on the LM3S1607 controller and supports the standard ARMv7 Protected Memory System Architecture (PMSA) model. The MPU provides full support for protection regions, overlapping protection regions, access permissions, and exporting memory attributes to the system.

## 2.2.6 Nested Vectored Interrupt Controller (NVIC)

The Nested Vectored Interrupt Controller (NVIC):

- Facilitates low-latency exception and interrupt handling
- Controls power management
- Implements system control registers

The NVIC supports up to 240 dynamically reprioritizable interrupts each with up to 256 levels of priority. The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts. The NVIC maintains knowledge of the stacked (nested) interrupts to enable tail-chaining of interrupts.

You can only fully access the NVIC from privileged mode, but you can pend interrupts in user-mode if you enable the Configuration Control Register (see the ARM® Cortex™-M3 Technical Reference Manual). Any other user-mode access causes a bus fault.

All NVIC registers are accessible using byte, halfword, and word unless otherwise stated.

### 2.2.6.1 Interrupts

The *ARM® Cortex™-M3 Technical Reference Manual* describes the maximum number of interrupts and interrupt priorities. The LM3S1607 microcontroller supports 27 interrupts with eight priority levels.

In addition to the peripheral interrupts, the system also provides for a non-maskable interrupt. The NMI is generally used in safety critical applications where the immediate execution of an interrupt handler is required. The NMI signal is available as an external signal so that it may be generated by external circuitry. The NMI is also used internally as part of the main oscillator verification circuitry. More information on the non-maskable interrupt is located in “Non-Maskable Interrupt” on page 63.

### 2.2.6.2 System Timer (SysTick)

Cortex-M3 includes an integrated system timer, SysTick. SysTick provides a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism. The counter can be used in several different ways, for example:

- An RTOS tick timer which fires at a programmable rate (for example, 100 Hz) and invokes a SysTick routine.
- A high-speed alarm timer using the system clock.
- A variable rate alarm or signal timer—the duration is range-dependent on the reference clock used and the dynamic range of the counter.
- A simple counter. Software can use this to measure time to completion and time used.
- An internal clock source control based on missing/meeting durations. The COUNTFLAG bit-field in the control and status register can be used to determine if an action completed within a set duration, as part of a dynamic clock management control loop.

#### **Functional Description**

The timer consists of three registers:

- A control and status counter to configure its clock, enable the counter, enable the SysTick interrupt, and determine counter status.
- The reload value for the counter, used to provide the counter's wrap value.
- The current value of the counter.

A fourth register, the SysTick Calibration Value Register, is not implemented in the Stellaris® devices.

When enabled, the timer counts down from the reload value to zero, reloads (wraps) to the value in the SysTick Reload Value register on the next clock edge, then decrements on subsequent clocks. Writing a value of zero to the Reload Value register disables the counter on the next wrap. When the counter reaches zero, the COUNTFLAG status bit is set. The COUNTFLAG bit clears on reads.

Writing to the Current Value register clears the register and the COUNTFLAG status bit. The write does not trigger the SysTick exception logic. On a read, the current value is the value of the register at the time the register is accessed.

If the core is in debug state (halted), the counter will not decrement. The timer is clocked with respect to a reference clock. The reference clock can be the core clock or an external clock source.

### **SysTick Control and Status Register**

Use the SysTick Control and Status Register to enable the SysTick features. The reset is 0x0000.0000.

Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	COUNTFLAG	R/W	0	Count Flag Returns 1 if timer counted to 0 since last time this was read. Clears on read by application. If read by the debugger using the DAP, this bit is cleared on read-only if the MasterType bit in the AHB-AP Control Register is set to 0. Otherwise, the COUNTFLAG bit is not changed by the debugger read.
15:3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	CLKSOURCE	R/W	0	Clock Source  Value Description 0 External reference clock. (Not implemented for Stellaris microcontrollers.) 1 Core clock  If no reference clock is provided, it is held at 1 and so gives the same time as the core clock. The core clock must be at least 2.5 times faster than the reference clock. If it is not, the count values are unpredictable.
1	TICKINT	R/W	0	Tick Interrupt  Value Description 0 Counting down to 0 does not generate the interrupt request to the NVIC. Software can use the COUNTFLAG to determine if ever counted to 0. 1 Counting down to 0 pends the SysTick handler.



Bit/Field	Name	Type	Reset	Description
0	ENABLE	R/W	0	Enable  Value Description 0 Counter disabled. 1 Counter operates in a multi-shot way. That is, counter loads with the Reload value and then begins counting down. On reaching 0, it sets the COUNTFLAG to 1 and optionally pends the SysTick handler, based on TICKINT. It then loads the Reload value again, and begins counting.

### ***SysTick Reload Value Register***

Use the SysTick Reload Value Register to specify the start value to load into the current value register when the counter reaches 0. It can be any value between 1 and 0x00FF.FFFF. A start value of 0 is possible, but has no effect because the SysTick interrupt and COUNTFLAG are activated when counting from 1 to 0.

Therefore, as a multi-shot timer, repeated over and over, it fires every N+1 clock pulse, where N is any value from 1 to 0x00FF.FFFF. So, if the tick interrupt is required every 100 clock pulses, 99 must be written into the RELOAD. If a new value is written on each tick interrupt, so treated as single shot, then the actual count down must be written. For example, if a tick is next required after 400 clock pulses, 400 must be written into the RELOAD.

Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:0	RELOAD	W1C	-	Reload  Value to load into the SysTick Current Value Register when the counter reaches 0.

### ***SysTick Current Value Register***

Use the SysTick Current Value Register to find the current value in the register.

Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:0	CURRENT	W1C	-	Current Value  Current value at the time the register is accessed. No read-modify-write protection is provided, so change with care.  This register is write-clear. Writing to it with any value clears the register to 0. Clearing this register also clears the COUNTFLAG bit of the SysTick Control and Status Register.

### ***SysTick Calibration Value Register***

The SysTick Calibration Value register is not implemented.

### 3 Memory Map

The memory map for the LM3S1607 controller is provided in Table 3-1 on page 42.

In this manual, register addresses are given as a hexadecimal increment, relative to the module's base address as shown in the memory map. See also Chapter 4, "Memory Map" in the *ARM® Cortex™-M3 Technical Reference Manual*.

**Table 3-1. Memory Map<sup>a</sup>**

Start	End	Description	For details on registers, see page ...
<b>Memory</b>			
0x0000.0000	0x0001.FFFF	On-chip flash <sup>b</sup>	153
0x0002.0000	0x00FF.FFFF	Reserved	-
0x0100.0000	0x0100.2BFF	On-chip ROM	152
0x0100.2C00	0x1FFF.FFFF	Reserved	-
0x2000.0000	0x2000.7FFF	Bit-banded on-chip SRAM <sup>c</sup>	153
0x2000.8000	0x21FF.FFFF	Reserved	-
0x2200.0000	0x220F.FFFF	Bit-band alias of 0x2000.0000 through 0x200F.FFFF	147
0x2210.0000	0x3FFF.FFFF	Reserved	-
<b>FIRM Peripherals</b>			
0x4000.0000	0x4000.0FFF	Watchdog timer	318
0x4000.1000	0x4000.3FFF	Reserved	-
0x4000.4000	0x4000.4FFF	GPIO Port A	245
0x4000.5000	0x4000.5FFF	GPIO Port B	245
0x4000.6000	0x4000.6FFF	GPIO Port C	245
0x4000.7000	0x4000.7FFF	GPIO Port D	245
0x4000.8000	0x4000.8FFF	SSI0	426
0x4000.9000	0x4000.BFFF	Reserved	-
0x4000.C000	0x4000.CFFF	UART0	379
0x4000.D000	0x4000.DFFF	UART1	379
0x4000.E000	0x4000.EFFF	UART2	379
0x4000.F000	0x4001.FFFF	Reserved	-
<b>Peripherals</b>			
0x4002.0000	0x4002.07FF	I2C Master 0	466
0x4002.0800	0x4002.0FFF	I2C Slave 0	479
0x4002.1000	0x4002.17FF	I2C Master 1	466
0x4002.1800	0x4002.1FFF	I2C Slave 1	479
0x4002.2000	0x4002.3FFF	Reserved	-
0x4002.4000	0x4002.4FFF	GPIO Port E	245
0x4002.5000	0x4002.FFFF	Reserved	-
0x4003.0000	0x4003.0FFF	Timer0	293
0x4003.1000	0x4003.1FFF	Timer1	293
0x4003.2000	0x4003.2FFF	Timer2	293

Start	End	Description	For details on registers, see page ...
0x4003.3000	0x4003.3FFF	Timer3	293
0x4003.4000	0x4003.7FFF	Reserved	-
0x4003.8000	0x4003.8FFF	ADC	346
0x4003.9000	0x4005.7FFF	Reserved	-
0x4005.8000	0x4005.8FFF	GPIO Port A (AHB aperture)	245
0x4005.9000	0x4005.9FFF	GPIO Port B (AHB aperture)	245
0x4005.A000	0x4005.AFFF	GPIO Port C (AHB aperture)	245
0x4005.B000	0x4005.BFFF	GPIO Port D (AHB aperture)	245
0x4005.C000	0x4005.CFFF	GPIO Port E (AHB aperture)	245
0x4005.D000	0x400F.BFFF	Reserved	-
0x400F.C000	0x400F.CFFF	Hibernation Module	133
0x400F.D000	0x400F.DFFF	Flash control	153
0x400F.E000	0x400F.EFFF	System control	70
0x400F.F000	0x400F.FFFF	uDMA	196
0x4010.0000	0x41FF.FFFF	Reserved	-
0x4200.0000	0x43FF.FFFF	Bit-banded alias of 0x4000.0000 through 0x400F.FFFF	-
0x4400.0000	0xDFFF.FFFF	Reserved	-
<b>Private Peripheral Bus</b>			
0xE000.0000	0xE000.0FFF	Instrumentation Trace Macrocell (ITM)	ARM® Cortex™-M3 Technical Reference Manual
0xE000.1000	0xE000.1FFF	Data Watchpoint and Trace (DWT)	ARM® Cortex™-M3 Technical Reference Manual
0xE000.2000	0xE000.2FFF	Flash Patch and Breakpoint (FPB)	ARM® Cortex™-M3 Technical Reference Manual
0xE000.3000	0xE000.DFFF	Reserved	-
0xE000.E000	0xE000.EFFF	Nested Vectored Interrupt Controller (NVIC)	ARM® Cortex™-M3 Technical Reference Manual
0xE000.F000	0xE003.FFFF	Reserved	-
0xE004.0000	0xE004.0FFF	Trace Port Interface Unit (TPIU)	ARM® Cortex™-M3 Technical Reference Manual
0xE004.1000	0xFFFF.FFFF	Reserved	-

- a. All reserved space returns a bus fault when read or written.  
b. The unavailable flash will bus fault throughout this range.

c. The unavailable SRAM will bus fault throughout this range.

## 4 Interrupts

The ARM Cortex-M3 processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions. All exceptions are handled in Handler Mode. The processor state is automatically stored to the stack on an exception, and automatically restored from the stack at the end of the Interrupt Service Routine (ISR). The vector is fetched in parallel to the state saving, which enables efficient interrupt entry. The processor supports tail-chaining, which enables back-to-back interrupts to be performed without the overhead of state saving and restoration.

Table 4-1 on page 45 lists all exception types. Software can set eight priority levels on seven of these exceptions (system handlers) as well as on 27 interrupts (listed in Table 4-2 on page 46).

Priorities on the system handlers are set with the NVIC System Handler Priority registers. Interrupts are enabled through the NVIC Interrupt Set Enable register and prioritized with the NVIC Interrupt Priority registers. You also can group priorities by splitting priority levels into pre-emption priorities and subpriorities. All of the interrupt registers are described in Chapter 8, “Nested Vectored Interrupt Controller” in the *ARM® Cortex™-M3 Technical Reference Manual*.

Internally, the highest user-settable priority (0) is treated as fourth priority, after a Reset, NMI, and a Hard Fault. Note that 0 is the default priority for all the settable priorities.

If you assign the same priority level to two or more interrupts, their hardware priority (the lower position number) determines the order in which the processor activates them. For example, if both GPIO Port A and GPIO Port B are priority level 1, then GPIO Port A has higher priority.

See Chapter 5, “Exceptions” and Chapter 8, “Nested Vectored Interrupt Controller” in the *ARM® Cortex™-M3 Technical Reference Manual* for more information on exceptions and interrupts.

**Table 4-1. Exception Types**

Exception Type	Vector Number	Priority <sup>a</sup>	Description
-	0	-	Stack top is loaded from first entry of vector table on reset.
Reset	1	-3 (highest)	Invoked on power up and warm reset. On first instruction, drops to lowest priority (and then is called the base level of activation). This is asynchronous.
Non-Maskable Interrupt (NMI)	2	-2	Cannot be stopped or preempted by any exception but reset. This is asynchronous.
Hard Fault	3	-1	All classes of Fault, when the fault cannot activate due to priority or the configurable fault handler has been disabled. This is synchronous.
Memory Management	4	settable	MPU mismatch, including access violation and no match. This is synchronous.  The priority of this exception can be changed.
Bus Fault	5	settable	Pre-fetch fault, memory access fault, and other address/memory related faults. This is synchronous when precise and asynchronous when imprecise.  You can enable or disable this fault.
Usage Fault	6	settable	Usage fault, such as undefined instruction executed or illegal state transition attempt. This is synchronous.
-	7-10	-	Reserved.
SVCcall	11	settable	System service call with SVC instruction. This is synchronous.
Debug Monitor	12	settable	Debug monitor (when not halting). This is synchronous, but only active when enabled. It does not activate if lower priority than the current activation.

Exception Type	Vector Number	Priority <sup>a</sup>	Description
-	13	-	Reserved.
PendSV	14	settable	Pendable request for system service. This is asynchronous and only pended by software.
SysTick	15	settable	System tick timer has fired. This is asynchronous.
Interrupts	16 and above	settable	Asserted from outside the ARM Cortex-M3 core and fed through the NVIC (prioritized). These are all asynchronous. Table 4-2 on page 46 lists the interrupts on the LM3S1607 controller.

a. 0 is the default priority for all the settable priorities.

**Table 4-2. Interrupts**

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Description
0-15	-	Processor exceptions
16	0	GPIO Port A
17	1	GPIO Port B
18	2	GPIO Port C
19	3	GPIO Port D
20	4	GPIO Port E
21	5	UART0
22	6	UART1
23	7	SSI0
24	8	I2C0
25-29	9-13	Reserved
30	14	ADC Sequence 0
31	15	ADC Sequence 1
32	16	ADC Sequence 2
33	17	ADC Sequence 3
34	18	Watchdog timer
35	19	Timer0 A
36	20	Timer0 B
37	21	Timer1 A
38	22	Timer1 B
39	23	Timer2 A
40	24	Timer2 B
41-43	25-27	Reserved
44	28	System Control
45	29	Flash Control
46-48	30-32	Reserved
49	33	UART2
50	34	Reserved
51	35	Timer3 A
52	36	Timer3 B
53	37	I2C1
54-58	38-42	Reserved

---

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Description
59	43	Hibernation Module
60-61	44-45	Reserved
62	46	uDMA Software
63	47	uDMA Error

## 5 JTAG Interface

The Joint Test Action Group (JTAG) port is an IEEE standard that defines a Test Access Port and Boundary Scan Architecture for digital integrated circuits and provides a standardized serial interface for controlling the associated test logic. The TAP, Instruction Register (IR), and Data Registers (DR) can be used to test the interconnections of assembled printed circuit boards and obtain manufacturing information on the components. The JTAG Port also provides a means of accessing and controlling design-for-test features such as I/O pin observation and control, scan testing, and debugging.

The JTAG port is comprised of four pins: TCK, TMS, TDI, and TDO. Data is transmitted serially into the controller on TDI and out of the controller on TDO. The interpretation of this data is dependent on the current state of the TAP controller. For detailed information on the operation of the JTAG port and TAP controller, please refer to the *IEEE Standard 1149.1-Test Access Port and Boundary-Scan Architecture*.

The Luminary Micro JTAG controller works with the ARM JTAG controller built into the Cortex-M3 core. This is implemented by multiplexing the TDO outputs from both JTAG controllers. ARM JTAG instructions select the ARM TDO output while Luminary Micro JTAG instructions select the Luminary Micro TDO outputs. The multiplexer is controlled by the Luminary Micro JTAG controller, which has comprehensive programming for the ARM, Luminary Micro, and unimplemented JTAG instructions.

The JTAG module has the following features:

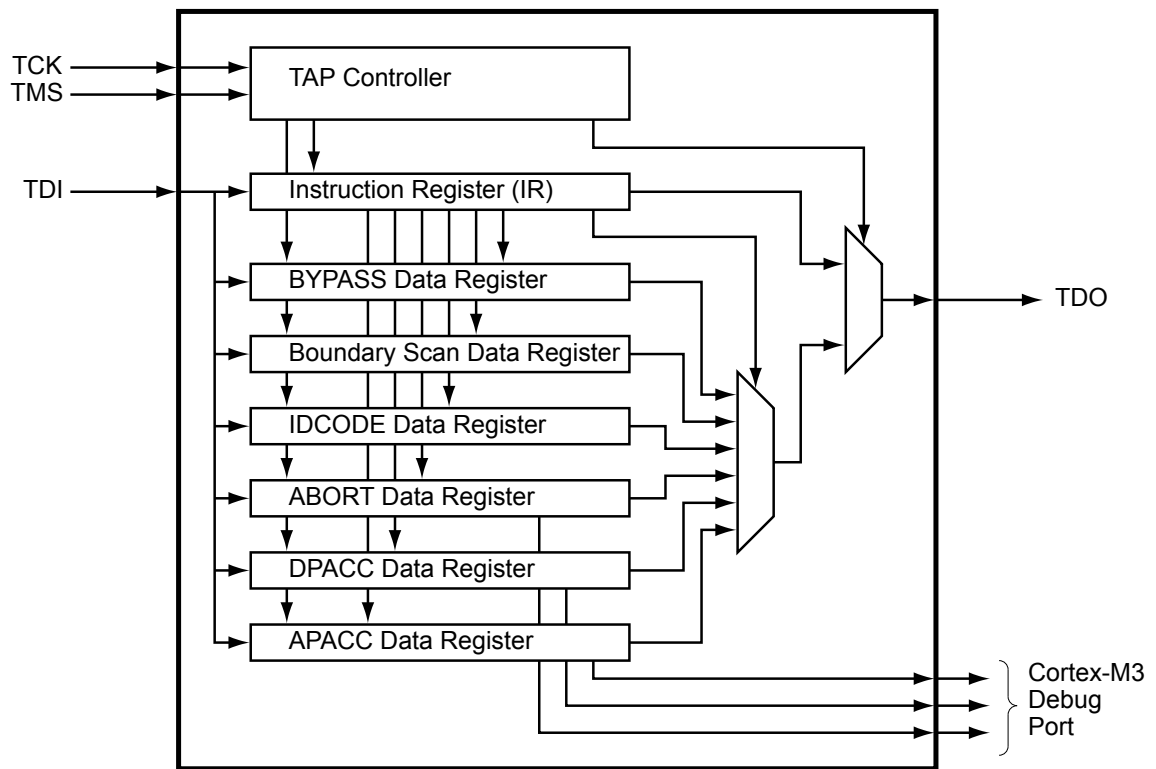
- IEEE 1149.1-1990 compatible Test Access Port (TAP) controller
- Four-bit Instruction Register (IR) chain for storing JTAG instructions
- IEEE standard instructions:
  - BYPASS instruction
  - IDCODE instruction
  - SAMPLE/PRELOAD instruction
  - EXTEST instruction
  - INTEST instruction
- ARM additional instructions:
  - APACC instruction
  - DPACC instruction
  - ABORT instruction
- Integrated ARM Serial Wire Debug (SWD)

See the *ARM® Cortex™-M3 Technical Reference Manual* for more information on the ARM JTAG controller.



## 5.1 Block Diagram

Figure 5-1. JTAG Module Block Diagram



## 5.2 Functional Description

A high-level conceptual drawing of the JTAG module is shown in Figure 5-1 on page 49. The JTAG module is composed of the Test Access Port (TAP) controller and serial shift chains with parallel update registers. The TAP controller is a simple state machine controlled by the TCK and TMS inputs. The current state of the TAP controller depends on the sequence of values captured on TMS at the rising edge of TCK. The TAP controller determines when the serial shift chains capture new data, shift data from TDI towards TDO, and update the parallel load registers. The current state of the TAP controller also determines whether the Instruction Register (IR) chain or one of the Data Register (DR) chains is being accessed.

The serial shift chains with parallel load registers are comprised of a single Instruction Register (IR) chain and multiple Data Register (DR) chains. The current instruction loaded in the parallel load register determines which DR chain is captured, shifted, or updated during the sequencing of the TAP controller.

Some instructions, like EXTEST and INTEST, operate on data currently in a DR chain and do not capture, shift, or update any of the chains. Instructions that are not implemented decode to the BYPASS instruction to ensure that the serial path between TDI and TDO is always connected (see Table 5-2 on page 55 for a list of implemented instructions).

See “JTAG and Boundary Scan” on page 508 for JTAG timing diagrams.

## 5.2.1 JTAG Interface Pins

The JTAG interface consists of four standard pins: TCK, TMS, TDI, and TDO. These pins and their associated reset state are given in Table 5-1 on page 50. Detailed information on each pin follows.

**Table 5-1. JTAG Port Pins Reset State**

Pin Name	Data Direction	Internal Pull-Up	Internal Pull-Down	Drive Strength	Drive Value
TCK	Input	Enabled	Disabled	N/A	N/A
TMS	Input	Enabled	Disabled	N/A	N/A
TDI	Input	Enabled	Disabled	N/A	N/A
TDO	Output	Enabled	Disabled	2-mA driver	High-Z

### 5.2.1.1 Test Clock Input (TCK)

The TCK pin is the clock for the JTAG module. This clock is provided so the test logic can operate independently of any other system clocks. In addition, it ensures that multiple JTAG TAP controllers that are daisy-chained together can synchronously communicate serial test data between components. During normal operation, TCK is driven by a free-running clock with a nominal 50% duty cycle. When necessary, TCK can be stopped at 0 or 1 for extended periods of time. While TCK is stopped at 0 or 1, the state of the TAP controller does not change and data in the JTAG Instruction and Data Registers is not lost.

By default, the internal pull-up resistor on the TCK pin is enabled after reset. This assures that no clocking occurs if the pin is not driven from an external source. The internal pull-up and pull-down resistors can be turned off to save internal power as long as the TCK pin is constantly being driven by an external source.

### 5.2.1.2 Test Mode Select (TMS)

The TMS pin selects the next state of the JTAG TAP controller. TMS is sampled on the rising edge of TCK. Depending on the current TAP state and the sampled value of TMS, the next state is entered. Because the TMS pin is sampled on the rising edge of TCK, the *IEEE Standard 1149.1* expects the value on TMS to change on the falling edge of TCK.

Holding TMS high for five consecutive TCK cycles drives the TAP controller state machine to the Test-Logic-Reset state. When the TAP controller enters the Test-Logic-Reset state, the JTAG module and associated registers are reset to their default values. This procedure should be performed to initialize the JTAG controller. The JTAG Test Access Port state machine can be seen in its entirety in Figure 5-2 on page 52.

By default, the internal pull-up resistor on the TMS pin is enabled after reset. Changes to the pull-up resistor settings on GPIO Port C should ensure that the internal pull-up resistor remains enabled on PC1/TMS; otherwise JTAG communication could be lost.

### 5.2.1.3 Test Data Input (TDI)

The TDI pin provides a stream of serial information to the IR chain and the DR chains. TDI is sampled on the rising edge of TCK and, depending on the current TAP state and the current instruction, presents this data to the proper shift register chain. Because the TDI pin is sampled on the rising edge of TCK, the *IEEE Standard 1149.1* expects the value on TDI to change on the falling edge of TCK.

By default, the internal pull-up resistor on the TDI pin is enabled after reset. Changes to the pull-up resistor settings on GPIO Port C should ensure that the internal pull-up resistor remains enabled on PC2/TDI; otherwise JTAG communication could be lost.

#### 5.2.1.4 Test Data Output (TDO)

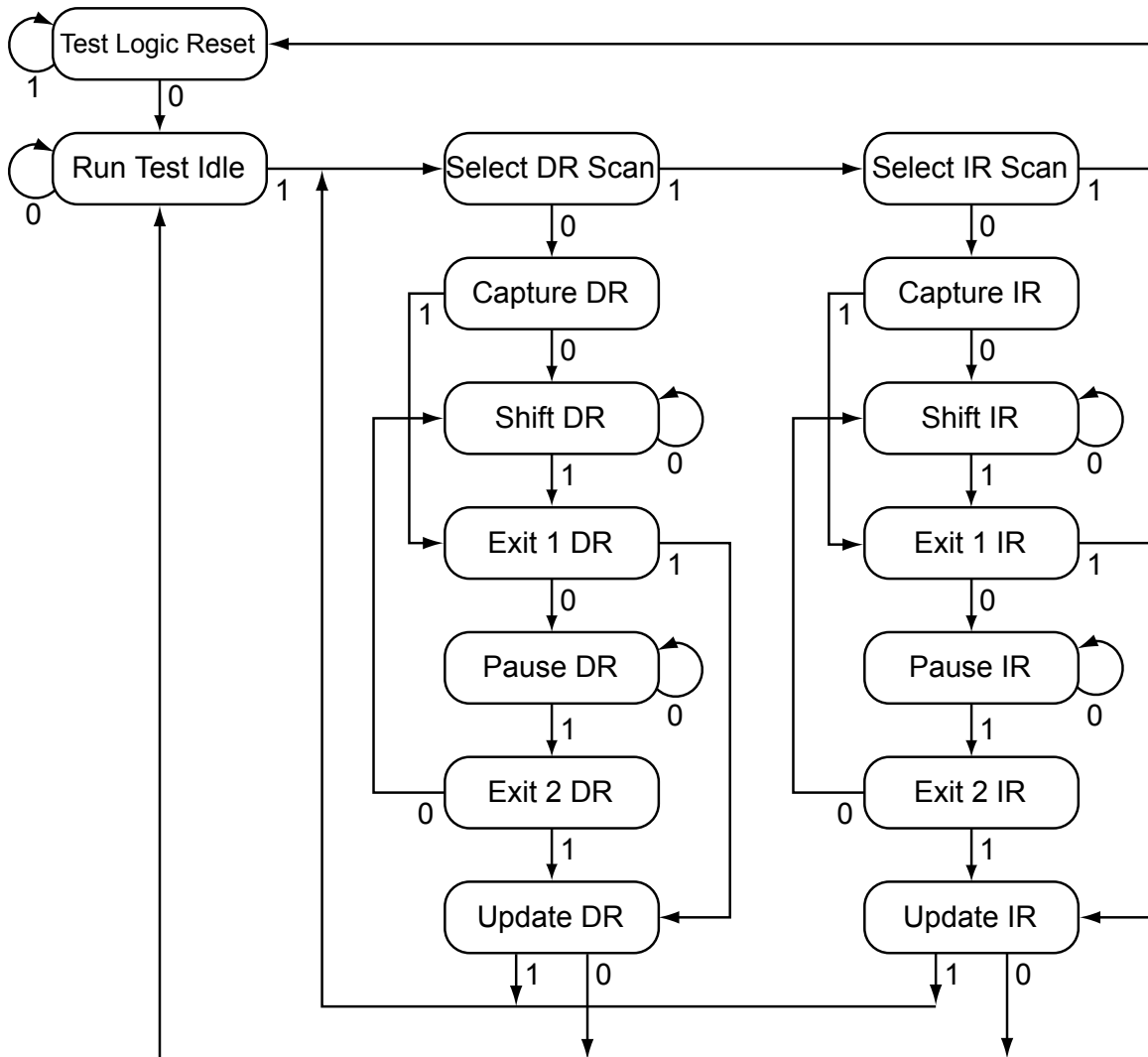
The TDO pin provides an output stream of serial information from the IR chain or the DR chains. The value of TDO depends on the current TAP state, the current instruction, and the data in the chain being accessed. In order to save power when the JTAG port is not being used, the TDO pin is placed in an inactive drive state when not actively shifting out data. Because TDO can be connected to the TDI of another controller in a daisy-chain configuration, the *IEEE Standard 1149.1* expects the value on TDO to change on the falling edge of TCK.

By default, the internal pull-up resistor on the TDO pin is enabled after reset. This assures that the pin remains at a constant logic level when the JTAG port is not being used. The internal pull-up and pull-down resistors can be turned off to save internal power if a High-Z output value is acceptable during certain TAP controller states.

#### 5.2.2 JTAG TAP Controller

The JTAG TAP controller state machine is shown in Figure 5-2 on page 52. The TAP controller state machine is reset to the Test-Logic-Reset state on the assertion of a Power-On-Reset (POR). Asserting the correct sequence on the TMS pin allows the JTAG module to shift in new instructions, shift in data, or idle during extended testing sequences. For detailed information on the function of the TAP controller and the operations that occur in each state, please refer to *IEEE Standard 1149.1*.

Figure 5-2. Test Access Port State Machine



### 5.2.3 Shift Registers

The Shift Registers consist of a serial shift register chain and a parallel load register. The serial shift register chain samples specific information during the TAP controller's CAPTURE states and allows this information to be shifted out of TDO during the TAP controller's SHIFT states. While the sampled data is being shifted out of the chain on TDO, new data is being shifted into the serial shift register on TDI. This new data is stored in the parallel load register during the TAP controller's UPDATE states. Each of the shift registers is discussed in detail in "Register Descriptions" on page 55.

### 5.2.4 Operational Considerations

There are certain operational considerations when using the JTAG module. Because the JTAG pins can be programmed to be GPIOs, board configuration and reset conditions on these pins must be considered. In addition, because the JTAG module has integrated ARM Serial Wire Debug, the method for switching between these two operational modes is described below.

### 5.2.4.1 GPIO Functionality

When the controller is reset with either a POR or  $\overline{\text{RST}}$ , the JTAG/SWD port pins default to their JTAG/SWD configurations. The default configuration includes enabling digital functionality (setting **GPIO DEN** to 1), enabling the pull-up resistors (setting **GPIO PUR** to 1), and enabling the alternate hardware function (setting **GPIO AFSEL** to 1) for the PC[3:0] JTAG/SWD pins.

It is possible for software to configure these pins as GPIOs after reset by writing 0s to PC[3:0] in the **GPIO AFSEL** register. If the user does not require the JTAG/SWD port for debugging or board-level testing, this provides four more GPIOs for use in the design.

---

**Caution – It is possible to create a software sequence that prevents the debugger from connecting to the Stellaris® microcontroller. If the program code loaded into flash immediately changes the JTAG pins to their GPIO functionality, the debugger may not have enough time to connect and halt the controller before the JTAG pin functionality switches. This may lock the debugger out of the part. This can be avoided with a software routine that restores JTAG functionality based on an external or software trigger.**

---

The commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Writes to protected bits of the **GPIO Alternate Function Select (GPIO AFSEL)** register (see page 255), **GPIO Pull-Up Select (GPIO PUR)** register (see page 261), and **GPIO Digital Enable (GPIO DEN)** register (see page 264) are not committed to storage unless the **GPIO Lock (GPIO LOCK)** register (see page 266) has been unlocked and the appropriate bits of the **GPIO Commit (GPIO CR)** register (see page 267) have been set to 1.

#### **Recovering a "Locked" Device**

**Note:** Performing the below sequence will cause the nonvolatile registers discussed in “Nonvolatile Register Programming” on page 150 to be restored to their factory default values. The mass erase of the flash memory caused by the below sequence occurs prior to the nonvolatile registers being restored.

If software configures any of the JTAG/SWD pins as GPIO and loses the ability to communicate with the debugger, there is a debug sequence that can be used to recover the device. Performing a total of ten JTAG-to-SWD and SWD-to-JTAG switch sequences while holding the device in reset mass erases the flash memory. The sequence to recover the device is:

1. Assert and hold the  $\overline{\text{RST}}$  signal.
2. Perform the JTAG-to-SWD switch sequence.
3. Perform the SWD-to-JTAG switch sequence.
4. Perform the JTAG-to-SWD switch sequence.
5. Perform the SWD-to-JTAG switch sequence.
6. Perform the JTAG-to-SWD switch sequence.
7. Perform the SWD-to-JTAG switch sequence.
8. Perform the JTAG-to-SWD switch sequence.
9. Perform the SWD-to-JTAG switch sequence.
10. Perform the JTAG-to-SWD switch sequence.

11. Perform the SWD-to-JTAG switch sequence.
12. Release the  $\overline{\text{RST}}$  signal.
13. Wait 400 ms.
14. Power-cycle the device.

The JTAG-to-SWD and SWD-to-JTAG switch sequences are described in “ARM Serial Wire Debug (SWD)” on page 54. When performing switch sequences for the purpose of recovering the debug capabilities of the device, only steps 1 and 2 of the switch sequence need to be performed.

#### 5.2.4.2 ARM Serial Wire Debug (SWD)

In order to seamlessly integrate the ARM Serial Wire Debug (SWD) functionality, a serial-wire debugger must be able to connect to the Cortex-M3 core without having to perform, or have any knowledge of, JTAG cycles. This is accomplished with a SWD preamble that is issued before the SWD session begins.

The preamble used to enable the SWD interface of the SWJ-DP module starts with the TAP controller in the Test-Logic-Reset state. From here, the preamble sequences the TAP controller through the following states: Run Test Idle, Select DR, Select IR, Test Logic Reset, Test Logic Reset, Run Test Idle, Run Test Idle, Select DR, Select IR, Test Logic Reset, Test Logic Reset, Run Test Idle, Run Test Idle, Select DR, Select IR, and Test Logic Reset states.

Stepping through this sequences of the TAP state machine enables the SWD interface and disables the JTAG interface. For more information on this operation and the SWD interface, see the *ARM® Cortex™-M3 Technical Reference Manual* and the *ARM® CoreSight Technical Reference Manual*.

Because this sequence is a valid series of JTAG operations that could be issued, the ARM JTAG TAP controller is not fully compliant to the *IEEE Standard 1149.1*. This is the only instance where the ARM JTAG TAP controller does not meet full compliance with the specification. Due to the low probability of this sequence occurring during normal operation of the TAP controller, it should not affect normal performance of the JTAG interface.

##### **JTAG-to-SWD Switching**

To switch the operating mode of the Debug Access Port (DAP) from JTAG to SWD mode, the external debug hardware must send a switch sequence to the device. The 16-bit switch sequence for switching to SWD mode is defined as b1110011110011110, transmitted LSB first. This can also be represented as 16'hE79E when transmitted LSB first. The complete switch sequence should consist of the following transactions on the TCK/SWCLK and TMS/SWDIO signals:

1. Send at least 50 TCK/SWCLK cycles with TMS/SWDIO set to 1. This ensures that both JTAG and SWD are in their reset/idle states.
2. Send the 16-bit JTAG-to-SWD switch sequence, 16'hE79E.
3. Send at least 50 TCK/SWCLK cycles with TMS/SWDIO set to 1. This ensures that if SWJ-DP was already in SWD mode, before sending the switch sequence, the SWD goes into the line reset state.

##### **SWD-to-JTAG Switching**

To switch the operating mode of the Debug Access Port (DAP) from SWD to JTAG mode, the external debug hardware must send a switch sequence to the device. The 16-bit switch sequence for switching to JTAG mode is defined as b1110011110011110, transmitted LSB first. This can also

be represented as 16'hE73C when transmitted LSB first. The complete switch sequence should consist of the following transactions on the TCK/SWCLK and TMS/SWDIO signals:

1. Send at least 50 TCK/SWCLK cycles with TMS/SWDIO set to 1. This ensures that both JTAG and SWD are in their reset/idle states.
2. Send the 16-bit SWD-to-JTAG switch sequence, 16'hE73C.
3. Send at least 5 TCK/SWCLK cycles with TMS/SWDIO set to 1. This ensures that if SWJ-DP was already in JTAG mode, before sending the switch sequence, the JTAG goes into the Test Logic Reset state.

## 5.3 Initialization and Configuration

After a Power-On-Reset or an external reset ( $\overline{RST}$ ), the JTAG pins are automatically configured for JTAG communication. No user-defined initialization or configuration is needed. However, if the user application changes these pins to their GPIO function, they must be configured back to their JTAG functionality before JTAG communication can be restored. This is done by enabling the four JTAG pins ( $PC[3:0]$ ) for their alternate function using the **GPIOAFSEL** register.

## 5.4 Register Descriptions

There are no APB-accessible registers in the JTAG TAP Controller or Shift Register chains. The registers within the JTAG controller are all accessed serially through the TAP Controller. The registers can be broken down into two main categories: Instruction Registers and Data Registers.

### 5.4.1 Instruction Register (IR)

The JTAG TAP Instruction Register (IR) is a four-bit serial scan chain with a parallel load register connected between the JTAG TDI and TDO pins. When the TAP Controller is placed in the correct states, bits can be shifted into the Instruction Register. Once these bits have been shifted into the chain and updated, they are interpreted as the current instruction. The decode of the Instruction Register bits is shown in Table 5-2 on page 55. A detailed explanation of each instruction, along with its associated Data Register, follows.

**Table 5-2. JTAG Instruction Register Commands**

IR[3:0]	Instruction	Description
0000	EXTEST	Drives the values preloaded into the Boundary Scan Chain by the SAMPLE/PRELOAD instruction onto the pads.
0001	INTEST	Drives the values preloaded into the Boundary Scan Chain by the SAMPLE/PRELOAD instruction into the controller.
0010	SAMPLE / PRELOAD	Captures the current I/O values and shifts the sampled values out of the Boundary Scan Chain while new preload data is shifted in.
1000	ABORT	Shifts data into the ARM Debug Port Abort Register.
1010	DPACC	Shifts data into and out of the ARM DP Access Register.
1011	APACC	Shifts data into and out of the ARM AC Access Register.
1110	IDCODE	Loads manufacturing information defined by the <i>IEEE Standard 1149.1</i> into the IDCODE chain and shifts it out.
1111	BYPASS	Connects TDI to TDO through a single Shift Register chain.
All Others	Reserved	Defaults to the BYPASS instruction to ensure that TDI is always connected to TDO.

#### 5.4.1.1 EXTEST Instruction

The EXTEST instruction does not have an associated Data Register chain. The EXTEST instruction uses the data that has been preloaded into the Boundary Scan Data Register using the SAMPLE/PRELOAD instruction. When the EXTEST instruction is present in the Instruction Register, the preloaded data in the Boundary Scan Data Register associated with the outputs and output enables are used to drive the GPIO pads rather than the signals coming from the core. This allows tests to be developed that drive known values out of the controller, which can be used to verify connectivity.

#### 5.4.1.2 INTEST Instruction

The INTEST instruction does not have an associated Data Register chain. The INTEST instruction uses the data that has been preloaded into the Boundary Scan Data Register using the SAMPLE/PRELOAD instruction. When the INTEST instruction is present in the Instruction Register, the preloaded data in the Boundary Scan Data Register associated with the inputs are used to drive the signals going into the core rather than the signals coming from the GPIO pads. This allows tests to be developed that drive known values into the controller, which can be used for testing.

#### 5.4.1.3 SAMPLE/PRELOAD Instruction

The SAMPLE/PRELOAD instruction connects the Boundary Scan Data Register chain between TDI and TDO. This instruction samples the current state of the pad pins for observation and preloads new test data. Each GPIO pad has an associated input, output, and output enable signal. When the TAP controller enters the Capture DR state during this instruction, the input, output, and output-enable signals to each of the GPIO pads are captured. These samples are serially shifted out of TDO while the TAP controller is in the Shift DR state and can be used for observation or comparison in various tests.

While these samples of the inputs, outputs, and output enables are being shifted out of the Boundary Scan Data Register, new data is being shifted into the Boundary Scan Data Register from TDI. Once the new data has been shifted into the Boundary Scan Data Register, the data is saved in the parallel load registers when the TAP controller enters the Update DR state. This update of the parallel load register preloads data into the Boundary Scan Data Register that is associated with each input, output, and output enable. This preloaded data can be used with the EXTEST and INTEST instructions to drive data into or out of the controller. Please see “Boundary Scan Data Register” on page 58 for more information.

#### 5.4.1.4 ABORT Instruction

The ABORT instruction connects the associated ABORT Data Register chain between TDI and TDO. This instruction provides read and write access to the ABORT Register of the ARM Debug Access Port (DAP). Shifting the proper data into this Data Register clears various error bits or initiates a DAP abort of a previous request. Please see the “ABORT Data Register” on page 59 for more information.

#### 5.4.1.5 DPACC Instruction

The DPACC instruction connects the associated DPACC Data Register chain between TDI and TDO. This instruction provides read and write access to the DPACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to the ARM debug and status registers. Please see “DPACC Data Register” on page 58 for more information.



### 5.4.1.6 APACC Instruction

The APACC instruction connects the associated APACC Data Register chain between TDI and TDO. This instruction provides read and write access to the APACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to internal components and buses through the Debug Port. Please see “APACC Data Register” on page 58 for more information.

### 5.4.1.7 IDCODE Instruction

The IDCODE instruction connects the associated IDCODE Data Register chain between TDI and TDO. This instruction provides information on the manufacturer, part number, and version of the ARM core. This information can be used by testing equipment and debuggers to automatically configure their input and output data streams. IDCODE is the default instruction that is loaded into the JTAG Instruction Register when a power-on-reset (POR) is asserted, or the Test-Logic-Reset state is entered. Please see “IDCODE Data Register” on page 57 for more information.

### 5.4.1.8 BYPASS Instruction

The BYPASS instruction connects the associated BYPASS Data Register chain between TDI and TDO. This instruction is used to create a minimum length serial path between the TDI and TDO ports. The BYPASS Data Register is a single-bit shift register. This instruction improves test efficiency by allowing components that are not needed for a specific test to be bypassed in the JTAG scan chain by loading them with the BYPASS instruction. Please see “BYPASS Data Register” on page 58 for more information.

## 5.4.2 Data Registers

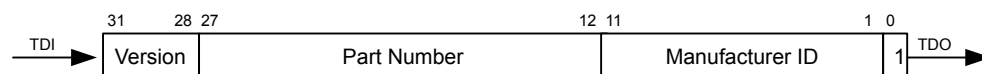
The JTAG module contains six Data Registers. These include: IDCODE, BYPASS, Boundary Scan, APACC, DPACC, and ABORT serial Data Register chains. Each of these Data Registers is discussed in the following sections.

### 5.4.2.1 IDCODE Data Register

The format for the 32-bit IDCODE Data Register defined by the *IEEE Standard 1149.1* is shown in Figure 5-3 on page 57. The standard requires that every JTAG-compliant device implement either the IDCODE instruction or the BYPASS instruction as the default instruction. The LSB of the IDCODE Data Register is defined to be a 1 to distinguish it from the BYPASS instruction, which has an LSB of 0. This allows auto configuration test tools to determine which instruction is the default instruction.

The major uses of the JTAG port are for manufacturer testing of component assembly, and program development and debug. To facilitate the use of auto-configuration debug tools, the IDCODE instruction outputs a value of 0x3BA00477. This value indicates an ARM Cortex-M3, Version 1 processor. This allows the debuggers to automatically configure themselves to work correctly with the Cortex-M3 during debug.

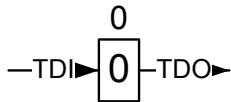
**Figure 5-3. IDCODE Register Format**



### 5.4.2.2 BYPASS Data Register

The format for the 1-bit BYPASS Data Register defined by the *IEEE Standard 1149.1* is shown in Figure 5-4 on page 58. The standard requires that every JTAG-compliant device implement either the BYPASS instruction or the IDCODE instruction as the default instruction. The LSB of the BYPASS Data Register is defined to be a 0 to distinguish it from the IDCODE instruction, which has an LSB of 1. This allows auto configuration test tools to determine which instruction is the default instruction.

Figure 5-4. BYPASS Register Format

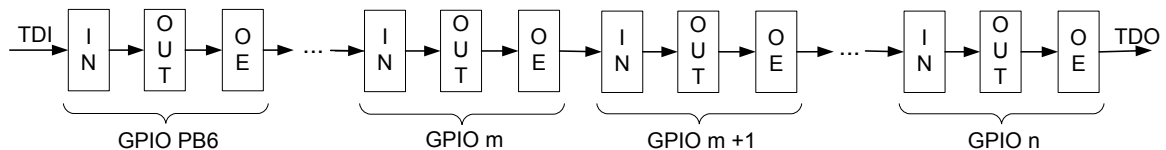


### 5.4.2.3 Boundary Scan Data Register

The format of the Boundary Scan Data Register is shown in Figure 5-5 on page 58. Each GPIO pin, in a counter-clockwise direction from the JTAG port pins, is included in the Boundary Scan Data Register. Each GPIO pin has three associated digital signals that are included in the chain. These signals are input, output, and output enable, and are arranged in that order as can be seen in the figure. In addition to the GPIO pins, the controller reset pin,  $\overline{RST}$ , is included in the chain. Because the reset pin is always an input, only the input signal is included in the Data Register chain.

When the Boundary Scan Data Register is accessed with the SAMPLE/PRELOAD instruction, the input, output, and output enable from each digital pad are sampled and then shifted out of the chain to be verified. The sampling of these values occurs on the rising edge of TCK in the Capture DR state of the TAP controller. While the sampled data is being shifted out of the Boundary Scan chain in the Shift DR state of the TAP controller, new data can be preloaded into the chain for use with the EXTEST and INTEST instructions. These instructions either force data out of the controller, with the EXTEST instruction, or into the controller, with the INTEST instruction.

Figure 5-5. Boundary Scan Register Format



For detailed information on the order of the input, output, and output enable bits for each of the GPIO ports, please refer to the Stellaris® Family Boundary Scan Description Language (BSDL) files, downloadable from [www.luminarymicro.com](http://www.luminarymicro.com).

### 5.4.2.4 APACC Data Register

The format for the 35-bit APACC Data Register defined by ARM is described in the *ARM® Cortex™-M3 Technical Reference Manual*.

### 5.4.2.5 DPACC Data Register

The format for the 35-bit DPACC Data Register defined by ARM is described in the *ARM® Cortex™-M3 Technical Reference Manual*.

#### 5.4.2.6 ABORT Data Register

The format for the 35-bit ABORT Data Register defined by ARM is described in the *ARM® Cortex™-M3 Technical Reference Manual*.

## 6 System Control

System control determines the overall operation of the device. It provides information about the device, controls the clocking to the core and individual peripherals, and handles reset detection and reporting.

### 6.1 Functional Description

The System Control module provides the following capabilities:

- Device identification, see “Device Identification” on page 60
- Local control, such as reset (see “Reset Control” on page 60), power (see “Power Control” on page 63) and clock control (see “Clock Control” on page 63)
- System control (Run, Sleep, and Deep-Sleep modes), see “System Control” on page 67

#### 6.1.1 Device Identification

Seven read-only registers provide software with information on the microcontroller, such as version, part number, SRAM size, flash size, and other features. See the **DID0**, **DID1**, and **DC0-DC7** registers.

#### 6.1.2 Reset Control

This section discusses aspects of hardware functions during reset as well as system software requirements following the reset sequence.

##### 6.1.2.1 Reset Sources

The controller has six sources of reset:

1. External reset input pin ( $\overline{\text{RST}}$ ) assertion, see “ $\overline{\text{RST}}$  Pin Assertion” on page 60.
2. Power-on reset (POR), see “Power-On Reset (POR)” on page 61.
3. Internal brown-out (BOR) detector, see “Brown-Out Reset (BOR)” on page 61.
4. Software-initiated reset (with the software reset registers), see “Software Reset” on page 62.
5. A watchdog timer reset condition violation, see “Watchdog Timer Reset” on page 62.
6. MOSC failure

After a reset, the **Reset Cause (RESC)** register is set with the reset cause. The bits in this register are sticky and maintain their state across multiple reset sequences, except when an internal POR is the cause, and then all the other bits in the **RESC** register are cleared except for the POR indicator.

##### 6.1.2.2 $\overline{\text{RST}}$ Pin Assertion

The external reset pin ( $\overline{\text{RST}}$ ) resets the controller. This resets the core and all the peripherals except the JTAG TAP controller (see “JTAG Interface” on page 48). The external reset sequence is as follows:

1. The external reset pin ( $\overline{\text{RST}}$ ) is asserted and then de-asserted.

- The internal reset is released and the core loads from memory the initial stack pointer, the initial program counter, the first instruction designated by the program counter, and begins execution. A few clocks cycles from  $\overline{\text{RST}}$  de-assertion to the start of the reset sequence is necessary for synchronization.

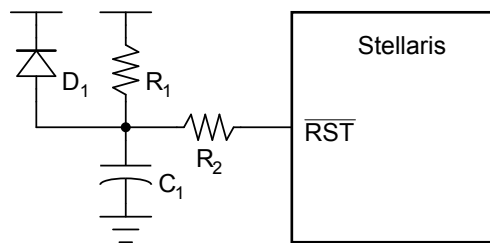
The external reset timing is shown in Figure 20-9 on page 510.

### 6.1.2.3 Power-On Reset (POR)

The Power-On Reset (POR) circuit monitors the power supply voltage ( $V_{DD}$ ). The POR circuit generates a reset signal to the internal logic when the power supply ramp reaches a threshold value ( $V_{TH}$ ). If the application only uses the POR circuit, the  $\overline{\text{RST}}$  input needs to be connected to the power supply ( $V_{DD}$ ) through a pull-up resistor (1K to 10K  $\Omega$ ).

The device must be operating within the specified operating parameters at the point when the on-chip power-on reset pulse is complete. The 3.3-V power supply to the device must reach 3.0 V within 10 msec of it crossing 2.0 V to guarantee proper operation. For applications that require the use of an external reset to hold the device in reset longer than the internal POR, the  $\overline{\text{RST}}$  input may be used with the circuit as shown in Figure 6-1 on page 61.

**Figure 6-1. External Circuitry to Extend Reset**



The  $R_1$  and  $C_1$  components define the power-on delay. The  $R_2$  resistor mitigates any leakage from the  $\overline{\text{RST}}$  input. The diode ( $D_1$ ) discharges  $C_1$  rapidly when the power supply is turned off.

The Power-On Reset sequence is as follows:

- The controller waits for the later of external reset ( $\overline{\text{RST}}$ ) or internal POR to go inactive.
- The internal reset is released and the core loads from memory the initial stack pointer, the initial program counter, the first instruction designated by the program counter, and begins execution.

The internal POR is only active on the initial power-up of the controller. The Power-On Reset timing is shown in Figure 20-10 on page 510.

**Note:** The power-on reset also resets the JTAG controller. An external reset does not.

### 6.1.2.4 Brown-Out Reset (BOR)

A drop in the input voltage resulting in the assertion of the internal brown-out detector can be used to reset the controller. This is initially disabled and may be enabled by software.

The system provides a brown-out detection circuit that triggers if the power supply ( $V_{DD}$ ) drops below a brown-out threshold voltage ( $V_{BTH}$ ). If a brown-out condition is detected, the system may generate a controller interrupt or a system reset.

Brown-out resets are controlled with the **Power-On and Brown-Out Reset Control (PBORCTL)** register. The `BORIOR` bit in the **PBORCTL** register must be set for a brown-out condition to trigger a reset.

The brown-out reset is equivalent to an assertion of the external  $\overline{RST}$  input and the reset is held active until the proper  $V_{DD}$  level is restored. The **RESC** register can be examined in the reset interrupt handler to determine if a Brown-Out condition was the cause of the reset, thus allowing software to determine what actions are required to recover.

The internal Brown-Out Reset timing is shown in Figure 20-11 on page 511.

### 6.1.2.5 Software Reset

Software can reset a specific peripheral or generate a reset to the entire system .

Peripherals can be individually reset by software via three registers that control reset signals to each peripheral (see the **SRCRn** registers). If the bit position corresponding to a peripheral is set and subsequently cleared, the peripheral is reset. The encoding of the reset registers is consistent with the encoding of the clock gating control for peripherals and on-chip functions (see “System Control” on page 67). Note that all reset signals for all clocks of the specified unit are asserted as a result of a software-initiated reset.

The entire system can be reset by software by setting the `SYSRESETREQ` bit in the Cortex-M3 Application Interrupt and Reset Control register resets the entire system including the core. The software-initiated system reset sequence is as follows:

1. A software system reset is initiated by writing the `SYSRESETREQ` bit in the ARM Cortex-M3 Application Interrupt and Reset Control register.
2. An internal reset is asserted.
3. The internal reset is deasserted and the controller loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

The software-initiated system reset timing is shown in Figure 20-12 on page 511.

### 6.1.2.6 Watchdog Timer Reset

The watchdog timer module's function is to prevent system hangs. The watchdog timer can be configured to generate an interrupt to the controller on its first time-out, and to generate a reset signal on its second time-out.

After the first time-out event, the 32-bit counter is reloaded with the value of the **Watchdog Timer Load (WDTLOAD)** register, and the timer resumes counting down from that value. If the timer counts down to its zero state again before the first time-out interrupt is cleared, and the reset signal has been enabled, the watchdog timer asserts its reset signal to the system. The watchdog timer reset sequence is as follows:

1. The watchdog timer times out for the second time without being serviced.
2. An internal reset is asserted.
3. The internal reset is released and the controller loads from memory the initial stack pointer, the initial program counter, the first instruction designated by the program counter, and begins execution.

The watchdog reset timing is shown in Figure 20-13 on page 511.

### 6.1.3 Non-Maskable Interrupt

The controller has two sources of non-maskable interrupt (NMI):

- The assertion of the `NMI` signal.
- A main oscillator verification error.

If both sources of NMI are enabled, software must check that the main oscillator verification is the cause of the interrupt in order to distinguish between the two sources.

#### 6.1.3.1 NMI Pin

The alternate function to GPIO port pin B7 is an NMI signal. The alternate function must be enabled in the GPIO for the signal to be used as an interrupt, as described in “General-Purpose Input/Outputs (GPIOs)” on page 237. Note that enabling the NMI alternate function requires the use of the GPIO lock and commit function just like the GPIO port pins associated with JTAG/SWD functionality. The active sense of the NMI signal is High; asserting the enabled NMI signal above  $V_{IH}$  initiates the NMI interrupt sequence.

#### 6.1.3.2 Main Oscillator Verification Failure

The main oscillator verification circuit may generate a reset event and then, during the subsequent POR, control is transferred to the NMI handler. The detection circuit is enabled using the `CVAL` bit in the **Main Oscillator Control (MOSCCTL)** register. The main oscillator verification error is indicated in the main oscillator fail status bit (`MOSCFAIL` bit in the **Reset Cause (RESC)** register. The main oscillator verification circuit action is described in more detail in “Clock Control” on page 63.

### 6.1.4 Power Control

The Stellaris<sup>®</sup> microcontroller provides an integrated LDO regulator that may be used to provide power to the majority of the controller’s internal logic. The LDO regulator provides software a mechanism to adjust the regulated value, in small increments (`VSTEP`), over the range of 2.25 V to 2.75 V (inclusive)—or  $2.5\text{ V} \pm 10\%$ . The adjustment is made by changing the value of the `VADJ` field in the **LDO Power Control (LDOPCTL)** register.

**Note:** On the printed circuit board, use the LDO output as the source of `VDD25` input. In addition, the LDO requires decoupling capacitors. See “On-Chip Low Drop-Out (LDO) Regulator Characteristics” on page 501.

### 6.1.5 Clock Control

System control determines the control of clocks in this part.

#### 6.1.5.1 Fundamental Clock Sources

There are four clock sources for use in the device:

- **Internal Oscillator (IOSC):** The internal oscillator is an on-chip clock source. It does not require the use of any external components. The frequency of the internal oscillator is  $12\text{ MHz} \pm 30\%$ . Applications that do not depend on accurate clock sources may use this clock source to reduce system cost. The internal oscillator is the clock source the device uses during and following POR. If the main oscillator is required, software must enable the main oscillator following reset and allow the main oscillator to stabilize before changing the clock reference.

- **Main Oscillator (MOSC):** The main oscillator provides a frequency-accurate clock source by one of two means: an external single-ended clock source is connected to the `OSC0` input pin, or an external crystal is connected across the `OSC0` input and `OSC1` output pins. If the PLL is being used, the crystal value must be one of the supported frequencies between 3.579545 MHz through 16.384 MHz (inclusive). If the PLL is not being used, the crystal may be any one of the supported frequencies between 1 MHz and 16.384 MHz. The single-ended clock source range is from DC through the specified speed of the device. The supported crystals are listed in the `XTAL` bit field in the **RCC** register (see page 79).
- **Internal 30-kHz Oscillator:** The internal 30-kHz oscillator is similar to the internal oscillator, except that it provides an operational frequency of 30 kHz  $\pm$  50%. It is intended for use during Deep-Sleep power-saving modes. This power-savings mode benefits from reduced internal switching and also allows the main oscillator to be powered down.
- **External Real-Time Oscillator:** The external real-time oscillator provides a low-frequency, accurate clock reference. It is intended to provide the system with a real-time clock source. The real-time oscillator is part of the Hibernation Module (“Hibernation Module” on page 124) and may also provide an accurate source of Deep-Sleep or Hibernate mode power savings.

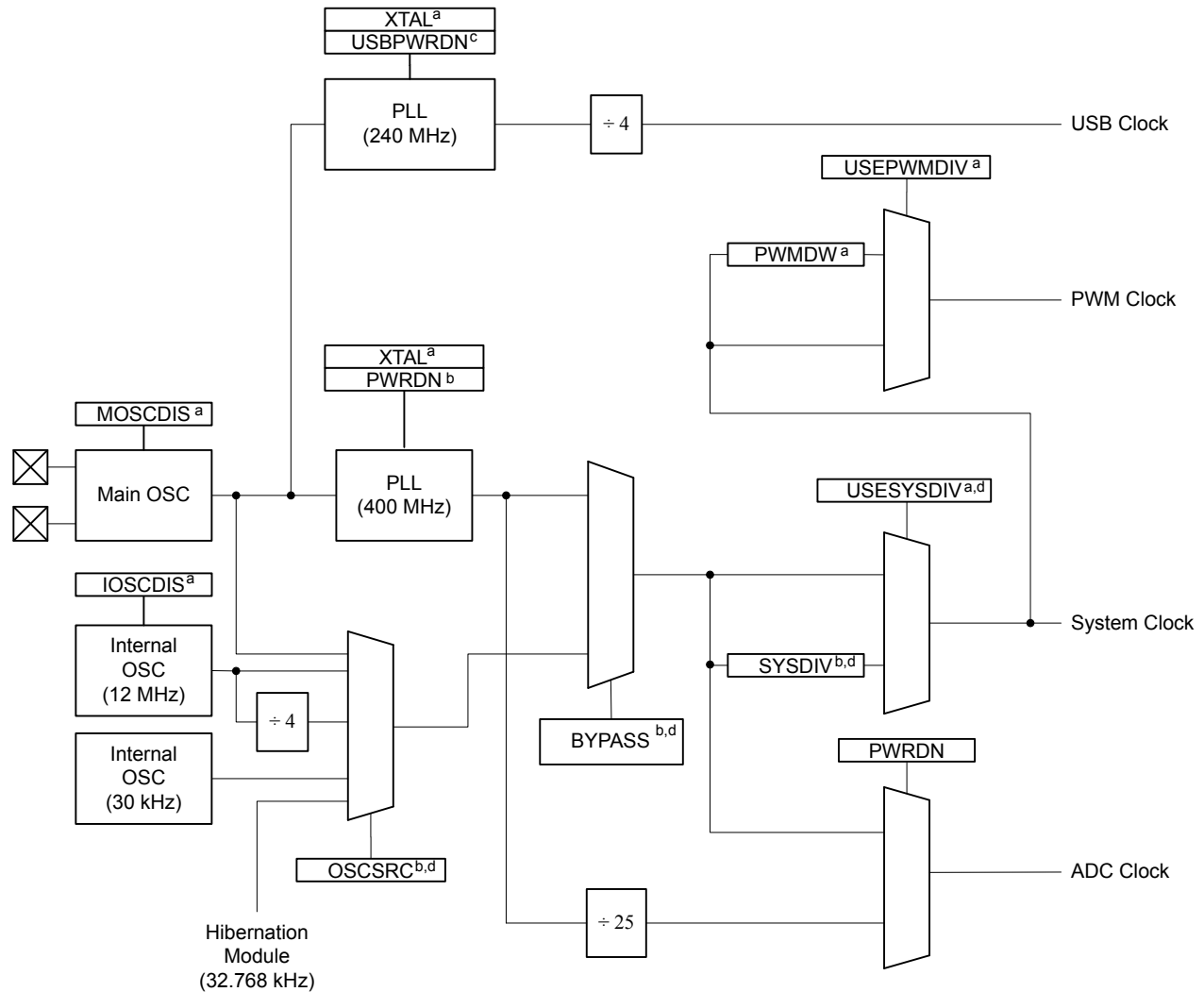
The internal system clock (SysClk), is derived from any of the four sources plus two others: the output of the main internal PLL, and the internal oscillator divided by four (3 MHz  $\pm$  30%). The frequency of the PLL clock reference must be in the range of 3.579545 MHz to 16.384 MHz (inclusive).

The **Run-Mode Clock Configuration (RCC)** and **Run-Mode Clock Configuration 2 (RCC2)** registers provide control for the system clock. The **RCC2** register is provided to extend fields that offer additional encodings over the **RCC** register. When used, the **RCC2** register field values are used by the logic over the corresponding field in the **RCC** register. In particular, **RCC2** provides for a larger assortment of clock configuration options.

Figure 6-2 on page 65 shows the logic for the main clock tree. The peripheral blocks are driven by the system clock signal and can be programmatically enabled/disabled. The ADC clock signal is automatically divided down to 16 MHz for proper ADC operation.



Figure 6-2. Main Clock Tree



- a. Control provided by RCC register bit/field.  
 b. Control provided by RCC register bit/field or RCC2 register bit/field, if overridden with RCC2 register bit USERCC2.  
 c. Control provided by RCC2 register bit/field.  
 d. Also may be controlled by DSLPCLKCFG when in deep sleep mode.

**Note:** The figure above shows all features available on all Stellaris® DustDevil-class devices.

### 6.1.5.2 Crystal Configuration for the Main Oscillator (MOSC)

The main oscillator supports the use of a select number of crystals. If the main oscillator is used by the PLL as a reference clock, the supported range of crystals is 3.579545 to 16.384 MHz, otherwise, the range of supported crystals is 1 to 16.384 MHz.

The XTAL bit in the **RCC** register (see page 79) describes the available crystal choices and default programming values.

Software configures the **RCC** register `XTAL` field with the crystal number. If the PLL is used in the design, the `XTAL` field value is internally translated to the PLL settings.

### 6.1.5.3 Main PLL Frequency Configuration

The main PLL is disabled by default during power-on reset and is enabled later by software if required. Software specifies the output divisor to set the system clock frequency, and enables the main PLL to drive the output.

If the main oscillator provides the clock reference to the main PLL, the translation provided by hardware and used to program the PLL is available for software in the **XTAL to PLL Translation (PLLCFG)** register (see page 83). The internal translation provides a translation within  $\pm 1\%$  of the targeted PLL VCO frequency.

The Crystal Value field (`XTAL`) on page 79 describes the available crystal choices and default programming of the **PLLCFG** register. The crystal number is written into the `XTAL` field of the **Run-Mode Clock Configuration (RCC)** register. Any time the `XTAL` field changes, the new settings are translated and the internal PLL settings are updated.

### 6.1.5.4 PLL Modes

The PLL has two modes of operation: Normal and Power-Down

- Normal: The PLL multiplies the input clock reference and drives the output.
- Power-Down: Most of the PLL internal circuitry is disabled and the PLL does not drive the output.

The modes are programmed using the **RCC/RCC2** register fields (see page 79 and page 85).

### 6.1.5.5 PLL Operation

If a PLL configuration is changed, the PLL output frequency is unstable until it reconverges (relocks) to the new setting. The time between the configuration change and relock is  $T_{\text{READY}}$  (see Table 20-7 on page 503). During the relock time, the affected PLL is not usable as a clock reference.

The PLL is changed by one of the following:

- Change to the `XTAL` value in the **RCC** register—writes of the same value do not cause a relock.
- Change in the PLL from Power-Down to Normal mode.

A counter is defined to measure the  $T_{\text{READY}}$  requirement. The counter is clocked by the main oscillator. The range of the main oscillator has been taken into account and the down counter is set to 0x1200 (that is,  $\sim 600 \mu\text{s}$  at an 8.192 MHz external oscillator clock). When the `XTAL` value is greater than 0x0f, the down counter is set to 0x2400 to maintain the required lock time on higher frequency crystal inputs. Hardware is provided to keep the PLL from being used as a system clock until the  $T_{\text{READY}}$  condition is met after one of the two changes above. It is the user's responsibility to have a stable clock source (like the main oscillator) before the **RCC/RCC2** register is switched to use the PLL.

If the main PLL is enabled and the system clock is switched to use the PLL in one step, the system control hardware continues to clock the controller from the oscillator selected by the **RCC/RCC2** register until the main PLL is stable ( $T_{\text{READY}}$  time met), after which it changes to the PLL. Software can use many methods to ensure that the system is clocked from the main PLL, including periodically polling the `PLLRLIS` bit in the **Raw Interrupt Status (RIS)** register, and enabling the PLL Lock interrupt.

### 6.1.5.6 Main Oscillator Verification Circuit

A circuit is added to ensure that the main oscillator is running at the appropriate frequency. The circuit monitors the main oscillator frequency and signals if the frequency is outside of the allowable band of attached crystals.

The detection circuit is enabled using the `CVAL` bit in the **Main Oscillator Control (MOSCCTL)** register. If this circuit is enabled and detects an error, the following sequence is performed by the hardware:

1. The `MOSCFAIL` bit in the **Reset Cause (RESC)** register is set.
2. If the internal oscillator (IOSC) is disabled, it is enabled.
3. The system clock is switched from the main oscillator to the IOSC.
4. A system-wide reset is initiated that lasts for 32 IOSC periods.
5. Reset is de-asserted and the processor is directed to the NMI handler during the reset sequence.

### 6.1.6 System Control

For power-savings purposes, the **RCGCn**, **SCGCn**, and **DCGCn** registers control the clock gating logic for each peripheral or block in the system while the controller is in Run, Sleep, and Deep-Sleep mode, respectively.

In Run mode, the processor executes code. In Sleep mode, the clock frequency of the active peripherals is unchanged, but the processor is not clocked and therefore no longer executes code. In Deep-Sleep mode, the clock frequency of the active peripherals may change (depending on the Run mode clock configuration) in addition to the processor clock being stopped. An interrupt returns the device to Run mode from one of the sleep modes; the sleep modes are entered on request from the code. Each mode is described in more detail below.

There are four levels of operation for the device defined as:

- **Run Mode.** Run mode provides normal operation of the processor and all of the peripherals that are currently enabled by the **RCGCn** registers. The system clock can be any of the available clock sources including the PLL.
- **Sleep Mode.** Sleep mode is entered by the Cortex-M3 core executing a `WFI` (Wait for Interrupt) instruction. Any properly configured interrupt event in the system will bring the processor back into Run mode. See the system control NVIC section of the *ARM® Cortex™-M3 Technical Reference Manual* for more details.

In Sleep mode, the Cortex-M3 processor core and the memory subsystem are not clocked. Peripherals are clocked that are enabled in the **SCGCn** register when auto-clock gating is enabled (see the **RCC** register) or the **RCGCn** register when the auto-clock gating is disabled. The system clock has the same source and frequency as that during Run mode.

- **Deep-Sleep Mode.** Deep-Sleep mode is entered by first writing the Deep Sleep Enable bit in the ARM Cortex-M3 NVIC system control register and then executing a `WFI` instruction. Any properly configured interrupt event in the system will bring the processor back into Run mode. See the system control NVIC section of the *ARM® Cortex™-M3 Technical Reference Manual* for more details.

The Cortex-M3 processor core and the memory subsystem are not clocked. Peripherals are clocked that are enabled in the **DCGCn** register when auto-clock gating is enabled (see the **RCC**

register) or the **RCGCn** register when auto-clock gating is disabled. The system clock source is the main oscillator by default or the internal oscillator specified in the **DSLPCCLKCFG** register if one is enabled. When the **DSLPCCLKCFG** register is used, the internal oscillator is powered up, if necessary, and the main oscillator is powered down. If the PLL is running at the time of the WFI instruction, hardware will power the PLL down and override the **SYSDIV** field of the active **RCC/RCC2** register to be /16 or /64, respectively. When the Deep-Sleep exit event occurs, hardware brings the system clock back to the source and frequency it had at the onset of Deep-Sleep mode before enabling the clocks that had been stopped during the Deep-Sleep duration.

- **Hibernate Mode.** In this mode, the power supplies are turned off to the main part of the device and only the Hibernation module's circuitry is active. An external wake event or RTC event is required to bring the device back to Run mode. The Cortex-M3 processor and peripherals outside of the Hibernation module see a normal "power on" sequence and the processor starts running code. It can determine that it has been restarted from Hibernate mode by inspecting the Hibernation module registers.

## 6.2 Initialization and Configuration

The PLL is configured using direct register writes to the **RCC/RCC2** register. If the **RCC2** register is being used, the **USERCC2** bit must be set and the appropriate **RCC2** bit/field is used. The steps required to successfully change the PLL-based system clock are:

1. Bypass the PLL and system clock divider by setting the **BYPASS** bit and clearing the **USESYS** bit in the **RCC** register. This configures the system to run off a "raw" clock source (using the main oscillator or internal oscillator) and allows for the new PLL configuration to be validated before switching the system clock to the PLL.
2. Select the crystal value (**XTAL**) and oscillator source (**OSCSRC**), and clear the **PWRDN** bit in **RCC/RCC2**. Setting the **XTAL** field automatically pulls valid PLL configuration data for the appropriate crystal, and clearing the **PWRDN** bit powers and enables the PLL and its output.
3. Select the desired system divider (**SYSDIV**) in **RCC/RCC2** and set the **USESYS** bit in **RCC**. The **SYSDIV** field determines the system frequency for the microcontroller.
4. Wait for the PLL to lock by polling the **PLLLRIS** bit in the **Raw Interrupt Status (RIS)** register.
5. Enable use of the PLL by clearing the **BYPASS** bit in **RCC/RCC2**.

## 6.3 Register Map

Table 6-1 on page 69 lists the System Control registers, grouped by function. The offset listed is a hexadecimal increment to the register's address, relative to the System Control base address of 0x400F.E000.

**Note:** Spaces in the System Control register space that are not used are reserved for future or internal use by Luminary Micro, Inc. Software should not modify any reserved memory address.

**Note:** Additional Flash and ROM registers defined in the System Control register space are described in the "Internal Memory" on page 147.

Table 6-1. System Control Register Map

Offset	Name	Type	Reset	Description	See page
0x000	DID0	RO	-	Device Identification 0	71
0x004	DID1	RO	-	Device Identification 1	89
0x008	DC0	RO	0x007F.003F	Device Capabilities 0	91
0x010	DC1	RO	0x0001.32FF	Device Capabilities 1	92
0x014	DC2	RO	0x000F.5017	Device Capabilities 2	94
0x018	DC3	RO	0xBFFF.0000	Device Capabilities 3	96
0x01C	DC4	RO	0x0000.301F	Device Capabilities 4	98
0x020	DC5	RO	0x0000.0000	Device Capabilities 5	99
0x024	DC6	RO	0x0000.0000	Device Capabilities 6	100
0x028	DC7	RO	0x00C0.0F00	Device Capabilities 7	101
0x030	PBORCTL	R/W	0x0000.7FFD	Brown-Out Reset Control	73
0x034	LDO PCTL	R/W	0x0000.0000	LDO Power Control	74
0x040	SRCR0	R/W	0x00000000	Software Reset Control 0	120
0x044	SRCR1	R/W	0x00000000	Software Reset Control 1	121
0x048	SRCR2	R/W	0x00000000	Software Reset Control 2	123
0x050	RIS	RO	0x0000.0000	Raw Interrupt Status	75
0x054	IMC	R/W	0x0000.0000	Interrupt Mask Control	76
0x058	MISC	R/W1C	0x0000.0000	Masked Interrupt Status and Clear	77
0x05C	RESC	R/W	-	Reset Cause	78
0x060	RCC	R/W	0x0780.3AD1	Run-Mode Clock Configuration	79
0x064	PLLCFG	RO	-	XTAL to PLL Translation	83
0x06C	GPIOHSCTL	R/W	0x0000.0000	GPIO High Speed Control	84
0x070	RCC2	R/W	0x0780.6810	Run-Mode Clock Configuration 2	85
0x07C	MOSCCTL	R/W	0x0000.0000	Main Oscillator Control	87
0x100	RCGC0	R/W	0x00000040	Run Mode Clock Gating Control Register 0	102
0x104	RCGC1	R/W	0x00000000	Run Mode Clock Gating Control Register 1	108
0x108	RCGC2	R/W	0x00000000	Run Mode Clock Gating Control Register 2	114
0x110	SCGC0	R/W	0x00000040	Sleep Mode Clock Gating Control Register 0	104
0x114	SCGC1	R/W	0x00000000	Sleep Mode Clock Gating Control Register 1	110
0x118	SCGC2	R/W	0x00000000	Sleep Mode Clock Gating Control Register 2	116
0x120	DCGC0	R/W	0x00000040	Deep Sleep Mode Clock Gating Control Register 0	106
0x124	DCGC1	R/W	0x00000000	Deep Sleep Mode Clock Gating Control Register 1	112

Offset	Name	Type	Reset	Description	See page
0x128	DCGC2	R/W	0x00000000	Deep Sleep Mode Clock Gating Control Register 2	118
0x144	DSLCLKCFG	R/W	0x0780.0000	Deep Sleep Clock Configuration	88

## 6.4 Register Descriptions

All addresses given are relative to the System Control base address of 0x400F.E000.

## Register 1: Device Identification 0 (DID0), offset 0x000

This register identifies the version of the device.

### Device Identification 0 (DID0)

Base 0x400F.E000

Offset 0x000

Type RO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved	VER			reserved				CLASS							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MAJOR								MINOR							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description				
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.				
30:28	VER	RO	0x1	<p>DID0 Version</p> <p>This field defines the <b>DID0</b> register format version. The version number is numeric. The value of the <code>VER</code> field is encoded as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x1</td> <td>Second version of the <b>DID0</b> register format.</td> </tr> </tbody> </table>	Value	Description	0x1	Second version of the <b>DID0</b> register format.
Value	Description							
0x1	Second version of the <b>DID0</b> register format.							
27:24	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.				
23:16	CLASS	RO	0x3	<p>Device Class</p> <p>The <code>CLASS</code> field value identifies the internal design from which all mask sets are generated for all devices in a particular product line. The <code>CLASS</code> field value is changed for new product lines, for changes in fab process (for example, a remap or shrink), or any case where the <code>MAJOR</code> or <code>MINOR</code> fields require differentiation from prior devices. The value of the <code>CLASS</code> field is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x3</td> <td>Stellaris® DustDevil-class devices</td> </tr> </tbody> </table>	Value	Description	0x3	Stellaris® DustDevil-class devices
Value	Description							
0x3	Stellaris® DustDevil-class devices							

Bit/Field	Name	Type	Reset	Description								
15:8	MAJOR	RO	-	<p>Major Revision</p> <p>This field specifies the major revision number of the device. The major revision reflects changes to base layers of the design. The major revision number is indicated in the part number as a letter (A for first revision, B for second, and so on). This field is encoded as follows:</p> <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0x0</td><td>Revision A (initial device)</td></tr><tr><td>0x1</td><td>Revision B (first base layer revision)</td></tr><tr><td>0x2</td><td>Revision C (second base layer revision)</td></tr></tbody></table> <p>and so on.</p>	Value	Description	0x0	Revision A (initial device)	0x1	Revision B (first base layer revision)	0x2	Revision C (second base layer revision)
Value	Description											
0x0	Revision A (initial device)											
0x1	Revision B (first base layer revision)											
0x2	Revision C (second base layer revision)											
7:0	MINOR	RO	-	<p>Minor Revision</p> <p>This field specifies the minor revision number of the device. The minor revision reflects changes to the metal layers of the design. The <code>MINOR</code> field value is reset when the <code>MAJOR</code> field is changed. This field is numeric and is encoded as follows:</p> <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0x0</td><td>Initial device, or a major revision update.</td></tr><tr><td>0x1</td><td>First metal layer change.</td></tr><tr><td>0x2</td><td>Second metal layer change.</td></tr></tbody></table> <p>and so on.</p>	Value	Description	0x0	Initial device, or a major revision update.	0x1	First metal layer change.	0x2	Second metal layer change.
Value	Description											
0x0	Initial device, or a major revision update.											
0x1	First metal layer change.											
0x2	Second metal layer change.											



## Register 2: Brown-Out Reset Control (PBORCTL), offset 0x030

This register is responsible for controlling reset conditions after initial power-on reset.

### Brown-Out Reset Control (PBORCTL)

Base 0x400F.E000

Offset 0x030

Type R/W, reset 0x0000.7FFD

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

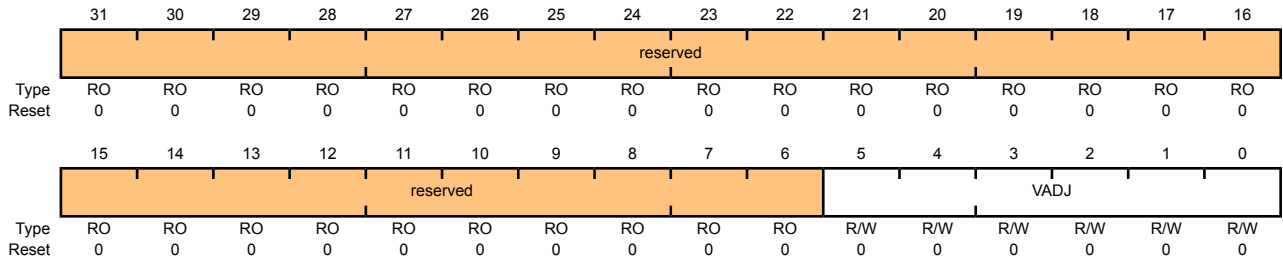
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BORIOR	R/W	0	BOR Interrupt or Reset  This bit controls how a BOR event is signaled to the controller. If set, a reset is signaled. Otherwise, an interrupt is signaled.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

### Register 3: LDO Power Control (LDOPCTL), offset 0x034

The V<sub>ADJ</sub> field in this register adjusts the on-chip output voltage (V<sub>OUT</sub>).

#### LDO Power Control (LDOPCTL)

Base 0x400F.E000  
 Offset 0x034  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
5:0	VADJ	R/W	0x0	LDO Output Voltage

This field sets the on-chip output voltage. The programming values for the V<sub>ADJ</sub> field are provided below.

Value	V <sub>OUT</sub> (V)
0x00	2.50
0x01	2.45
0x02	2.40
0x03	2.35
0x04	2.30
0x05	2.25
0x06-0x3F	Reserved
0x1B	2.75
0x1C	2.70
0x1D	2.65
0x1E	2.60
0x1F	2.55

## Register 4: Raw Interrupt Status (RIS), offset 0x050

Central location for system control raw interrupts. These are set and cleared by hardware.

### Raw Interrupt Status (RIS)

Base 0x400F.E000

Offset 0x050

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved							MOSCPUPRIS	reserved	PLLLRIS	reserved				BORRIS	reserved
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	MOSCPUPRIS	RO	0	MOSC Power Up Raw Interrupt Status This bit is set when the PLL $T_{MOSCPUP}$ Timer asserts.
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PLLLRIS	RO	0	PLL Lock Raw Interrupt Status This bit is set when the PLL $T_{READY}$ Timer asserts.
5:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BORRIS	RO	0	Brown-Out Reset Raw Interrupt Status This bit is the raw interrupt status for any brown-out conditions. If set, a brown-out condition is currently active. This is an unregistered signal from the brown-out detection circuit. An interrupt is reported if the <b>BORIM</b> bit in the <b>IMC</b> register is set and the <b>BORIOR</b> bit in the <b>PBORCTL</b> register is cleared.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 5: Interrupt Mask Control (IMC), offset 0x054

Central location for system control interrupt masks.

### Interrupt Mask Control (IMC)

Base 0x400F.E000  
Offset 0x054  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved							MOSCPUPIM	reserved	PLLLIM	reserved				BORIM	reserved
Type	RO	RO	RO	RO	RO	RO	RO	R/W	RO	R/W	RO	RO	RO	RO	R/W	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	MOSCPUPIM	R/W	0	MOSC Power Up Interrupt Mask  This bit specifies whether a current limit detection is promoted to a controller interrupt. If set, an interrupt is generated if MOSCPUPRIS in <b>RIS</b> is set; otherwise, an interrupt is not generated.
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PLLLIM	R/W	0	PLL Lock Interrupt Mask  This bit specifies whether a current limit detection is promoted to a controller interrupt. If set, an interrupt is generated if PLLLRIS in <b>RIS</b> is set; otherwise, an interrupt is not generated.
5:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BORIM	R/W	0	Brown-Out Reset Interrupt Mask  This bit specifies whether a brown-out condition is promoted to a controller interrupt. If set, an interrupt is generated if BORRIS is set; otherwise, an interrupt is not generated.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 6: Masked Interrupt Status and Clear (MISC), offset 0x058

On a read, this register gives the current masked status value of the corresponding interrupt. All of the bits are R/W1C and this action also clears the corresponding raw interrupt bit in the **RIS** register (see page 75).

### Masked Interrupt Status and Clear (MISC)

Base 0x400F.E000

Offset 0x058

Type R/W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved							MOSCPUPMIS	reserved	PLLLMIS	reserved				BORMIS	reserved
Type	RO	RO	RO	RO	RO	RO	RO	R/W1C	RO	R/W1C	RO	RO	RO	RO	R/W1C	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	MOSCPUPMIS	R/W1C	0	MOSC Power Up Masked Interrupt Status  This bit is set when the $T_{MOSCPUP}$ timer asserts. The interrupt is cleared by writing a 1 to this bit.
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PLLLMIS	R/W1C	0	PLL Lock Masked Interrupt Status  This bit is set when the PLL $T_{READY}$ timer asserts. The interrupt is cleared by writing a 1 to this bit.
5:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BORMIS	R/W1C	0	BOR Masked Interrupt Status  The <b>BORMIS</b> is simply the <b>BORRIS</b> ANDed with the mask value, <b>BORIM</b> .
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 7: Reset Cause (RESC), offset 0x05C

This register is set with the reset cause after reset. The bits in this register are sticky and maintain their state across multiple reset sequences, except when an external reset is the cause, and then all the other bits in the **RESC** register are cleared.

### Reset Cause (RESC)

Base 0x400F.E000

Offset 0x05C

Type R/W, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															MOSCFAIL
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved											SW	WDT	BOR	POR	EXT
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	MOSCFAIL	R/W	-	MOSC Failure Reset When set, indicates the MOSC circuit was enable for clock validation and failed. This generated a reset event.
15:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SW	R/W	-	Software Reset When set, indicates a software reset is the cause of the reset event.
3	WDT	R/W	-	Watchdog Timer Reset When set, indicates a watchdog reset is the cause of the reset event.
2	BOR	R/W	-	Brown-Out Reset When set, indicates a brown-out reset is the cause of the reset event.
1	POR	R/W	-	Power-On Reset When set, indicates a power-on reset is the cause of the reset event.
0	EXT	R/W	-	External Reset When set, indicates an external reset ( $\overline{RST}$ assertion) is the cause of the reset event.

## Register 8: Run-Mode Clock Configuration (RCC), offset 0x060

This register is defined to provide source control and frequency speed.

### Run-Mode Clock Configuration (RCC)

Base 0x400F.E000

Offset 0x060

Type R/W, reset 0x0780.3AD1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved				ACG	SYSDIV				USESYS DIV	reserved					
Type	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		PWRDN	reserved	BYPASS	XTAL				OSCSRC		reserved		IOSCDIS	MOSCDIS	
Type	RO	RO	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO	R/W	R/W
Reset	0	0	1	1	1	0	1	0	1	1	0	1	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:28	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
27	ACG	R/W	0	<p>Auto Clock Gating</p> <p>This bit specifies whether the system uses the <b>Sleep-Mode Clock Gating Control (SCGCn)</b> registers and <b>Deep-Sleep-Mode Clock Gating Control (DCGCn)</b> registers if the controller enters a Sleep or Deep-Sleep mode (respectively). If set, the <b>SCGCn</b> or <b>DCGCn</b> registers are used to control the clocks distributed to the peripherals when the controller is in a sleep mode. Otherwise, the <b>Run-Mode Clock Gating Control (RCGCn)</b> registers are used when the controller enters a sleep mode.</p> <p>The <b>RCGCn</b> registers are always used to control the clocks in Run mode.</p> <p>This allows peripherals to consume less power when the controller is in a sleep mode and the peripheral is unused.</p>

Bit/Field	Name	Type	Reset	Description																																																			
26:23	SYSDIV	R/W	0xF	<p>System Clock Divisor</p> <p>Specifies which divisor is used to generate the system clock from the PLL output.</p> <p>The PLL VCO frequency is 400 MHz.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Divisor (BYPASS=1)</th> <th>Frequency (BYPASS=0)</th> </tr> </thead> <tbody> <tr><td>0x0</td><td>reserved</td><td>reserved</td></tr> <tr><td>0x1</td><td>/2</td><td>reserved</td></tr> <tr><td>0x2</td><td>/3</td><td>reserved</td></tr> <tr><td>0x3</td><td>/4</td><td>50 MHz</td></tr> <tr><td>0x4</td><td>/5</td><td>40 MHz</td></tr> <tr><td>0x5</td><td>/6</td><td>33.33 MHz</td></tr> <tr><td>0x6</td><td>/7</td><td>28.57 MHz</td></tr> <tr><td>0x7</td><td>/8</td><td>25 MHz</td></tr> <tr><td>0x8</td><td>/9</td><td>22.22 MHz</td></tr> <tr><td>0x9</td><td>/10</td><td>20 MHz</td></tr> <tr><td>0xA</td><td>/11</td><td>18.18 MHz</td></tr> <tr><td>0xB</td><td>/12</td><td>16.67 MHz</td></tr> <tr><td>0xC</td><td>/13</td><td>15.38 MHz</td></tr> <tr><td>0xD</td><td>/14</td><td>14.29 MHz</td></tr> <tr><td>0xE</td><td>/15</td><td>13.33 MHz</td></tr> <tr><td>0xF</td><td>/16</td><td>12.5 MHz (default)</td></tr> </tbody> </table> <p>When reading the <b>Run-Mode Clock Configuration (RCC)</b> register (see page 79), the SYSDIV value is MINSYSDIV if a lower divider was requested and the PLL is being used. This lower value is allowed to divide a non-PLL source.</p>	Value	Divisor (BYPASS=1)	Frequency (BYPASS=0)	0x0	reserved	reserved	0x1	/2	reserved	0x2	/3	reserved	0x3	/4	50 MHz	0x4	/5	40 MHz	0x5	/6	33.33 MHz	0x6	/7	28.57 MHz	0x7	/8	25 MHz	0x8	/9	22.22 MHz	0x9	/10	20 MHz	0xA	/11	18.18 MHz	0xB	/12	16.67 MHz	0xC	/13	15.38 MHz	0xD	/14	14.29 MHz	0xE	/15	13.33 MHz	0xF	/16	12.5 MHz (default)
Value	Divisor (BYPASS=1)	Frequency (BYPASS=0)																																																					
0x0	reserved	reserved																																																					
0x1	/2	reserved																																																					
0x2	/3	reserved																																																					
0x3	/4	50 MHz																																																					
0x4	/5	40 MHz																																																					
0x5	/6	33.33 MHz																																																					
0x6	/7	28.57 MHz																																																					
0x7	/8	25 MHz																																																					
0x8	/9	22.22 MHz																																																					
0x9	/10	20 MHz																																																					
0xA	/11	18.18 MHz																																																					
0xB	/12	16.67 MHz																																																					
0xC	/13	15.38 MHz																																																					
0xD	/14	14.29 MHz																																																					
0xE	/15	13.33 MHz																																																					
0xF	/16	12.5 MHz (default)																																																					
22	USESYSCLK	R/W	0	<p>Enable System Clock Divider</p> <p>Use the system clock divider as the source for the system clock. The system clock divider is forced to be used when the PLL is selected as the source.</p>																																																			
21:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.																																																			
13	PWRDN	R/W	1	<p>PLL Power Down</p> <p>This bit connects to the PLL PWRDN input. The reset value of 1 powers down the PLL.</p>																																																			
12	reserved	RO	1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.																																																			



Bit/Field	Name	Type	Reset	Description																																																																								
11	BYPASS	R/W	1	<p>PLL Bypass</p> <p>Chooses whether the system clock is derived from the PLL output or the OSC source. If set, the clock that drives the system is the OSC source. Otherwise, the clock that drives the system is the PLL output clock divided by the system divider.</p> <p><b>Note:</b> The ADC must be clocked from the PLL or directly from a 14-MHz to 18-MHz clock source to operate properly. While the ADC works in a 14-18 MHz range, to maintain a 1 M sample/second rate, the ADC must be provided a 16-MHz clock source.</p>																																																																								
10:6	XTAL	R/W	0xB	<p>Crystal Value</p> <p>This field specifies the crystal value attached to the main oscillator. The encoding for this field is provided below.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Crystal Frequency (MHz) Not Using the PLL</th> <th>Crystal Frequency (MHz) Using the PLL</th> </tr> </thead> <tbody> <tr><td>0x00</td><td>1.000</td><td>reserved</td></tr> <tr><td>0x01</td><td>1.8432</td><td>reserved</td></tr> <tr><td>0x02</td><td>2.000</td><td>reserved</td></tr> <tr><td>0x03</td><td>2.4576</td><td>reserved</td></tr> <tr><td>0x04</td><td></td><td>3.579545 MHz</td></tr> <tr><td>0x05</td><td></td><td>3.6864 MHz</td></tr> <tr><td>0x06</td><td></td><td>4 MHz</td></tr> <tr><td>0x07</td><td></td><td>4.096 MHz</td></tr> <tr><td>0x08</td><td></td><td>4.9152 MHz</td></tr> <tr><td>0x09</td><td></td><td>5 MHz</td></tr> <tr><td>0x0A</td><td></td><td>5.12 MHz</td></tr> <tr><td>0x0B</td><td></td><td>6 MHz (reset value)</td></tr> <tr><td>0x0C</td><td></td><td>6.144 MHz</td></tr> <tr><td>0x0D</td><td></td><td>7.3728 MHz</td></tr> <tr><td>0x0E</td><td></td><td>8 MHz</td></tr> <tr><td>0x0F</td><td></td><td>8.192 MHz</td></tr> <tr><td>0x10</td><td></td><td>10.0 MHz</td></tr> <tr><td>0x11</td><td></td><td>12.0 MHz</td></tr> <tr><td>0x12</td><td></td><td>12.288 MHz</td></tr> <tr><td>0x13</td><td></td><td>13.56 MHz</td></tr> <tr><td>0x14</td><td></td><td>14.31818 MHz</td></tr> <tr><td>0x15</td><td></td><td>16.0 MHz</td></tr> <tr><td>0x16</td><td></td><td>16.384 MHz</td></tr> </tbody> </table>	Value	Crystal Frequency (MHz) Not Using the PLL	Crystal Frequency (MHz) Using the PLL	0x00	1.000	reserved	0x01	1.8432	reserved	0x02	2.000	reserved	0x03	2.4576	reserved	0x04		3.579545 MHz	0x05		3.6864 MHz	0x06		4 MHz	0x07		4.096 MHz	0x08		4.9152 MHz	0x09		5 MHz	0x0A		5.12 MHz	0x0B		6 MHz (reset value)	0x0C		6.144 MHz	0x0D		7.3728 MHz	0x0E		8 MHz	0x0F		8.192 MHz	0x10		10.0 MHz	0x11		12.0 MHz	0x12		12.288 MHz	0x13		13.56 MHz	0x14		14.31818 MHz	0x15		16.0 MHz	0x16		16.384 MHz
Value	Crystal Frequency (MHz) Not Using the PLL	Crystal Frequency (MHz) Using the PLL																																																																										
0x00	1.000	reserved																																																																										
0x01	1.8432	reserved																																																																										
0x02	2.000	reserved																																																																										
0x03	2.4576	reserved																																																																										
0x04		3.579545 MHz																																																																										
0x05		3.6864 MHz																																																																										
0x06		4 MHz																																																																										
0x07		4.096 MHz																																																																										
0x08		4.9152 MHz																																																																										
0x09		5 MHz																																																																										
0x0A		5.12 MHz																																																																										
0x0B		6 MHz (reset value)																																																																										
0x0C		6.144 MHz																																																																										
0x0D		7.3728 MHz																																																																										
0x0E		8 MHz																																																																										
0x0F		8.192 MHz																																																																										
0x10		10.0 MHz																																																																										
0x11		12.0 MHz																																																																										
0x12		12.288 MHz																																																																										
0x13		13.56 MHz																																																																										
0x14		14.31818 MHz																																																																										
0x15		16.0 MHz																																																																										
0x16		16.384 MHz																																																																										

Bit/Field	Name	Type	Reset	Description
5:4	OSCSRC	R/W	0x1	Oscillator Source Picks among the four input sources for the OSC. The values are:  Value Input Source 0x0 Main oscillator 0x1 Internal oscillator (default) 0x2 Internal oscillator / 4 (this is necessary if used as input to PLL) 0x3 30 KHz internal oscillator
3:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	IOSCDIS	R/W	0	Internal Oscillator Disable 0: Internal oscillator (IOSC) is enabled. 1: Internal oscillator is disabled.
0	MOSCDIS	R/W	1	Main Oscillator Disable 0: Main oscillator is enabled . 1: Main oscillator is disabled (default).

## Register 9: XTAL to PLL Translation (PLLCFG), offset 0x064

This register provides a means of translating external crystal frequencies into the appropriate PLL settings. This register is initialized during the reset sequence and updated anytime that the XTAL field changes in the **Run-Mode Clock Configuration (RCC)** register (see page 79).

The PLL frequency is calculated using the **PLLCFG** field values, as follows:

$$\text{PLLFreq} = \text{OSCFreq} * F / (R + 1)$$

### XTAL to PLL Translation (PLLCFG)

Base 0x400F.E000

Offset 0x064

Type RO, reset -

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

Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13:5	F	RO	-	PLL F Value This field specifies the value supplied to the PLL's F input.
4:0	R	RO	-	PLL R Value This field specifies the value supplied to the PLL's R input.

### Register 10: GPIO High Speed Control (GPIOHSCTL), offset 0x06C

This register provides the user the ability to change the GPIO ports to run on a single-cycle bus equivalent to the processor clock instead of the legacy bus with two-cycle access. The address aperture in the memory map will change for the ports that are enabled for high-speed access (see Table 10-3 on page 244).

#### GPIO High Speed Control (GPIOHSCTL)

Base 0x400F.E000  
 Offset 0x06C  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved												PORTEHS	PORTDHS	PORTCHS	PORTBHS	PORTAHS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	PORTEHS	R/W	0	Port E High-Speed  When set, the memory aperture for Port H is selected to be high speed (single-cycle). Otherwise, the legacy aperture (two-cycle) is chosen.
3	PORTDHS	R/W	0	Port D High-Speed  When set, the memory aperture for Port H is selected to be high speed (single-cycle). Otherwise, the legacy aperture (two-cycle) is chosen.
2	PORTCHS	R/W	0	Port C High-Speed  When set, the memory aperture for Port H is selected to be high speed (single-cycle). Otherwise, the legacy aperture (two-cycle) is chosen.
1	PORTBHS	R/W	0	Port B High-Speed  When set, the memory aperture for Port H is selected to be high speed (single-cycle). Otherwise, the legacy aperture (two-cycle) is chosen.
0	PORTAHS	R/W	0	Port A High-Speed  When set, the memory aperture for Port H is selected to be high speed (single-cycle). Otherwise, the legacy aperture (two-cycle) is chosen.

## Register 11: Run-Mode Clock Configuration 2 (RCC2), offset 0x070

This register overrides the **RCC** equivalent register fields when the `USERCC2` bit is set. This allows **RCC2** to be used to extend the capabilities, while also providing a means to be backward-compatible to previous parts. The fields within the **RCC2** register occupy the same bit positions as they do within the **RCC** register as LSB-justified.

The `SYSDIV2` field is wider so that additional larger divisors are possible. This allows a lower system clock frequency for improved Deep Sleep power consumption.

### Run-Mode Clock Configuration 2 (RCC2)

Base 0x400F.E000  
Offset 0x070  
Type R/W, reset 0x0780.6810

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	USERCC2	reserved		SYSDIV2						reserved						
Type	R/W	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	reserved-1	PWRDN2	reserved	BYPASS2	reserved				OSCSRC2			reserved			
Type	RO	RO	R/W	RO	R/W	RO	RO	RO	RO	R/W	R/W	R/W	RO	RO	RO	RO
Reset	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	USERCC2	R/W	0	Use RCC2  When set, overrides the <b>RCC</b> register fields.
30:29	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28:23	SYSDIV2	R/W	0x0F	System Clock Divisor  Specifies which divisor is used to generate the system clock from the PLL output.  The PLL VCO frequency is 400 MHz.  This field is wider than the <b>RCC</b> register <code>SYSDIV</code> field in order to provide additional divisor values. This permits the system clock to be run at much lower frequencies during Deep Sleep mode. For example, where the <b>RCC</b> register <code>SYSDIV</code> encoding of 1111 provides /16, the <b>RCC2</b> register <code>SYSDIV2</code> encoding of 111111 provides /64.
22:15	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	reserved-1	RO	1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.  Note that reset value is 1.
13	PWRDN2	R/W	1	Power-Down PLL  When set, powers down the PLL.

Bit/Field	Name	Type	Reset	Description
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	BYPASS2	R/W	1	Bypass PLL When set, bypasses the PLL for the clock source.
10:7	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:4	OSCSRC2	R/W	0x1	Oscillator Source Picks among the input sources for the OSC. The values are:  Value Description 0x0 Main oscillator (MOSC) 0x1 Internal oscillator (IOSC) 0x2 Internal oscillator / 4 0x3 30 kHz internal oscillator 0x7 32 kHz external oscillator
3:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 12: Main Oscillator Control (MOSCCTL), offset 0x07C

This register provides control over the features of the main oscillator, including the ability to enable the MOSC clock validation circuit. When enabled, this circuit monitors the energy on the MOSC pins to provide a Clock Valid signal. If the clock goes invalid after being enabled, the part does a hardware reset and reboots to the NMI handler.

### Main Oscillator Control (MOSCCTL)

Base 0x400F.E000  
Offset 0x07C  
Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	CVAL	R/W	0	Clock Validation for MOSC When set, the monitor circuit is enabled.

## Register 13: Deep Sleep Clock Configuration (DSLPCCLKCFG), offset 0x144

This register provides configuration information for the hardware control of Deep Sleep Mode.

### Deep Sleep Clock Configuration (DSLPCCLKCFG)

Base 0x400F.E000  
 Offset 0x144  
 Type R/W, reset 0x0780.0000

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

Bit/Field	Name	Type	Reset	Description										
31:29	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
28:23	DSDIVORIDE	R/W	0x0F	Divider Field Override 6-bit system divider field to override when Deep-Sleep occurs with PLL running.										
22:7	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
6:4	DSOSCSRC	R/W	0x0	Clock Source Specifies the clock source during Deep-Sleep mode.  <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0x0</td> <td>NOORIDE No override to the oscillator clock source is done.</td> </tr> <tr> <td>0x1</td> <td>IOSC Use internal 12 MHz oscillator as source.</td> </tr> <tr> <td>0x3</td> <td>30kHz Use 30 kHz internal oscillator.</td> </tr> <tr> <td>0x7</td> <td>32kHz Use 32 kHz external oscillator.</td> </tr> </table>	Value	Description	0x0	NOORIDE No override to the oscillator clock source is done.	0x1	IOSC Use internal 12 MHz oscillator as source.	0x3	30kHz Use 30 kHz internal oscillator.	0x7	32kHz Use 32 kHz external oscillator.
Value	Description													
0x0	NOORIDE No override to the oscillator clock source is done.													
0x1	IOSC Use internal 12 MHz oscillator as source.													
0x3	30kHz Use 30 kHz internal oscillator.													
0x7	32kHz Use 32 kHz external oscillator.													
3:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										



## Register 14: Device Identification 1 (DID1), offset 0x004

This register identifies the device family, part number, temperature range, and package type.

### Device Identification 1 (DID1)

Base 0x400F.E000

Offset 0x004

Type RO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	VER				FAM				PARTNO							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PINCOUNT			reserved				TEMP			PKG		ROHS	QUAL		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	1	1	0	0	0	0	0	-	-	-	-	-	1	-	-

Bit/Field	Name	Type	Reset	Description				
31:28	VER	RO	0x1	<p>DID1 Version</p> <p>This field defines the <b>DID1</b> register format version. The version number is numeric. The value of the <code>VER</code> field is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x1</td> <td>Second version of the <b>DID1</b> register format.</td> </tr> </tbody> </table>	Value	Description	0x1	Second version of the <b>DID1</b> register format.
Value	Description							
0x1	Second version of the <b>DID1</b> register format.							
27:24	FAM	RO	0x0	<p>Family</p> <p>This field provides the family identification of the device within the Luminary Micro product portfolio. The value is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Stellaris family of microcontrollers, that is, all devices with external part numbers starting with LM3S.</td> </tr> </tbody> </table>	Value	Description	0x0	Stellaris family of microcontrollers, that is, all devices with external part numbers starting with LM3S.
Value	Description							
0x0	Stellaris family of microcontrollers, that is, all devices with external part numbers starting with LM3S.							
23:16	PARTNO	RO	0x06	<p>Part Number</p> <p>This field provides the part number of the device within the family. The value is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x06</td> <td>LM3S1607</td> </tr> </tbody> </table>	Value	Description	0x06	LM3S1607
Value	Description							
0x06	LM3S1607							
15:13	PINCOUNT	RO	0x3	<p>Package Pin Count</p> <p>This field specifies the number of pins on the device package. The value is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x3</td> <td>64-pin package</td> </tr> </tbody> </table>	Value	Description	0x3	64-pin package
Value	Description							
0x3	64-pin package							

Bit/Field	Name	Type	Reset	Description
12:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:5	TEMP	RO	-	Temperature Range  This field specifies the temperature rating of the device. The value is encoded as follows (all other encodings are reserved):  Value Description 0x0 Commercial temperature range (0°C to 70°C) 0x1 Industrial temperature range (-40°C to 85°C) 0x2 Extended temperature range (-40°C to 105°C)
4:3	PKG	RO	-	Package Type  This field specifies the package type. The value is encoded as follows (all other encodings are reserved):  Value Description 0x0 SOIC package 0x1 LQFP package 0x2 BGA package
2	ROHS	RO	1	RoHS-Compliance  This bit specifies whether the device is RoHS-compliant. A 1 indicates the part is RoHS-compliant.
1:0	QUAL	RO	-	Qualification Status  This field specifies the qualification status of the device. The value is encoded as follows (all other encodings are reserved):  Value Description 0x0 Engineering Sample (unqualified) 0x1 Pilot Production (unqualified) 0x2 Fully Qualified

## Register 15: Device Capabilities 0 (DC0), offset 0x008

This register is predefined by the part and can be used to verify features.

### Device Capabilities 0 (DC0)

Base 0x400F.E000

Offset 0x008

Type RO, reset 0x007F.003F

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

Bit/Field	Name	Type	Reset	Description
31:16	SRAMSZ	RO	0x007F	SRAM Size Indicates the size of the on-chip SRAM memory.  Value Description 0x007F 32 KB of SRAM
15:0	FLASHSZ	RO	0x003F	Flash Size Indicates the size of the on-chip flash memory.  Value Description 0x003F 128 KB of Flash

## Register 16: Device Capabilities 1 (DC1), offset 0x010

This register is predefined by the part and can be used to verify features. The `PWM`, `SARADC0`, `MAXADCSPD`, `WDT`, `SWO`, `SWD`, and `JTAG` bits mask the `RCGC0`, `SCGC0`, and `DCGC0` registers. Other bits are passed as 0. `MAXADCSPD` is clipped to the maximum value specified in **DC1**.

### Device Capabilities 1 (DC1)

Base 0x400F.E000  
 Offset 0x010  
 Type RO, reset 0x0001.32FF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															ADC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MINSYSDIV				reserved		MAXADCSPD	MPU	HIB	TEMPSNS	PLL	WDT	SWO	SWD	JTAG	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	1	1	0	0	1	0	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	ADC	RO	1	ADC Module Present. When set, indicates that the ADC module is present.
15:12	MINSYSDIV	RO	0x3	System Clock Divider. Minimum 4-bit divider value for system clock. The reset value is hardware-dependent. See the <b>RCC</b> register for how to change the system clock divisor using the <code>SYSDIV</code> bit.  Value Description 0x3 Specifies a 50-MHz CPU clock with a PLL divider of 4.
11:10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:8	MAXADCSPD	RO	0x2	Max ADC Speed. This field indicates the maximum rate at which the ADC samples data.  Value Description 0x2 500K samples/second
7	MPU	RO	1	MPU Present. When set, indicates that the Cortex-M3 Memory Protection Unit (MPU) module is present. See the ARM Cortex-M3 Technical Reference Manual for details on the MPU.
6	HIB	RO	1	Hibernation Module Present. When set, indicates that the Hibernation module is present.
5	TEMPSNS	RO	1	Temp Sensor Present. When set, indicates that the on-chip temperature sensor is present.

Bit/Field	Name	Type	Reset	Description
4	PLL	RO	1	PLL Present. When set, indicates that the on-chip Phase Locked Loop (PLL) is present.
3	WDT	RO	1	Watchdog Timer Present. When set, indicates that a watchdog timer is present.
2	SWO	RO	1	SWO Trace Port Present. When set, indicates that the Serial Wire Output (SWO) trace port is present.
1	SWD	RO	1	SWD Present. When set, indicates that the Serial Wire Debugger (SWD) is present.
0	JTAG	RO	1	JTAG Present. When set, indicates that the JTAG debugger interface is present.

## Register 17: Device Capabilities 2 (DC2), offset 0x014

This register is predefined by the part and can be used to verify features.

### Device Capabilities 2 (DC2)

Base 0x400F.E000

Offset 0x014

Type RO, reset 0x000F.5017

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved												TIMER3	TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	I2C1	reserved	I2C0	reserved							SSI0	reserved	UART2	UART1	UART0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	1	0	1	0	0	0	0	0	0	0	1	0	1	1	1

Bit/Field	Name	Type	Reset	Description
31:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	RO	1	Timer 3 Present. When set, indicates that General-Purpose Timer module 3 is present.
18	TIMER2	RO	1	Timer 2 Present. When set, indicates that General-Purpose Timer module 2 is present.
17	TIMER1	RO	1	Timer 1 Present. When set, indicates that General-Purpose Timer module 1 is present.
16	TIMER0	RO	1	Timer 0 Present. When set, indicates that General-Purpose Timer module 0 is present.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	RO	1	I2C Module 1 Present. When set, indicates that I2C module 1 is present.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	RO	1	I2C Module 0 Present. When set, indicates that I2C module 0 is present.
11:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SSI0	RO	1	SSI0 Present. When set, indicates that SSI module 0 is present.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	RO	1	UART2 Present. When set, indicates that UART module 2 is present.
1	UART1	RO	1	UART1 Present. When set, indicates that UART module 1 is present.

Bit/Field	Name	Type	Reset	Description
0	UART0	RO	1	UART0 Present. When set, indicates that UART module 0 is present.

## Register 18: Device Capabilities 3 (DC3), offset 0x018

This register is predefined by the part and can be used to verify features.

### Device Capabilities 3 (DC3)

Base 0x400F.E000  
 Offset 0x018  
 Type RO, reset 0xBFFF.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	32KHZ	reserved	CCP5	CCP4	CCP3	CCP2	CCP1	CCP0	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	32KHZ	RO	1	32KHz Input Clock Available. When set, indicates an even CCP pin is present and can be used as a 32-KHz input clock.
30	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
29	CCP5	RO	1	CCP5 Pin Present. When set, indicates that Capture/Compare/PWM pin 5 is present.
28	CCP4	RO	1	CCP4 Pin Present. When set, indicates that Capture/Compare/PWM pin 4 is present.
27	CCP3	RO	1	CCP3 Pin Present. When set, indicates that Capture/Compare/PWM pin 3 is present.
26	CCP2	RO	1	CCP2 Pin Present. When set, indicates that Capture/Compare/PWM pin 2 is present.
25	CCP1	RO	1	CCP1 Pin Present. When set, indicates that Capture/Compare/PWM pin 1 is present.
24	CCP0	RO	1	CCP0 Pin Present. When set, indicates that Capture/Compare/PWM pin 0 is present.
23	ADC7	RO	1	ADC7 Pin Present. When set, indicates that ADC pin 7 is present.
22	ADC6	RO	1	ADC6 Pin Present. When set, indicates that ADC pin 6 is present.
21	ADC5	RO	1	ADC5 Pin Present. When set, indicates that ADC pin 5 is present.
20	ADC4	RO	1	ADC4 Pin Present. When set, indicates that ADC pin 4 is present.
19	ADC3	RO	1	ADC3 Pin Present. When set, indicates that ADC pin 3 is present.
18	ADC2	RO	1	ADC2 Pin Present. When set, indicates that ADC pin 2 is present.
17	ADC1	RO	1	ADC1 Pin Present. When set, indicates that ADC pin 1 is present.
16	ADC0	RO	1	ADC0 Pin Present. When set, indicates that ADC pin 0 is present.



Bit/Field	Name	Type	Reset	Description
15:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 19: Device Capabilities 4 (DC4), offset 0x01C

This register is predefined by the part and can be used to verify features.

### Device Capabilities 4 (DC4)

Base 0x400F.E000  
 Offset 0x01C  
 Type RO, reset 0x0000.301F

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved		UDMA	ROM	reserved								GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	1	1	0	0	0	0	0	0	0	1	1	1	1	1	

Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	RO	1	Micro-DMA is present
12	ROM	RO	1	Internal Code ROM is present
11:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	GPIOE	RO	1	GPIO Port E Present. When set, indicates that GPIO Port E is present.
3	GPIOD	RO	1	GPIO Port D Present. When set, indicates that GPIO Port D is present.
2	GPIOC	RO	1	GPIO Port C Present. When set, indicates that GPIO Port C is present.
1	GPIOB	RO	1	GPIO Port B Present. When set, indicates that GPIO Port B is present.
0	GPIOA	RO	1	GPIO Port A Present. When set, indicates that GPIO Port A is present.

## Register 20: Device Capabilities 5 (DC5), offset 0x020

This register is predefined by the part and can be used to verify features.

### Device Capabilities 5 (DC5)

Base 0x400F.E000

Offset 0x020

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 21: Device Capabilities 6 (DC6), offset 0x024

This register is predefined by the part and can be used to verify features.

### Device Capabilities 6 (DC6)

Base 0x400F.E000

Offset 0x024

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 22: Device Capabilities 7 (DC7), offset 0x028

This register is predefined by the part and can be used to verify uDMA channel features.

### Device Capabilities 7 (DC7)

Base 0x400F.E000  
Offset 0x028  
Type RO, reset 0x00C0.0F00

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved								UART1_TX	UART1_RX	reserved					
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				SSI0_TX	SSI0_RX	UART0_TX	UART0_RX	reserved							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23	UART1_TX	RO	1	UART1 TX on uDMA Ch23. When set, indicates uDMA channel 23 is available and connected to the transmit path of UART module 1.
22	UART1_RX	RO	1	UART1 RX on uDMA Ch22. When set, indicates uDMA channel 22 is available and connected to the receive path of UART module 1.
21:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	SSI0_TX	RO	1	SSI0 TX on uDMA Ch11. When set, indicates uDMA channel 11 is available and connected to the transmit path of SSI module 0.
10	SSI0_RX	RO	1	SSI0 RX on uDMA Ch10. When set, indicates uDMA channel 10 is available and connected to the receive path of SSI module 0.
9	UART0_TX	RO	1	UART0 TX on uDMA Ch9. When set, indicates uDMA channel 9 is available and connected to the transmit path of UART module 0.
8	UART0_RX	RO	1	UART0 RX on uDMA Ch8. When set, indicates uDMA channel 8 is available and connected to the receive path of UART module 0.
7:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

### Register 23: Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100

This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled (saving power). If the unit is unclocked, reads or writes to the unit will generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional units are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or units to control. This is to assure reasonable code compatibility with other family and future parts. **RCGC0** is the clock configuration register for running operation, **SCGC0** for Sleep operation, and **DCGC0** for Deep-Sleep operation. Setting the **ACG** bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

#### Run Mode Clock Gating Control Register 0 (RCGC0)

Base 0x400F.E000  
 Offset 0x100  
 Type R/W, reset 0x00000040

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															ADC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved						MAXADCSPD	reserved	HIB	reserved	WDT	reserved				
Type	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	R/W	RO	RO	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description								
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								
16	ADC	R/W	0	ADC0 Clock Gating Control. This bit controls the clock gating for SAR ADC module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, a read or write to the unit generates a bus fault.								
15:10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								
9:8	MAXADCSPD	R/W	0	ADC Sample Speed. This field sets the rate at which the ADC samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADCSPD bit as follows:  <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x2</td> <td>500K samples/second</td> </tr> <tr> <td>0x1</td> <td>250K samples/second</td> </tr> <tr> <td>0x0</td> <td>125K samples/second</td> </tr> </tbody> </table>	Value	Description	0x2	500K samples/second	0x1	250K samples/second	0x0	125K samples/second
Value	Description											
0x2	500K samples/second											
0x1	250K samples/second											
0x0	125K samples/second											
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								

---

Bit/Field	Name	Type	Reset	Description
6	HIB	R/W	0	HIB Clock Gating Control. This bit controls the clock gating for the Hibernation module. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled.
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT	R/W	0	WDT Clock Gating Control. This bit controls the clock gating for the WDT module. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, a read or write to the unit generates a bus fault.
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 24: Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110

This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled (saving power). If the unit is unclocked, reads or writes to the unit will generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional units are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or units to control. This is to assure reasonable code compatibility with other family and future parts. **RCGC0** is the clock configuration register for running operation, **SCGC0** for Sleep operation, and **DCGC0** for Deep-Sleep operation. Setting the **ACG** bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

### Sleep Mode Clock Gating Control Register 0 (SCGC0)

Base 0x400F.E000  
Offset 0x110  
Type R/W, reset 0x00000040

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															ADC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved						MAXADCSPD	reserved	HIB	reserved	WDT	reserved				
Type	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	R/W	RO	RO	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	ADC	R/W	0	ADC0 Clock Gating Control. This bit controls the clock gating for general SAR ADC module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, a read or write to the unit generates a bus fault.
15:10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:8	MAXADCSPD	R/W	0	ADC Sample Speed. This field sets the rate at which the ADC samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the <code>MAXADCSPD</code> bit as follows:
				Value Description
				0x2 500K samples/second
				0x1 250K samples/second
				0x0 125K samples/second
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.



---

Bit/Field	Name	Type	Reset	Description
6	HIB	R/W	0	HIB Clock Gating Control. This bit controls the clock gating for the Hibernation module. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled.
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT	R/W	0	WDT Clock Gating Control. This bit controls the clock gating for the WDT module. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, a read or write to the unit generates a bus fault.
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 25: Deep Sleep Mode Clock Gating Control Register 0 (DCGC0), offset 0x120

This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled (saving power). If the unit is unlocked, reads or writes to the unit will generate a bus fault. The reset state of these bits is 0 (unlocked) unless otherwise noted, so that all functional units are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or units to control. This is to assure reasonable code compatibility with other family and future parts. **RCGC0** is the clock configuration register for running operation, **SCGC0** for Sleep operation, and **DCGC0** for Deep-Sleep operation. Setting the **ACG** bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

### Deep Sleep Mode Clock Gating Control Register 0 (DCGC0)

Base 0x400F.E000  
Offset 0x120  
Type R/W, reset 0x00000040

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															ADC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved						MAXADCSPD	reserved	HIB	reserved	WDT	reserved				
Type	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	R/W	RO	RO	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	ADC	R/W	0	ADC0 Clock Gating Control. This bit controls the clock gating for general SAR ADC module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, a read or write to the unit generates a bus fault.
15:10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:8	MAXADCSPD	R/W	0	ADC Sample Speed. This field sets the rate at which the ADC samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the <code>MAXADCSPD</code> bit as follows:
				Value Description
				0x2 500K samples/second
				0x1 250K samples/second
				0x0 125K samples/second
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

---

Bit/Field	Name	Type	Reset	Description
6	HIB	R/W	0	HIB Clock Gating Control. This bit controls the clock gating for the Hibernation module. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled.
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT	R/W	0	WDT Clock Gating Control. This bit controls the clock gating for the WDT module. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, a read or write to the unit generates a bus fault.
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 26: Run Mode Clock Gating Control Register 1 (RCGC1), offset 0x104

This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled (saving power). If the unit is unclocked, reads or writes to the unit will generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional units are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or units to control. This is to assure reasonable code compatibility with other family and future parts. **RCGC1** is the clock configuration register for running operation, **SCGC1** for Sleep operation, and **DCGC1** for Deep-Sleep operation. Setting the **ACG** bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

### Run Mode Clock Gating Control Register 1 (RCGC1)

Base 0x400F.E000

Offset 0x104

Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved												TIMER3	TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	I2C1	reserved	I2C0	reserved							SSI0	reserved	UART2	UART1	UART0
Type	RO	R/W	RO	R/W	RO	RO	RO	RO	RO	RO	RO	R/W	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	R/W	0	Timer 3 Clock Gating Control. This bit controls the clock gating for General-Purpose Timer module 3. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
18	TIMER2	R/W	0	Timer 2 Clock Gating Control. This bit controls the clock gating for General-Purpose Timer module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
17	TIMER1	R/W	0	Timer 1 Clock Gating Control. This bit controls the clock gating for General-Purpose Timer module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control. This bit controls the clock gating for General-Purpose Timer module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
14	I2C1	R/W	0	I2C1 Clock Gating Control. This bit controls the clock gating for I2C module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Clock Gating Control. This bit controls the clock gating for I2C module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
11:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SSI0	R/W	0	SSI0 Clock Gating Control. This bit controls the clock gating for SSI module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	R/W	0	UART2 Clock Gating Control. This bit controls the clock gating for UART module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
1	UART1	R/W	0	UART1 Clock Gating Control. This bit controls the clock gating for UART module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
0	UART0	R/W	0	UART0 Clock Gating Control. This bit controls the clock gating for UART module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

## Register 27: Sleep Mode Clock Gating Control Register 1 (SCGC1), offset 0x114

This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled (saving power). If the unit is unlocked, reads or writes to the unit will generate a bus fault. The reset state of these bits is 0 (unlocked) unless otherwise noted, so that all functional units are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or units to control. This is to assure reasonable code compatibility with other family and future parts. **RCGC1** is the clock configuration register for running operation, **SCGC1** for Sleep operation, and **DCGC1** for Deep-Sleep operation. Setting the **ACG** bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

### Sleep Mode Clock Gating Control Register 1 (SCGC1)

Base 0x400F.E000  
Offset 0x114  
Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved												TIMER3	TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	I2C1	reserved	I2C0	reserved							SSI0	reserved	UART2	UART1	UART0
Type	RO	R/W	RO	R/W	RO	RO	RO	RO	RO	RO	RO	R/W	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	R/W	0	Timer 3 Clock Gating Control. This bit controls the clock gating for General-Purpose Timer module 3. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
18	TIMER2	R/W	0	Timer 2 Clock Gating Control. This bit controls the clock gating for General-Purpose Timer module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
17	TIMER1	R/W	0	Timer 1 Clock Gating Control. This bit controls the clock gating for General-Purpose Timer module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control. This bit controls the clock gating for General-Purpose Timer module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
14	I2C1	R/W	0	I2C1 Clock Gating Control. This bit controls the clock gating for I2C module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Clock Gating Control. This bit controls the clock gating for I2C module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
11:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SSI0	R/W	0	SSI0 Clock Gating Control. This bit controls the clock gating for SSI module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	R/W	0	UART2 Clock Gating Control. This bit controls the clock gating for UART module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
1	UART1	R/W	0	UART1 Clock Gating Control. This bit controls the clock gating for UART module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
0	UART0	R/W	0	UART0 Clock Gating Control. This bit controls the clock gating for UART module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

## Register 28: Deep Sleep Mode Clock Gating Control Register 1 (DCGC1), offset 0x124

This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled (saving power). If the unit is unlocked, reads or writes to the unit will generate a bus fault. The reset state of these bits is 0 (unlocked) unless otherwise noted, so that all functional units are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or units to control. This is to assure reasonable code compatibility with other family and future parts. **RCGC1** is the clock configuration register for running operation, **SCGC1** for Sleep operation, and **DCGC1** for Deep-Sleep operation. Setting the **ACG** bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

### Deep Sleep Mode Clock Gating Control Register 1 (DCGC1)

Base 0x400F.E000  
Offset 0x124  
Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved												TIMER3	TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	I2C1	reserved	I2C0	reserved							SSI0	reserved	UART2	UART1	UART0
Type	RO	R/W	RO	R/W	RO	RO	RO	RO	RO	RO	RO	R/W	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	R/W	0	Timer 3 Clock Gating Control. This bit controls the clock gating for General-Purpose Timer module 3. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
18	TIMER2	R/W	0	Timer 2 Clock Gating Control. This bit controls the clock gating for General-Purpose Timer module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
17	TIMER1	R/W	0	Timer 1 Clock Gating Control. This bit controls the clock gating for General-Purpose Timer module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control. This bit controls the clock gating for General-Purpose Timer module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.



Bit/Field	Name	Type	Reset	Description
14	I2C1	R/W	0	I2C1 Clock Gating Control. This bit controls the clock gating for I2C module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Clock Gating Control. This bit controls the clock gating for I2C module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
11:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SSI0	R/W	0	SSI0 Clock Gating Control. This bit controls the clock gating for SSI module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	R/W	0	UART2 Clock Gating Control. This bit controls the clock gating for UART module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
1	UART1	R/W	0	UART1 Clock Gating Control. This bit controls the clock gating for UART module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
0	UART0	R/W	0	UART0 Clock Gating Control. This bit controls the clock gating for UART module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

## Register 29: Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108

This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled (saving power). If the unit is unlocked, reads or writes to the unit will generate a bus fault. The reset state of these bits is 0 (unlocked) unless otherwise noted, so that all functional units are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or units to control. This is to assure reasonable code compatibility with other family and future parts. **RCGC2** is the clock configuration register for running operation, **SCGC2** for Sleep operation, and **DCGC2** for Deep-Sleep operation. Setting the **ACG** bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

### Run Mode Clock Gating Control Register 2 (RCGC2)

Base 0x400F.E000

Offset 0x108

Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
	reserved																			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
	reserved		UDMA	reserved												GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	R/W	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	R/W	0	UDMA Clock Gating Control. This bit controls the clock gating for Port H. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
12:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	GPIOE	R/W	0	Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
3	GPIOD	R/W	0	Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
2	GPIOC	R/W	0	Port C Clock Gating Control. This bit controls the clock gating for Port C. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

---

Bit/Field	Name	Type	Reset	Description
1	GPIOB	R/W	0	Port B Clock Gating Control. This bit controls the clock gating for Port B. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
0	GPIOA	R/W	0	Port A Clock Gating Control. This bit controls the clock gating for Port A. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

## Register 30: Sleep Mode Clock Gating Control Register 2 (SCGC2), offset 0x118

This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled (saving power). If the unit is unlocked, reads or writes to the unit will generate a bus fault. The reset state of these bits is 0 (unlocked) unless otherwise noted, so that all functional units are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or units to control. This is to assure reasonable code compatibility with other family and future parts. **RCGC2** is the clock configuration register for running operation, **SCGC2** for Sleep operation, and **DCGC2** for Deep-Sleep operation. Setting the **ACG** bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

### Sleep Mode Clock Gating Control Register 2 (SCGC2)

Base 0x400F.E000  
Offset 0x118  
Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved		UDMA	reserved									GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	R/W	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	R/W	0	UDMA Clock Gating Control. This bit controls the clock gating for Port H. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
12:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	GPIOE	R/W	0	Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
3	GPIOD	R/W	0	Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
2	GPIOC	R/W	0	Port C Clock Gating Control. This bit controls the clock gating for Port C. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

Bit/Field	Name	Type	Reset	Description
1	GPIOB	R/W	0	Port B Clock Gating Control. This bit controls the clock gating for Port B. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
0	GPIOA	R/W	0	Port A Clock Gating Control. This bit controls the clock gating for Port A. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

## Register 31: Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128

This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled (saving power). If the unit is unlocked, reads or writes to the unit will generate a bus fault. The reset state of these bits is 0 (unlocked) unless otherwise noted, so that all functional units are disabled. It is the responsibility of software to enable the ports necessary for the application. Note that these registers may contain more bits than there are interfaces, functions, or units to control. This is to assure reasonable code compatibility with other family and future parts. **RCGC2** is the clock configuration register for running operation, **SCGC2** for Sleep operation, and **DCGC2** for Deep-Sleep operation. Setting the **ACG** bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

### Deep Sleep Mode Clock Gating Control Register 2 (DCGC2)

Base 0x400F.E000  
Offset 0x128  
Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved		UDMA	reserved									GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	R/W	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	R/W	0	UDMA Clock Gating Control. This bit controls the clock gating for Port H. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
12:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	GPIOE	R/W	0	Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
3	GPIOD	R/W	0	Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
2	GPIOC	R/W	0	Port C Clock Gating Control. This bit controls the clock gating for Port C. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

Bit/Field	Name	Type	Reset	Description
1	GPIOB	R/W	0	Port B Clock Gating Control. This bit controls the clock gating for Port B. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
0	GPIOA	R/W	0	Port A Clock Gating Control. This bit controls the clock gating for Port A. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

### Register 32: Software Reset Control 0 (SRCR0), offset 0x040

Writes to this register are masked by the bits in the **Device Capabilities 1 (DC1)** register.

#### Software Reset Control 0 (SRCR0)

Base 0x400F.E000  
 Offset 0x040  
 Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															ADC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved							HIB	reserved		WDT	reserved				
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	RO	RO	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	ADC	R/W	0	ADC0 Reset Control. Reset control for SAR ADC module 0.
15:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	HIB	R/W	0	HIB Reset Control. Reset control for the Hibernation module.
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT	R/W	0	WDT Reset Control. Reset control for Watchdog unit.
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.



**Register 33: Software Reset Control 1 (SRCR1), offset 0x044**Writes to this register are masked by the bits in the **Device Capabilities 2 (DC2)** register.

## Software Reset Control 1 (SRCR1)

Base 0x400F.E000

Offset 0x044

Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved												TIMER3	TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	I2C1	reserved	I2C0	reserved							SSI0	reserved	UART2	UART1	UART0
Type	RO	R/W	RO	R/W	RO	RO	RO	RO	RO	RO	RO	R/W	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	R/W	0	Timer 3 Reset Control. Reset control for General-Purpose Timer module 3.
18	TIMER2	R/W	0	Timer 2 Reset Control. Reset control for General-Purpose Timer module 2.
17	TIMER1	R/W	0	Timer 1 Reset Control. Reset control for General-Purpose Timer module 1.
16	TIMER0	R/W	0	Timer 0 Reset Control. Reset control for General-Purpose Timer module 0.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	R/W	0	I2C1 Reset Control. Reset control for I2C unit 1.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Reset Control. Reset control for I2C unit 0.
11:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SSI0	R/W	0	SSI0 Reset Control. Reset control for SSI unit 0.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	R/W	0	UART2 Reset Control. Reset control for UART unit 2.
1	UART1	R/W	0	UART1 Reset Control. Reset control for UART unit 1.

Bit/Field	Name	Type	Reset	Description
0	UART0	R/W	0	UART0 Reset Control. Reset control for UART unit 0.

**Register 34: Software Reset Control 2 (SRCR2), offset 0x048**Writes to this register are masked by the bits in the **Device Capabilities 4 (DC4)** register.

## Software Reset Control 2 (SRCR2)

Base 0x400F.E000

Offset 0x048

Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved		UDMA	reserved									GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	R/W	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	R/W	0	UDMA Reset Control. Reset control for uDMA unit.
12:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	GPIOE	R/W	0	Port E Reset Control. Reset control for GPIO Port E.
3	GPIOD	R/W	0	Port D Reset Control. Reset control for GPIO Port D.
2	GPIOC	R/W	0	Port C Reset Control. Reset control for GPIO Port C.
1	GPIOB	R/W	0	Port B Reset Control. Reset control for GPIO Port B.
0	GPIOA	R/W	0	Port A Reset Control. Reset control for GPIO Port A.

## 7 Hibernation Module

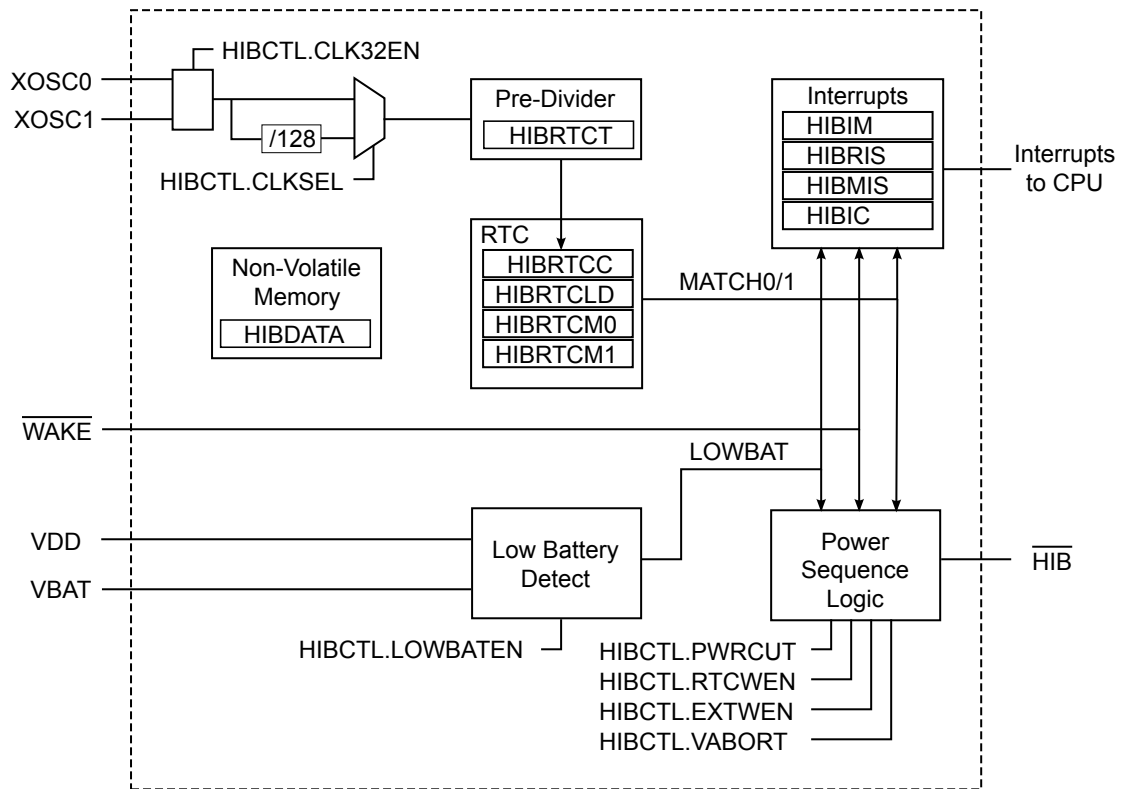
The Hibernation Module manages removal and restoration of power to the rest of the microcontroller to provide a means for reducing power consumption. When the processor and peripherals are idle, power can be completely removed with only the Hibernation Module remaining powered. Power can be restored based on an external signal, or at a certain time using the built-in real-time clock (RTC). The Hibernation module can be independently supplied from a battery or an auxiliary power supply.

The Hibernation module has the following features:

- Power-switching logic to discrete external regulator
- Dedicated pin for waking from an external signal
- Low-battery detection, signaling, and interrupt generation
- 32-bit real-time counter (RTC)
- Two 32-bit RTC match registers for timed wake-up and interrupt generation
- Clock source from a 32.768-kHz external oscillator or a 4.194304-MHz crystal
- RTC predivider trim for making fine adjustments to the clock rate
- 64 32-bit words of non-volatile memory
- Programmable interrupts for RTC match, external wake, and low battery events

## 7.1 Block Diagram

Figure 7-1. Hibernation Module Block Diagram



## 7.2 Functional Description

The Hibernation module controls the power to the processor with an enable signal ( $\overline{HIB}$ ) that signals an external voltage regulator to turn off. The Hibernation module power is determined dynamically. The supply voltage of the Hibernation module is the larger of the main voltage source (VDD) or the battery/auxilliary voltage source (VBAT). A voting circuit indicates the larger and an internal power switch selects the appropriate voltage source. The Hibernation module also has a separate clock source to maintain a real-time clock (RTC). Once in hibernation, the module signals an external voltage regulator to turn back on the power when an external pin ( $\overline{WAKE}$ ) is asserted, or when the internal RTC reaches a certain value. The Hibernation module can also detect when the battery voltage is low, and optionally prevent hibernation when this occurs.

Power-up from a power cut to code execution is defined as the regulator turn-on time (specified at  $t_{HIB\_TO\_VDD}$  maximum) plus the normal chip POR (see "Hibernation Module" on page 505).

### 7.2.1 Register Access Timing

Because the Hibernation module has an independent clocking domain, certain registers must be written only with a timing gap between accesses. The delay time is  $t_{HIB\_REG\_WRITE}$ , therefore software must guarantee that a delay of  $t_{HIB\_REG\_WRITE}$  is inserted between back-to-back writes to certain Hibernation registers, or between a write followed by a read to those same registers. There is no

restriction on timing for back-to-back reads from the Hibernation module. Software may make use of the `WRC` bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll **HIBCTL** for `WRC=1` prior to accessing any affected register. The following registers are subject to this timing restriction:

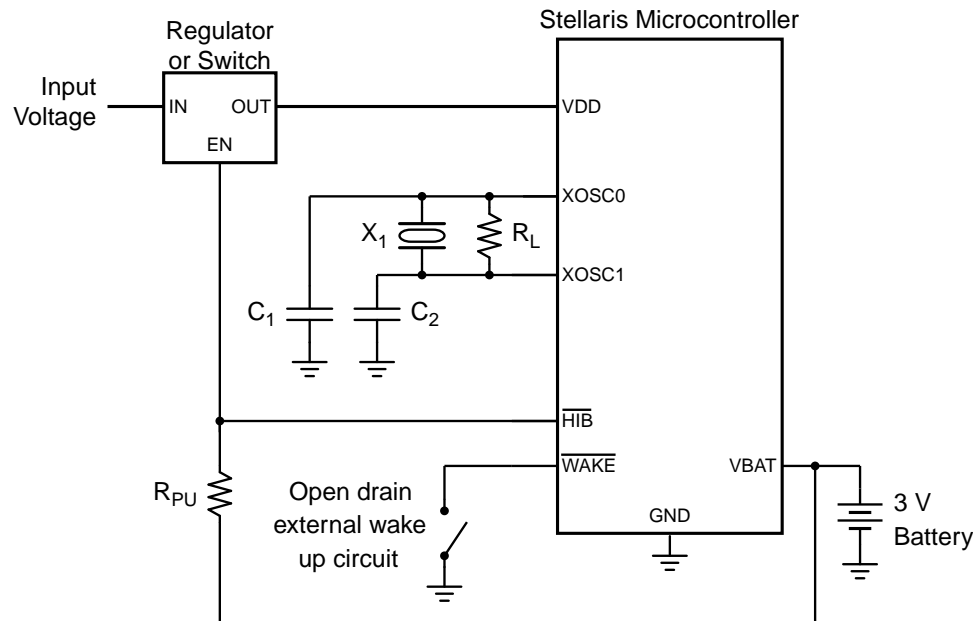
- **Hibernation RTC Counter (HIBRTCC)**
- **Hibernation RTC Match 0 (HIBRTCM0)**
- **Hibernation RTC Match 1 (HIBRTCM1)**
- **Hibernation RTC Load (HIBRTCLD)**
- **Hibernation RTC Trim (HIBRTCT)**
- **Hibernation Data (HIBDATA)**

## 7.2.2 Clock Source

The Hibernation module must be clocked by an external source, even if the RTC feature will not be used. An external oscillator or crystal can be used for this purpose. To use a crystal, a 4.194304-MHz crystal is connected to the `XOSC0` and `XOSC1` pins. This clock signal is divided by 128 internally to produce the 32.768-kHz clock reference. To use a more precise clock source, a 32.768-kHz oscillator can be connected to the `XOSC0` pin. See Figure 7-2 on page 127 and Figure 7-3 on page 128. Note that these diagrams only show the connection to the Hibernation pins and not to the full system. See “Hibernation Module” on page 505 for specific values.

The clock source is enabled by setting the `CLK32EN` bit of the **HIBCTL** register. The type of clock source is selected by setting the `CLKSEL` bit to 0 for a 4.194304-MHz clock source, and to 1 for a 32.768-kHz clock source. If the bit is set to 0, the input clock is divided by 128, resulting in a 32.768-kHz clock source. If a crystal is used for the clock source, the software must leave a delay of `tXOSC_SETTLE` after setting the `CLK32EN` bit and before any other accesses to the Hibernation module registers. The delay allows the crystal to power up and stabilize. If an oscillator is used for the clock source, no delay is needed.

Figure 7-2. Clock Source Using Crystal

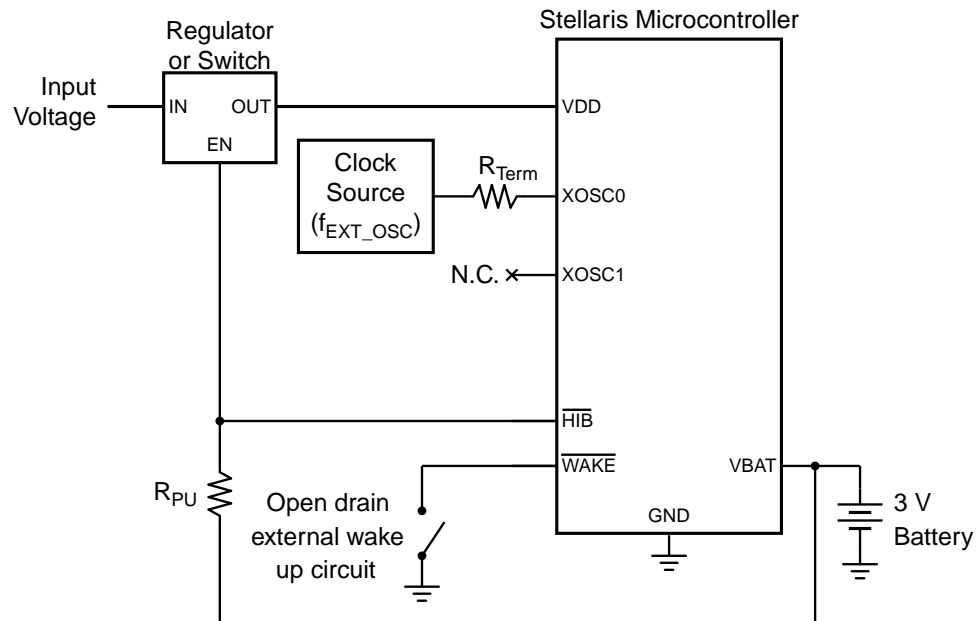


**Note:** R<sub>TERM</sub> = Optional series termination resistor.

R<sub>PU</sub> = Pull-up resistor (1 M<sup>1</sup>/<sub>2</sub>).

See "Hibernation Module" on page 505 for specific parameter values.

Figure 7-3. Clock Source Using Dedicated Oscillator



**Note:**  $X_1$  = Crystal frequency is  $f_{XOSC\_XTAL}$ .

$R_L$  = Load resistor is  $R_{XOSC\_LOAD}$ .

$C_{1,2}$  = Capacitor value derived from crystal vendor load capacitance specifications.

$R_{PU}$  = Pull-up resistor (1 M $\frac{1}{2}$ ).

See "Hibernation Module" on page 505 for specific parameter values.

### 7.2.3 Battery Management

The Hibernation module can be independently powered by a battery or an auxiliary power source. The module can monitor the voltage level of the battery and detect when the voltage drops below 2.35 V. When this happens, an interrupt can be generated. The module also can be configured so that it will not go into Hibernate mode if the battery voltage drops below this threshold. Battery voltage is not measured while in Hibernate mode.

**Important:** System level factors may affect the accuracy of the low battery detect circuit. The designer should consider battery type, discharge characteristics, and a test load during battery voltage measurements.

Note that the Hibernation module draws power from whichever source (VBAT or VDD) has the higher voltage. Therefore, it is important to design the circuit to ensure that VDD is higher than VBAT under nominal conditions or else the Hibernation module draws power from the battery even when VDD is available.

The Hibernation module can be configured to detect a low battery condition by setting the LOWBATEN bit of the HIBCTL register. In this configuration, the LOWBAT bit of the HIBRIS register will be set when the battery level is low. If the VABORT bit is also set, then the module is prevented from entering Hibernation mode when a low battery is detected. The module can also be configured to generate an interrupt for the low-battery condition (see "Interrupts and Status" on page 130).



## 7.2.4 Real-Time Clock

The Hibernation module includes a 32-bit counter that increments once per second with a proper clock source and configuration (see “Clock Source” on page 126). The 32.768-kHz clock signal is fed into a predivider register which counts down the 32.768-kHz clock ticks to achieve a once per second clock rate for the RTC. The rate can be adjusted to compensate for inaccuracies in the clock source by using the predivider trim register, **HIBRTCT**. This register has a nominal value of 0x7FFF, and is used for one second out of every 64 seconds to divide the input clock. This allows the software to make fine corrections to the clock rate by adjusting the predivider trim register up or down from 0x7FFF. The predivider trim should be adjusted up from 0x7FFF in order to slow down the RTC rate, and down from 0x7FFF in order to speed up the RTC rate.

The Hibernation module includes two 32-bit match registers that are compared to the value of the RTC counter. The match registers can be used to wake the processor from hibernation mode, or to generate an interrupt to the processor if it is not in hibernation.

The RTC must be enabled with the **RTCEN** bit of the **HIBCTL** register. The value of the RTC can be set at any time by writing to the **HIBRTCLD** register. The predivider trim can be adjusted by reading and writing the **HIBRTCT** register. The predivider uses this register once every 64 seconds to adjust the clock rate. The two match registers can be set by writing to the **HIBRTCM0** and **HIBRTCM1** registers. The RTC can be configured to generate interrupts by using the interrupt registers (see “Interrupts and Status” on page 130).

## 7.2.5 Non-Volatile Memory

The Hibernation module contains 64 32-bit words of memory which are retained during hibernation. This memory is powered from the battery or auxiliary power supply during hibernation. The processor software can save state information in this memory prior to hibernation, and can then recover the state upon waking. The non-volatile memory can be accessed through the **HIBDATA** registers.

## 7.2.6 Power Control

**Important:** The Hibernation Module requires special system implementation considerations since it is intended to power-down all other sections of its host device. The system power-supply distribution and interfaces of the system must be driven to 0 V<sub>DC</sub> or powered down with the same regulator controlled by  $\overline{\text{HIB}}$ . See “Hibernation Module” on page 505 for more details.

The Hibernation module controls power to the processor through the use of the  $\overline{\text{HIB}}$  pin, which is intended to be connected to the enable signal of the external regulator(s) providing 3.3 V and/or 2.5 V to the microcontroller. When the  $\overline{\text{HIB}}$  signal is asserted by the Hibernation module, the external regulator is turned off and no longer powers the microcontroller. The Hibernation module remains powered from the **V<sub>BAT</sub>** supply, which could be a battery or an auxiliary power source. Hibernation mode is initiated by the microcontroller setting the **HIBREQ** bit of the **HIBCTL** register. Prior to doing this, a wake-up condition must be configured, either from the external  $\overline{\text{WAKE}}$  pin, or by using an RTC match.

The Hibernation module is configured to wake from the external  $\overline{\text{WAKE}}$  pin by setting the **PINWEN** bit of the **HIBCTL** register. It is configured to wake from RTC match by setting the **RTCWEN** bit. Either one or both of these bits can be set prior to going into hibernation. The  $\overline{\text{WAKE}}$  pin includes a weak internal pull-up. Note that both the  $\overline{\text{HIB}}$  and  $\overline{\text{WAKE}}$  pins use the Hibernation module's internal power supply as the logic 1 reference.

When the Hibernation module wakes, the microcontroller will see a normal power-on reset. It can detect that the power-on was due to a wake from hibernation by examining the raw interrupt status

register (see “Interrupts and Status” on page 130) and by looking for state data in the non-volatile memory (see “Non-Volatile Memory” on page 129).

When the  $\overline{\text{HIB}}$  signal deasserts, enabling the external regulator, the external regulator must reach the operating voltage within  $t_{\text{HIB\_TO\_VDD}}$ .

### 7.2.7 Interrupts and Status

The Hibernation module can generate interrupts when the following conditions occur:

- Assertion of  $\overline{\text{WAKE}}$  pin
- RTC match
- Low battery detected

All of the interrupts are ORed together before being sent to the interrupt controller, so the Hibernation module can only generate a single interrupt request to the controller at any given time. The software interrupt handler can service multiple interrupt events by reading the **HIBMIS** register. Software can also read the status of the Hibernation module at any time by reading the **HIBRIS** register which shows all of the pending events. This register can be used at power-on to see if a wake condition is pending, which indicates to the software that a hibernation wake occurred.

The events that can trigger an interrupt are configured by setting the appropriate bits in the **HIBIM** register. Pending interrupts can be cleared by writing the corresponding bit in the **HIBIC** register.

## 7.3 Initialization and Configuration

The Hibernation module can be set in several different configurations. The following sections show the recommended programming sequence for various scenarios. The examples below assume that a 32.768-kHz oscillator is used, and thus always show bit 2 (**CLKSEL**) of the **HIBCTL** register set to 1. If a 4.194304-MHz crystal is used instead, then the **CLKSEL** bit remains cleared. Because the Hibernation module runs at 32 kHz and is asynchronous to the rest of the system, software must allow a delay of  $t_{\text{HIB\_REG\_WRITE}}$  after writes to certain registers (see “Register Access Timing” on page 125). The registers that require a delay are listed in a note in “Register Map” on page 132 as well as in each register description.

### 7.3.1 Initialization

The clock source must be enabled first, even if the RTC will not be used. If a 4.194304-MHz crystal is used, perform the following steps:

1. Write 0x40 to the **HIBCTL** register at offset 0x10 to enable the crystal and select the divide-by-128 input path.
2. Wait for a time of  $t_{\text{XOSC\_SETTLE}}$  for the crystal to power up and stabilize before performing any other operations with the Hibernation module.

If a 32.678-kHz oscillator is used, then perform the following steps:

1. Write 0x44 to the **HIBCTL** register at offset 0x10 to enable the oscillator input.
2. No delay is necessary.

The above is only necessary when the entire system is initialized for the first time. If the processor is powered due to a wake from hibernation, then the Hibernation module has already been powered

up and the above steps are not necessary. The software can detect that the Hibernation module and clock are already powered by examining the `CLK32EN` bit of the **HIBCTL** register.

### 7.3.2 RTC Match Functionality (No Hibernation)

Use the following steps to implement the RTC match functionality of the Hibernation module:

1. Write the required RTC match value to one of the **HIBRTCMn** registers at offset 0x004 or 0x008.
2. Write the required RTC load value to the **HIBRTCLD** register at offset 0x00C.
3. Set the required RTC match interrupt mask in the `RTCALTO` and `RTCALTO1` bits (bits 1:0) in the **HIBIM** register at offset 0x014.
4. Write 0x0000.0041 to the **HIBCTL** register at offset 0x010 to enable the RTC to begin counting.

### 7.3.3 RTC Match/Wake-Up from Hibernation

Use the following steps to implement the RTC match and wake-up functionality of the Hibernation module:

1. Write the required RTC match value to the **HIBRTCMn** registers at offset 0x004 or 0x008.
2. Write the required RTC load value to the **HIBRTCLD** register at offset 0x00C.
3. Write any data to be retained during power cut to the **HIBDATA** register at offsets 0x030-0x12C.
4. Set the RTC Match Wake-Up and start the hibernation sequence by writing 0x0000.004F to the **HIBCTL** register at offset 0x010.

### 7.3.4 External Wake-Up from Hibernation

Use the following steps to implement the Hibernation module with the external `WAKE` pin as the wake-up source for the microcontroller:

1. Write any data to be retained during power cut to the **HIBDATA** register at offsets 0x030-0x12C.
2. Enable the external wake and start the hibernation sequence by writing 0x0000.0056 to the **HIBCTL** register at offset 0x010.

### 7.3.5 RTC/External Wake-Up from Hibernation

1. Write the required RTC match value to the **HIBRTCMn** registers at offset 0x004 or 0x008.
2. Write the required RTC load value to the **HIBRTCLD** register at offset 0x00C.
3. Write any data to be retained during power cut to the **HIBDATA** register at offsets 0x030-0x12C.
4. Set the RTC Match/External Wake-Up and start the hibernation sequence by writing 0x0000.005F to the **HIBCTL** register at offset 0x010.

### 7.3.6 Register Reset

The Hibernation module handles resets according to the following conditions:

- Cold Reset

When the hibernation module has no externally applied voltage and detects a change to either VDD or VBAT, it resets all hibernation module registers to the value in Table 7-1 on page 132.

- **Reset During Hibernation Module Disable**

When the module has either not been enabled or has been disabled by software, the reset is passed through to the Hibernation module circuitry and the internal state of the module is reset.

- **Reset While HIB Module is in Hibernation Mode**

While in Hibernation mode, or while transitioning from Hibernation mode to run mode (leaving the power cut), the reset generated by the POR circuitry of the device is suppressed, and the state of the Hibernation module's registers is unaffected.

- **Reset While HIB Module is in Normal Mode**

While in normal mode (not hibernating), any reset is suppressed, and the content/state of the control and data registers is unaffected.

Software must initialize any control or data registers in this condition. Therefore, software is the only mechanism to enable or disable the oscillator and real-time clock operation, or to clear contents of the data memory. The only state that must be cleared by a reset operation while not in Hibernation mode is any state that prevents software from managing the interface.

## 7.4 Register Map

Table 7-1 on page 132 lists the Hibernation registers. All addresses given are relative to the Hibernation Module base address at 0x400F.C000.

**Note:** **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and require a delay of  $t_{\text{HIB\_REG\_WRITE}}$  between write accesses. See “Register Access Timing” on page 125.

**Table 7-1. Hibernation Module Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	HIBRTCC	RO	0x0000.0000	Hibernation RTC Counter	134
0x004	HIBRTCM0	R/W	0xFFFF.FFFF	Hibernation RTC Match 0	135
0x008	HIBRTCM1	R/W	0xFFFF.FFFF	Hibernation RTC Match 1	136
0x00C	HIBRTCLD	R/W	0xFFFF.FFFF	Hibernation RTC Load	137
0x010	HIBCTL	R/W	0x0000.0000	Hibernation Control	138
0x014	HIBIM	R/W	0x0000.0000	Hibernation Interrupt Mask	141
0x018	HIBRIS	RO	0x0000.0000	Hibernation Raw Interrupt Status	142
0x01C	HIBMIS	RO	0x0000.0000	Hibernation Masked Interrupt Status	143
0x020	HIBIC	R/W1C	0x0000.0000	Hibernation Interrupt Clear	144
0x024	HIBRTCT	R/W	0x0000.7FFF	Hibernation RTC Trim	145
0x030-0x12C	HIBDATA	R/W	0x0000.0000	Hibernation Data	146

## 7.5 Register Descriptions

The remainder of this section lists and describes the Hibernation module registers, in numerical order by address offset.

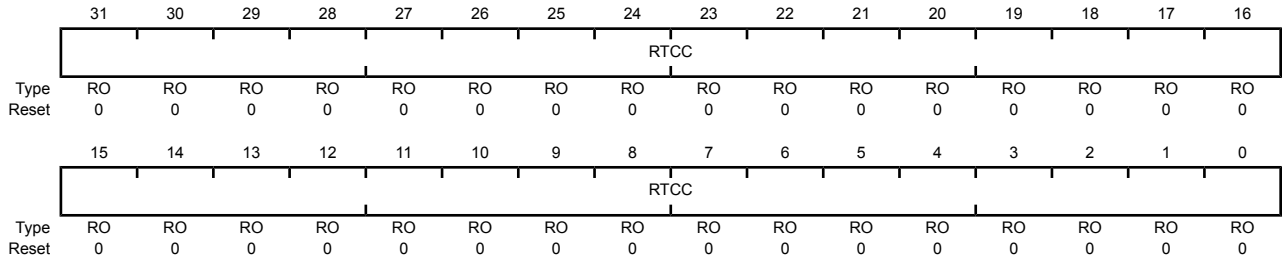
### Register 1: Hibernation RTC Counter (HIBRTCC), offset 0x000

This register is the current 32-bit value of the RTC counter.

**Note:** **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and require a delay of  $t_{\text{HIB\_REG\_WRITE}}$  between write accesses. See “Register Access Timing” on page 125.

#### Hibernation RTC Counter (HIBRTCC)

Base 0x400F.C000  
 Offset 0x000  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	RTCC	RO	0x0000.0000	RTC Counter

A read returns the 32-bit counter value. This register is read-only. To change the value, use the **HIBRTCLD** register.

**Register 2: Hibernation RTC Match 0 (HIBRTCM0), offset 0x004**

This register is the 32-bit match 0 register for the RTC counter.

**Note:** **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and require a delay of  $t_{\text{HIB\_REG\_WRITE}}$  between write accesses. See “Register Access Timing” on page 125.

**Hibernation RTC Match 0 (HIBRTCM0)**

Base 0x400F.C000

Offset 0x004

Type R/W, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RTCM0															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RTCM0															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:0	RTCM0	R/W	0xFFFF.FFFF	RTC Match 0

A write loads the value into the RTC match register.

A read returns the current match value.

### Register 3: Hibernation RTC Match 1 (HIBRTCM1), offset 0x008

This register is the 32-bit match 1 register for the RTC counter.

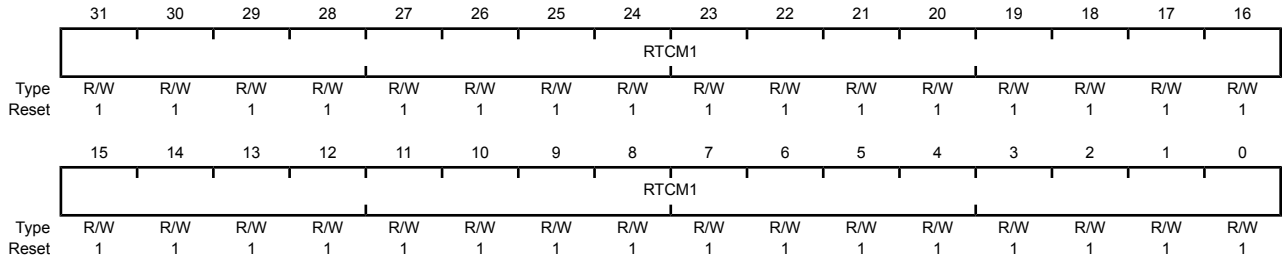
**Note:** HIBRTCC, HIBRTCM0, HIBRTCM1, HIBRTCLD, HIBRTCT, and HIBDATA are on the Hibernation module clock domain and require a delay of  $t_{HIB\_REG\_WRITE}$  between write accesses. See “Register Access Timing” on page 125.

#### Hibernation RTC Match 1 (HIBRTCM1)

Base 0x400F.C000

Offset 0x008

Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	RTCM1	R/W	0xFFFF.FFFF	RTC Match 1

A write loads the value into the RTC match register.

A read returns the current match value.



## Register 4: Hibernation RTC Load (HIBRTCLD), offset 0x00C

This register is the 32-bit value loaded into the RTC counter.

**Note:** **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and require a delay of  $t_{\text{HIB\_REG\_WRITE}}$  between write accesses. See “Register Access Timing” on page 125.

### Hibernation RTC Load (HIBRTCLD)

Base 0x400F.C000

Offset 0x00C

Type R/W, reset 0xFFFF.FFFF

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

Bit/Field	Name	Type	Reset	Description
31:0	RTCLD	R/W	0xFFFF.FFFF	RTC Load

A write loads the current value into the RTC counter (RTCC).

A read returns the 32-bit load value.

## Register 5: Hibernation Control (HIBCTL), offset 0x010

This register is the control register for the Hibernation module.

### Hibernation Control (HIBCTL)

Base 0x400F.C000  
 Offset 0x010  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WRC	reserved														
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								VABORT	CLK32EN	LOWBATEN	PINWEN	RTCWEN	CLKSEL	HIBREQ	RTCEN
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description				
31	WRC	RO	1	<p>Write Complete/Capable</p> <p>This bit indicates whether the hibernation module can receive a write operation.</p> <p>Value Description</p> <table border="0"> <tr> <td>0</td> <td>The interface is processing a prior write and is busy. Any write operation that is attempted while WRC is 0 results in undetermined behavior.</td> </tr> <tr> <td>1</td> <td>The interface is ready to accept a write.</td> </tr> </table> <p>Software must poll this bit between write requests and defer writes until WRC=1 to ensure proper operation.</p> <p>This difference may be exploited by software at reset time to detect which method of programming is appropriate: 0 = software delay loops required; 1 = WRC paced available.</p> <p>The bit name WRC means "Write Complete," which is the normal use of the bit (between write accesses). However, because the bit is set out-of-reset, the name can also mean "Write Capable" which simply indicates that the interface may be written to by software. This meaning also has more meaning to the out-of-reset sense.</p>	0	The interface is processing a prior write and is busy. Any write operation that is attempted while WRC is 0 results in undetermined behavior.	1	The interface is ready to accept a write.
0	The interface is processing a prior write and is busy. Any write operation that is attempted while WRC is 0 results in undetermined behavior.							
1	The interface is ready to accept a write.							
30:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.				
7	VABORT	R/W	0	<p>Power Cut Abort Enable</p> <p>Value Description</p> <table border="0"> <tr> <td>0</td> <td>Power cut occurs during a low-battery alert.</td> </tr> <tr> <td>1</td> <td>Power cut is aborted.</td> </tr> </table>	0	Power cut occurs during a low-battery alert.	1	Power cut is aborted.
0	Power cut occurs during a low-battery alert.							
1	Power cut is aborted.							

Bit/Field	Name	Type	Reset	Description
6	CLK32EN	R/W	0	<p>32-kHz Oscillator Enable</p> <p>Value Description</p> <p>0 Disabled</p> <p>1 Enabled</p> <p>This bit must be enabled to use the Hibernation module. If a crystal is used, then software should wait 20 ms after setting this bit to allow the crystal to power up and stabilize.</p>
5	LOWBATEN	R/W	0	<p>Low Battery Monitoring Enable</p> <p>Value Description</p> <p>0 Disabled</p> <p>1 Enabled</p> <p>When set, low battery voltage detection is enabled (VBAT &lt; 2.35 V).</p>
4	PINWEN	R/W	0	<p>External <math>\overline{\text{WAKE}}</math> Pin Enable</p> <p>Value Description</p> <p>0 Disabled</p> <p>1 Enabled</p> <p>When set, an external event on the <math>\overline{\text{WAKE}}</math> pin will re-power the device.</p>
3	RTCWEN	R/W	0	<p>RTC Wake-up Enable</p> <p>Value Description</p> <p>0 Disabled</p> <p>1 Enabled</p> <p>When set, an RTC match event (RTCM0 or RTCM1) will re-power the device based on the RTC counter value matching the corresponding match register 0 or 1.</p>
2	CLKSEL	R/W	0	<p>Hibernation Module Clock Select</p> <p>Value Description</p> <p>0 Use Divide by 128 output. Use this value for a 4-MHz crystal.</p> <p>1 Use raw output. Use this value for a 32-kHz oscillator.</p>
1	HIBREQ	R/W	0	<p>Hibernation Request</p> <p>Value Description</p> <p>0 Disabled</p> <p>1 Hibernation initiated</p> <p>After a wake-up event, this bit is cleared by hardware.</p>

Bit/Field	Name	Type	Reset	Description
0	RTCEN	R/W	0	RTC Timer Enable
				Value Description
				0 Disabled
				1 Enabled

## Register 6: Hibernation Interrupt Mask (HIBIM), offset 0x014

This register is the interrupt mask register for the Hibernation module interrupt sources.

### Hibernation Interrupt Mask (HIBIM)

Base 0x400F.C000

Offset 0x014

Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved												EXTW	LOWBAT	RTCALT1	RTCALT0	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

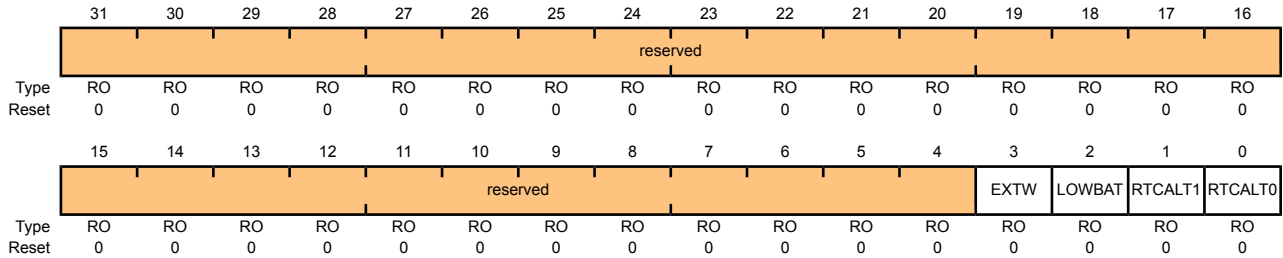
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	R/W	0	External Wake-Up Interrupt Mask  Value Description 0 Masked 1 Unmasked
2	LOWBAT	R/W	0	Low Battery Voltage Interrupt Mask  Value Description 0 Masked 1 Unmasked
1	RTCALT1	R/W	0	RTC Alert1 Interrupt Mask  Value Description 0 Masked 1 Unmasked
0	RTCALT0	R/W	0	RTC Alert0 Interrupt Mask  Value Description 0 Masked 1 Unmasked

### Register 7: Hibernation Raw Interrupt Status (HIBRIS), offset 0x018

This register is the raw interrupt status for the Hibernation module interrupt sources.

#### Hibernation Raw Interrupt Status (HIBRIS)

Base 0x400F.C000  
 Offset 0x018  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	RO	0	External Wake-Up Raw Interrupt Status
2	LOWBAT	RO	0	Low Battery Voltage Raw Interrupt Status
1	RTCALT1	RO	0	RTC Alert1 Raw Interrupt Status
0	RTCALT0	RO	0	RTC Alert0 Raw Interrupt Status

## Register 8: Hibernation Masked Interrupt Status (HIBMIS), offset 0x01C

This register is the masked interrupt status for the Hibernation module interrupt sources.

### Hibernation Masked Interrupt Status (HIBMIS)

Base 0x400F.C000

Offset 0x01C

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												EXTW	LOWBAT	RTCALT1	RTCALT0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	RO	0	External Wake-Up Masked Interrupt Status
2	LOWBAT	RO	0	Low Battery Voltage Masked Interrupt Status
1	RTCALT1	RO	0	RTC Alert1 Masked Interrupt Status
0	RTCALT0	RO	0	RTC Alert0 Masked Interrupt Status

### Register 9: Hibernation Interrupt Clear (HIBIC), offset 0x020

This register is the interrupt write-one-to-clear register for the Hibernation module interrupt sources.

#### Hibernation Interrupt Clear (HIBIC)

Base 0x400F.C000

Offset 0x020

Type R/W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												EXTW	LOWBAT	RTCALT1	RTCALT0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	R/W1C	0	External Wake-Up Masked Interrupt Clear Reads return an indeterminate value.
2	LOWBAT	R/W1C	0	Low Battery Voltage Masked Interrupt Clear Reads return an indeterminate value.
1	RTCALT1	R/W1C	0	RTC Alert1 Masked Interrupt Clear Reads return an indeterminate value.
0	RTCALT0	R/W1C	0	RTC Alert0 Masked Interrupt Clear Reads return an indeterminate value.



## Register 10: Hibernation RTC Trim (HIBRTCT), offset 0x024

This register contains the value that is used to trim the RTC clock predivider. It represents the computed underflow value that is used during the trim cycle. It is represented as  $0x7FFF \pm N$  clock cycles.

**Note:** **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and require a delay of  $t_{\text{HIB\_REG\_WRITE}}$  between write accesses. See “Register Access Timing” on page 125.

### Hibernation RTC Trim (HIBRTCT)

Base 0x400F.C000

Offset 0x024

Type R/W, reset 0x0000.7FFF

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TRIM	R/W	0x7FFF	RTC Trim Value  This value is loaded into the RTC predivider every 64 seconds. It is used to adjust the RTC rate to account for drift and inaccuracy in the clock source. The compensation is made by software by adjusting the default value of 0x7FFF up or down.

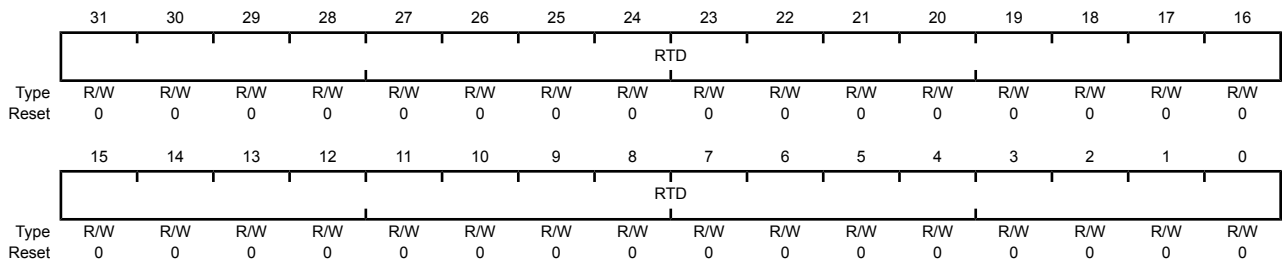
### Register 11: Hibernation Data (HIBDATA), offset 0x030-0x12C

This address space is implemented as a 64x32-bit memory (256 bytes). It can be loaded by the system processor in order to store any non-volatile state data and will not lose power during a power cut operation.

**Note:** HIBRTCC, HIBRTCM0, HIBRTCM1, HIBRTCLD, HIBRTCT, and HIBDATA are on the Hibernation module clock domain and require a delay of  $t_{HIB\_REG\_WRITE}$  between write accesses. See “Register Access Timing” on page 125.

#### Hibernation Data (HIBDATA)

Base 0x400F.C000  
 Offset 0x030-0x12C  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	RTD	R/W	0x0000.0000	Hibernation Module NV Registers[63:0]

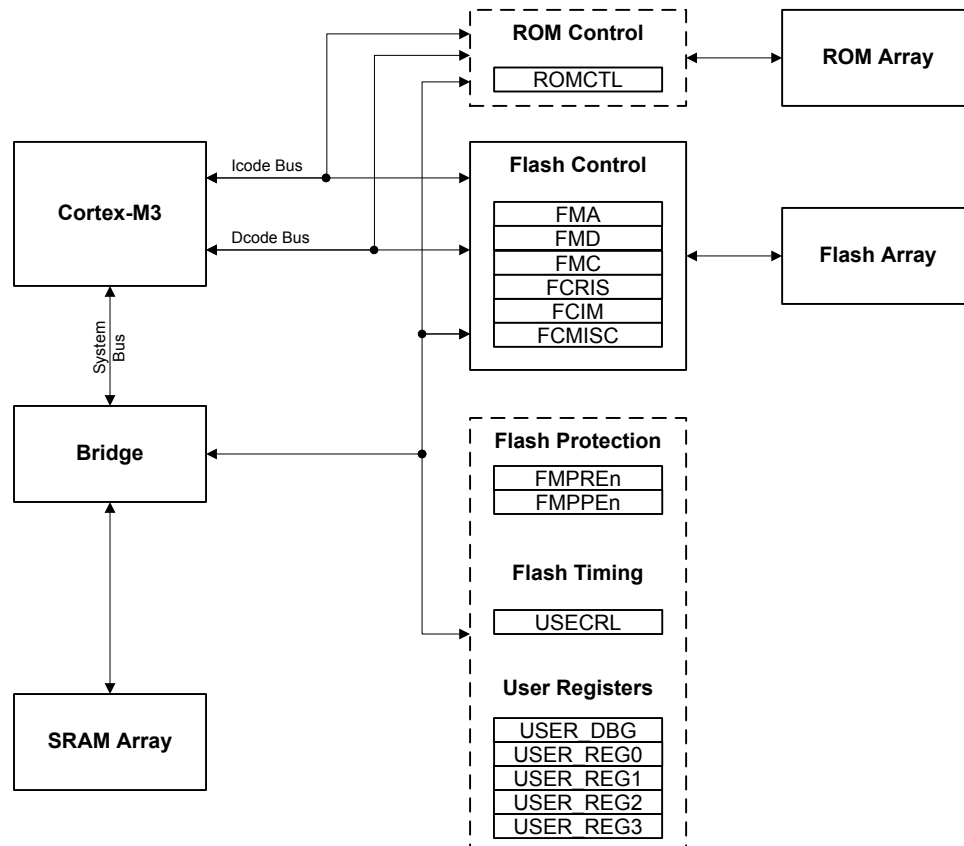
## 8 Internal Memory

The LM3S1607 microcontroller comes with 32 KB of bit-banded SRAM and 128 KB of flash memory. The flash controller provides a user-friendly interface, making flash programming a simple task. Flash protection can be applied to the flash memory on a 2-KB block basis.

### 8.1 Block Diagram

Figure 8-1 on page 147 illustrates the Flash functions. The dashed boxes in the figure indicate registers residing in the System Control module rather than the Flash Control module.

Figure 8-1. Flash Block Diagram



### 8.2 Functional Description

This section describes the functionality of the SRAM, ROM, and Flash memories.

#### 8.2.1 SRAM Memory

**Note:** The SRAM memory is implemented using two 32-bit wide SRAM banks (separate SRAM arrays). The banks are partitioned so that one bank contains all even words (the even bank) and the other contains all odd words (the odd bank). A write access that is followed immediately by a read access to the same bank will incur a stall of a single clock cycle. However, a write to one bank followed by a read of the other bank can occur in successive clock cycles without incurring any delay.

The internal SRAM of the Stellaris<sup>®</sup> devices is located at address 0x2000.0000 of the device memory map. To reduce the number of time consuming read-modify-write (RMW) operations, ARM has introduced *bit-banding* technology in the Cortex-M3 processor. With a bit-band-enabled processor, certain regions in the memory map (SRAM and peripheral space) can use address aliases to access individual bits in a single, atomic operation.

The bit-band alias is calculated by using the formula:

$$\text{bit-band alias} = \text{bit-band base} + (\text{byte offset} * 32) + (\text{bit number} * 4)$$

For example, if bit 3 at address 0x2000.1000 is to be modified, the bit-band alias is calculated as:

$$0x2200.0000 + (0x1000 * 32) + (3 * 4) = 0x2202.000C$$

With the alias address calculated, an instruction performing a read/write to address 0x2202.000C allows direct access to only bit 3 of the byte at address 0x2000.1000.

For details about bit-banding, please refer to Chapter 4, “Memory Map” in the *ARM<sup>®</sup> Cortex<sup>™</sup>-M3 Technical Reference Manual*.

## 8.2.2 ROM Memory

The 16 KB of internal ROM of the Stellaris<sup>®</sup> device is located at address 0x0100.0000 of the device memory map and contains the following components:

- A copy of the Serial Flash Loader and vector table
- A copy of the peripheral driver library (DriverLib) release for product-specific peripherals and interfaces
- Some pre-loaded code provided for manufacturing tests

## 8.2.3 Flash Memory

The flash is organized as a set of 1-KB blocks that can be individually erased. Erasing a block causes the entire contents of the block to be reset to all 1s. An individual 32-bit word can be programmed to change bits that are currently 1 to a 0. These blocks are paired into a set of 2-KB blocks that can be individually protected. The protection allows blocks to be marked as read-only or execute-only, providing different levels of code protection. Read-only blocks cannot be erased or programmed, protecting the contents of those blocks from being modified. Execute-only blocks cannot be erased or programmed, and can only be read by the controller instruction fetch mechanism, protecting the contents of those blocks from being read by either the controller or by a debugger.

### 8.2.3.1 Flash Memory Timing

The timing for the flash is automatically handled by the flash controller. However, in order to do so, it must know the clock rate of the system in order to time its internal signals properly. The number of clock cycles per microsecond must be provided to the flash controller for it to accomplish this timing. It is software's responsibility to keep the flash controller updated with this information via the **Usec Reload (USECRL)** register.

On reset, the **USECRL** register is loaded with a value that configures the flash timing so that it works with the maximum clock rate of the part. If software changes the system operating frequency, the new operating frequency minus 1 (in MHz) must be loaded into **USECRL** before any flash modifications are attempted. For example, if the device is operating at a speed of 20 MHz, a value of 0x13 (20-1) must be written to the **USECRL** register.

### 8.2.3.2 Flash Memory Protection

The user is provided two forms of flash protection per 2-KB flash blocks in two pairs of 32-bit wide registers. The protection policy for each form is controlled by individual bits (per policy per block) in the **FMPPEn** and **FMPREn** registers.

- **Flash Memory Protection Program Enable (FMPPEn)**: If set, the block may be programmed (written) or erased. If cleared, the block may not be changed.
- **Flash Memory Protection Read Enable (FMPREn)**: If set, the block may be executed or read by software or debuggers. If cleared, the block may only be executed and contents of the memory block are prohibited from being accessed as data.

The policies may be combined as shown in Table 8-1 on page 149.

**Table 8-1. Flash Protection Policy Combinations**

FMPPEn	FMPREn	Protection
0	0	Execute-only protection. The block may only be executed and may not be written or erased. This mode is used to protect code.
1	0	The block may be written, erased or executed, but not read. This combination is unlikely to be used.
0	1	Read-only protection. The block may be read or executed but may not be written or erased. This mode is used to lock the block from further modification while allowing any read or execute access.
1	1	No protection. The block may be written, erased, executed or read.

An access that attempts to program or erase a PE-protected block is prohibited. A controller interrupt may be optionally generated (by setting the **AMASK** bit in the **FIM** register) to alert software developers of poorly behaving software during the development and debug phases.

An access that attempts to read an RE-protected block is prohibited. Such accesses return data filled with all 0s. A controller interrupt may be optionally generated to alert software developers of poorly behaving software during the development and debug phases.

The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This implements a policy of open access and programmability. The register bits may be changed by writing the specific register bit. The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. Details on programming these bits are discussed in “Nonvolatile Register Programming” on page 150.

## 8.3 Flash Memory Initialization and Configuration

### 8.3.1 Flash Programming

The Stellaris<sup>®</sup> devices provide a user-friendly interface for flash programming. All erase/program operations are handled via three registers: **FMA**, **FMD**, and **FMC**.

#### 8.3.1.1 To program a 32-bit word

1. Write source data to the **FMD** register.
2. Write the target address to the **FMA** register.
3. Write the flash write key and the **WRITE** bit (a value of 0xA442.0001) to the **FMC** register.

4. Poll the **FMC** register until the `WRITE` bit is cleared.

### 8.3.1.2 To perform an erase of a 1-KB page

1. Write the page address to the **FMA** register.
2. Write the flash write key and the `ERASE` bit (a value of 0xA442.0002) to the **FMC** register.
3. Poll the **FMC** register until the `ERASE` bit is cleared.

### 8.3.1.3 To perform a mass erase of the flash

1. Write the flash write key and the `MERASE` bit (a value of 0xA442.0004) to the **FMC** register.
2. Poll the **FMC** register until the `MERASE` bit is cleared.

## 8.3.2 Nonvolatile Register Programming

This section discusses how to update registers that are resident within the flash memory itself. These registers exist in a separate space from the main flash array and are not affected by an `ERASE` or `MASS ERASE` operation. These nonvolatile registers are updated by using the `COMT` bit in the **FMC** register to activate a write operation. For the **USER\_DBG** register, the data to be written must be loaded into the **FMD** register before it is "committed". All other registers are R/W and can have their operation tried before committing them to nonvolatile memory.

**Important:** These registers can only have bits changed from 1 to 0 by user programming, but can be restored to their factory default values by performing the sequence described in the section called "Recovering a "Locked" Device" on page 53. The mass erase of the main flash array caused by the sequence is performed prior to restoring these registers.

In addition, the **USER\_REG0**, **USER\_REG1**, and **USER\_DBG** use bit 31 (NW) of their respective registers to indicate that they are available for user write. These three registers can only be written once whereas the flash protection registers may be written multiple times. Table 8-2 on page 150 provides the FMA address required for commitment of each of the registers and the source of the data to be written when the `COMT` bit of the **FMC** register is written with a value of 0xA442.0008. After writing the `COMT` bit, the user may poll the **FMC** register to wait for the commit operation to complete.

**Table 8-2. Flash Resident Registers<sup>a</sup>**

Register to be Committed	FMA Value	Data Source
FMPRE0	0x0000.0000	FMPRE0
FMPRE1	0x0000.0002	FMPRE1
FMPRE2	0x0000.0004	FMPRE2
FMPRE3	0x0000.0008	FMPRE3
FMPPE0	0x0000.0001	FMPPE0
FMPPE1	0x0000.0003	FMPPE1
FMPPE2	0x0000.0005	FMPPE2
FMPPE3	0x0000.0007	FMPPE3
USER_REG0	0x8000.0000	USER_REG0
USER_REG1	0x8000.0001	USER_REG1

Register to be Committed	FMA Value	Data Source
USER_DBG	0x7510.0000	FMD

a. Which FMPREn and FMPPEn registers are available depend on the flash size of your particular Stellaris® device.

## 8.4 Register Map

Table 8-3 on page 151 lists the ROM Controller registers and the Flash memory and control registers. The offset listed is a hexadecimal increment to the register's address. The ROM Controller registers are relative to the System Control base address of 0x400F.E000. The **FMA**, **FMD**, **FMC**, **FCRIS**, **FCIM**, and **FCMISC** registers are relative to the Flash control base address of 0x400F.D000. The **FMPREn**, **FMPPEn**, **USECRL**, **USER\_DBG**, and **USER\_REGn** registers are relative to the System Control base address of 0x400F.E000.

**Table 8-3. Flash Register Map**

Offset	Name	Type	Reset	Description	See page
<b>ROM Registers (System Control Offset)</b>					
0x0F0	RMCTL	R/W1C	-	ROM Control	153
<b>Flash Registers (Flash Control Offset)</b>					
0x000	FMA	R/W	0x0000.0000	Flash Memory Address	154
0x004	FMD	R/W	0x0000.0000	Flash Memory Data	155
0x008	FMC	R/W	0x0000.0000	Flash Memory Control	156
0x00C	FCRIS	RO	0x0000.0000	Flash Controller Raw Interrupt Status	158
0x010	FCIM	R/W	0x0000.0000	Flash Controller Interrupt Mask	159
0x014	FCMISC	R/W1C	0x0000.0000	Flash Controller Masked Interrupt Status and Clear	160
<b>Flash Registers (System Control Offset)</b>					
0x0F4	RMVER	RO	0x0000.0000	ROM Version Register	162
0x130	FMPRE0	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 0	163
0x200	FMPRE0	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 0	163
0x134	FMPPe0	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 0	164
0x400	FMPPe0	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 0	164
0x140	USECRL	R/W	0x31	USec Reload	161
0x1D0	USER_DBG	R/W	0xFFFF.FFFE	User Debug	165
0x1E0	USER_REG0	R/W	0xFFFF.FFFF	User Register 0	166
0x1E4	USER_REG1	R/W	0xFFFF.FFFF	User Register 1	167
0x1E8	USER_REG2	R/W	0xFFFF.FFFF	User Register 2	168
0x1EC	USER_REG3	R/W	0xFFFF.FFFF	User Register 3	169
0x204	FMPRE1	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 1	170
0x208	FMPRE2	R/W	0x0000.0000	Flash Memory Protection Read Enable 2	171

Offset	Name	Type	Reset	Description	See page
0x20C	FMPRE3	R/W	0x0000.0000	Flash Memory Protection Read Enable 3	172
0x404	FMPPE1	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 1	173
0x408	FMPPE2	R/W	0x0000.0000	Flash Memory Protection Program Enable 2	174
0x40C	FMPPE3	R/W	0x0000.0000	Flash Memory Protection Program Enable 3	175

## 8.5 ROM Register Descriptions (System Control Offset)

This section lists and describes the ROM Controller registers, in numerical order by address offset. Registers in this section are relative to the System Control base address of 0x400F.E000.



## Register 1: ROM Control (RMCTL), offset 0x0F0

This register provides control of the ROM controller state.

### ROM Control (RMCTL)

Base 0x400F.E000

Offset 0x0F0

Type R/W1C, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	BA	R/W1C	-	Boot Alias

- The device has ROM.
- The first two words of the Flash memory contain 0xFFFF.FFFF.

This bit is cleared by writing a 1 to this bit position.

When the BA bit is set, the boot alias is in effect and the ROM appears at address 0x0. When the BA bit is clear, the Flash appears at address 0x0.

## 8.6 Flash Register Descriptions (Flash Control Offset)

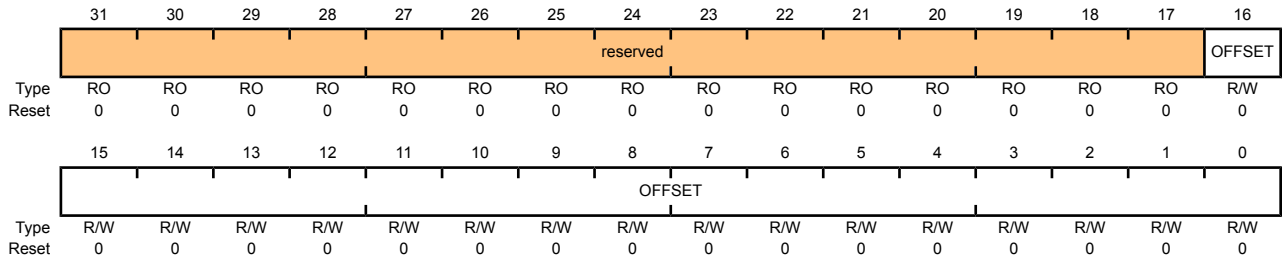
This section lists and describes the Flash Memory registers, in numerical order by address offset. Registers in this section are relative to the Flash control base address of 0x400F.D000.

## Register 2: Flash Memory Address (FMA), offset 0x000

During a write operation, this register contains a 4-byte-aligned address and specifies where the data is written. During erase operations, this register contains a 1 KB-aligned address and specifies which page is erased. Note that the alignment requirements must be met by software or the results of the operation are unpredictable.

### Flash Memory Address (FMA)

Base 0x400F.D000  
 Offset 0x000  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16:0	OFFSET	R/W	0x0	Address Offset  Address offset in flash where operation is performed, except for nonvolatile registers (see "Nonvolatile Register Programming" on page 150 for details on values for this field).

### Register 3: Flash Memory Data (FMD), offset 0x004

This register contains the data to be written during the programming cycle or read during the read cycle. Note that the contents of this register are undefined for a read access of an execute-only block. This register is not used during the erase cycles.

#### Flash Memory Data (FMD)

Base 0x400F.D000

Offset 0x004

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:0	DATA	R/W	0x0	Data Value Data value for write operation.

## Register 4: Flash Memory Control (FMC), offset 0x008

When this register is written, the flash controller initiates the appropriate access cycle for the location specified by the **Flash Memory Address (FMA)** register (see page 154). If the access is a write access, the data contained in the **Flash Memory Data (FMD)** register (see page 155) is written.

This is the final register written and initiates the memory operation. There are four control bits in the lower byte of this register that, when set, initiate the memory operation. The most used of these register bits are the `ERASE` and `WRITE` bits.

It is a programming error to write multiple control bits and the results of such an operation are unpredictable.

### Flash Memory Control (FMC)

Base 0x400F.D000  
Offset 0x008  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WRKEY															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												COMT	MERASE	ERASE	WRITE
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	WRKEY	WO	0x0	Flash Write Key  This field contains a write key, which is used to minimize the incidence of accidental flash writes. The value 0xA442 must be written into this field for a write to occur. Writes to the <b>FMC</b> register without this <code>WRKEY</code> value are ignored. A read of this field returns the value 0.
15:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	COMT	R/W	0	Commit Register Value  Commit (write) of register value to nonvolatile storage. A write of 0 has no effect on the state of this bit.  If read, the state of the previous commit access is provided. If the previous commit access is complete, a 0 is returned; otherwise, if the commit access is not complete, a 1 is returned.  This can take up to 50 $\mu$ s.
2	MERASE	R/W	0	Mass Erase Flash Memory  If this bit is set, the flash main memory of the device is all erased. A write of 0 has no effect on the state of this bit.  If read, the state of the previous mass erase access is provided. If the previous mass erase access is complete, a 0 is returned; otherwise, if the previous mass erase access is not complete, a 1 is returned.  This can take up to 250 ms.

---

Bit/Field	Name	Type	Reset	Description
1	ERASE	R/W	0	<p>Erase a Page of Flash Memory</p> <p>If this bit is set, the page of flash main memory as specified by the contents of <b>FMA</b> is erased. A write of 0 has no effect on the state of this bit.</p> <p>If read, the state of the previous erase access is provided. If the previous erase access is complete, a 0 is returned; otherwise, if the previous erase access is not complete, a 1 is returned.</p> <p>This can take up to 25 ms.</p>
0	WRITE	R/W	0	<p>Write a Word into Flash Memory</p> <p>If this bit is set, the data stored in <b>FMD</b> is written into the location as specified by the contents of <b>FMA</b>. A write of 0 has no effect on the state of this bit.</p> <p>If read, the state of the previous write update is provided. If the previous write access is complete, a 0 is returned; otherwise, if the write access is not complete, a 1 is returned.</p> <p>This can take up to 50 <math>\mu</math>s.</p>

## Register 5: Flash Controller Raw Interrupt Status (FCRIS), offset 0x00C

This register indicates that the flash controller has an interrupt condition. An interrupt is only signaled if the corresponding **FCIM** register bit is set.

### Flash Controller Raw Interrupt Status (FCRIS)

Base 0x400F.D000

Offset 0x00C

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved															PRIS	ARIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PRIS	RO	0	<p>Programming Raw Interrupt Status</p> <p>This bit indicates the current state of the programming cycle. If set, the programming cycle completed; if cleared, the programming cycle has not completed. Programming cycles are either write or erase actions generated through the <b>Flash Memory Control (FMC)</b> register bits (see page 156).</p>
0	ARIS	RO	0	<p>Access Raw Interrupt Status</p> <p>This bit indicates if the flash was improperly accessed. If set, the program tried to access the flash counter to the policy as set in the <b>Flash Memory Protection Read Enable (FMPREn)</b> and <b>Flash Memory Protection Program Enable (FMPPEn)</b> registers. Otherwise, no access has tried to improperly access the flash.</p>

## Register 6: Flash Controller Interrupt Mask (FCIM), offset 0x010

This register controls whether the flash controller generates interrupts to the controller.

### Flash Controller Interrupt Mask (FCIM)

Base 0x400F.D000

Offset 0x010

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PMASK	R/W	0	Programming Interrupt Mask  This bit controls the reporting of the programming raw interrupt status to the controller. If set, a programming-generated interrupt is promoted to the controller. Otherwise, interrupts are recorded but suppressed from the controller.
0	AMASK	R/W	0	Access Interrupt Mask  This bit controls the reporting of the access raw interrupt status to the controller. If set, an access-generated interrupt is promoted to the controller. Otherwise, interrupts are recorded but suppressed from the controller.

## Register 7: Flash Controller Masked Interrupt Status and Clear (FCMISC), offset 0x014

This register provides two functions. First, it reports the cause of an interrupt by indicating which interrupt source or sources are signalling the interrupt. Second, it serves as the method to clear the interrupt reporting.

### Flash Controller Masked Interrupt Status and Clear (FCMISC)

Base 0x400F.D000

Offset 0x014

Type R/W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														PMISC	AMISC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PMISC	R/W1C	0	Programming Masked Interrupt Status and Clear  This bit indicates whether an interrupt was signaled because a programming cycle completed and was not masked. This bit is cleared by writing a 1. The <code>PRIS</code> bit in the <code>FCRIS</code> register (see page 158) is also cleared when the <code>PMISC</code> bit is cleared.
0	AMISC	R/W1C	0	Access Masked Interrupt Status and Clear  This bit indicates whether an interrupt was signaled because an improper access was attempted and was not masked. This bit is cleared by writing a 1. The <code>ARIS</code> bit in the <code>FCRIS</code> register is also cleared when the <code>AMISC</code> bit is cleared.

## 8.7 Flash Register Descriptions (System Control Offset)

The remainder of this section lists and describes the Flash Memory registers, in numerical order by address offset. Registers in this section are relative to the System Control base address of 0x400F.E000.



## Register 8: USec Reload (USECRL), offset 0x140

**Note:** Offset is relative to System Control base address of 0x400F.E000

This register is provided as a means of creating a 1- $\mu$ s tick divider reload value for the flash controller. The internal flash has specific minimum and maximum requirements on the length of time the high voltage write pulse can be applied. It is required that this register contain the operating frequency (in MHz -1) whenever the flash is being erased or programmed. The user is required to change this value if the clocking conditions are changed for a flash erase/program operation.

### USec Reload (USECRL)

Base 0x400F.E000

Offset 0x140

Type R/W, reset 0x31

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	USEC	R/W	0x31	Microsecond Reload Value  MHz -1 of the controller clock when the flash is being erased or programmed.  If the maximum system frequency is being used, USEC should be set to 0x31 (50 MHz) whenever the flash is being erased or programmed.

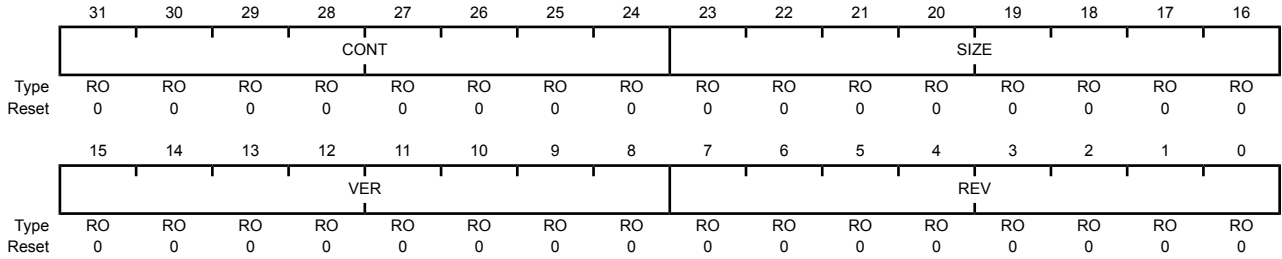
### Register 9: ROM Version Register (RMVER), offset 0x0F4

**Note:** Offset is relative to System Control base address of 0x400FE000.

A 32-bit read-only register containing the ROM content version information.

#### ROM Version Register (RMVER)

Base 0x400F.E000  
 Offset 0x0F4  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:24	CONT	RO	0x0	ROM Contents This field specifies the contents of the ROM.  Value Description 0x0 Stellaris Boot Loader & DriverLib
23:16	SIZE	RO	0x0	ROM Size This field encodes the size of the ROM.  Value Description 0x0 11 KB
15:8	VER	RO	0x0	ROM Version
7:0	REV	RO	0x0	ROM Revision

## Register 10: Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x130 and 0x200

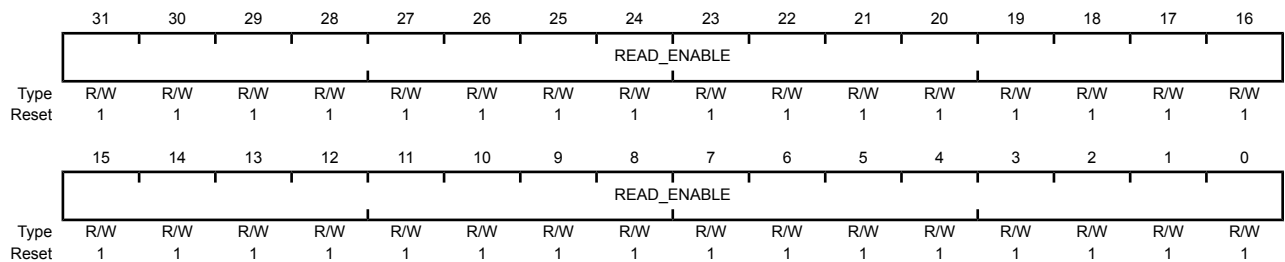
**Note:** This register is aliased for backwards compatibility.

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). This register is loaded during the power-on reset sequence. The factory settings for the **FMPREN** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. For additional information, see the "Flash Memory Protection" section.

### Flash Memory Protection Read Enable 0 (FMPRE0)

Base 0x400F.E000  
Offset 0x130 and 0x200  
Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0xFFFFFFFF	Flash Read Enable. Enables 2-KB flash blocks to be executed or read. The policies may be combined as shown in the table "Flash Protection Policy Combinations".
			Value	Description
			0xFFFFFFFF	Enables 128 KB of flash.

## Register 11: Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x134 and 0x400

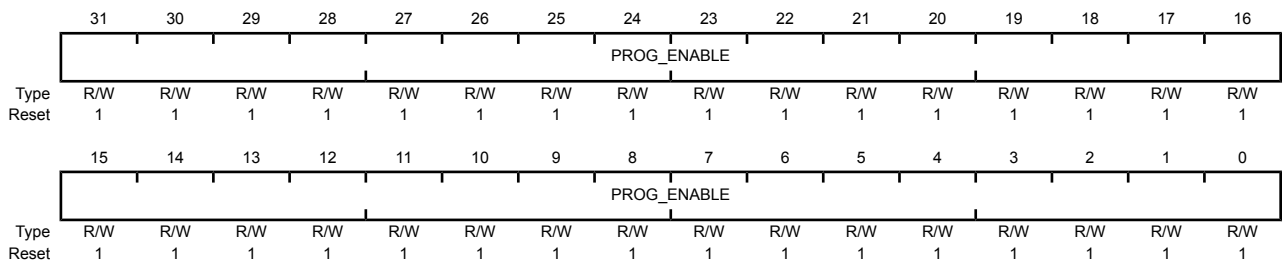
**Note:** This register is aliased for backwards compatibility.

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register stores the execute-only protection bits for each 2-KB flash block (**FMPREN** stores the execute-only bits). This register is loaded during the power-on reset sequence. The factory settings for the **FMPREN** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. For additional information, see the "Flash Memory Protection" section.

### Flash Memory Protection Program Enable 0 (FMPPE0)

Base 0x400F.E000  
Offset 0x134 and 0x400  
Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0xFFFFFFFF	Flash Programming Enable

Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table "Flash Protection Policy Combinations".

Value	Description
0xFFFFFFFF	Enables 128 KB of flash.

## Register 12: User Debug (USER\_DBG), offset 0x1D0

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register provides a write-once mechanism to disable external debugger access to the device in addition to 27 additional bits of user-defined data. The `DBG0` bit (bit 0) is set to 0 from the factory and the `DBG1` bit (bit 1) is set to 1, which enables external debuggers. Changing the `DBG1` bit to 0 disables any external debugger access to the device permanently, starting with the next power-up cycle of the device. The `NOTWRITTEN` bit (bit 31) indicates that the register is available to be written and is controlled through hardware to ensure that the register is only written once.

### User Debug (USER\_DBG)

Base 0x400F.E000

Offset 0x1D0

Type R/W, reset 0xFFFF.FFFE

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	NW	DATA															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	DATA														DBG1	DBG0	
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Bit/Field	Name	Type	Reset	Description
31	NW	R/W	1	User Debug Not Written. Specifies that this 32-bit dword has not been written.
30:2	DATA	R/W	0x1FFFFFFF	User Data. Contains the user data value. This field is initialized to all 1s and can only be written once.
1	DBG1	R/W	1	Debug Control 1. The <code>DBG1</code> bit must be 1 and <code>DBG0</code> must be 0 for debug to be available.
0	DBG0	R/W	0	Debug Control 0. The <code>DBG1</code> bit must be 1 and <code>DBG0</code> must be 0 for debug to be available.

### Register 13: User Register 0 (USER\_REG0), offset 0x1E0

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register provides 31 bits of user-defined data that is non-volatile and can only be written once. Bit 31 indicates that the register is available to be written and is controlled through hardware to ensure that the register is only written once. The write-once characteristics of this register are useful for keeping static information like communication addresses that need to be unique per part and would otherwise require an external EEPROM or other non-volatile device.

#### User Register 0 (USER\_REG0)

Base 0x400F.E000

Offset 0x1E0

Type R/W, reset 0xFFFF.FFFF

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

Bit/Field	Name	Type	Reset	Description
31	NW	R/W	1	Not Written. Specifies that this 32-bit dword has not been written.
30:0	DATA	R/W	0x7FFFFFFF	User Data. Contains the user data value. This field is initialized to all 1s and can only be written once.

## Register 14: User Register 1 (USER\_REG1), offset 0x1E4

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register provides 31 bits of user-defined data that is non-volatile and can only be written once. Bit 31 indicates that the register is available to be written and is controlled through hardware to ensure that the register is only written once. The write-once characteristics of this register are useful for keeping static information like communication addresses that need to be unique per part and would otherwise require an external EEPROM or other non-volatile device.

### User Register 1 (USER\_REG1)

Base 0x400F.E000

Offset 0x1E4

Type R/W, reset 0xFFFF.FFFF

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

Bit/Field	Name	Type	Reset	Description
31	NW	R/W	1	Not Written. Specifies that this 32-bit dword has not been written.
30:0	DATA	R/W	0x7FFFFFFF	User Data. Contains the user data value. This field is initialized to all 1s and can only be written once.

**Register 15: User Register 2 (USER\_REG2), offset 0x1E8**

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register provides 31 bits of user-defined data that is non-volatile and can only be written once. Bit 31 indicates that the register is available to be written and is controlled through hardware to ensure that the register is only written once. The write-once characteristics of this register are useful for keeping static information like communication addresses that need to be unique per part and would otherwise require an external EEPROM or other non-volatile device.

**User Register 2 (USER\_REG2)**

Base 0x400F.E000

Offset 0x1E8

Type R/W, reset 0xFFFF.FFFF

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

Bit/Field	Name	Type	Reset	Description
31	NW	R/W	1	Not Written. Specifies that this 32-bit dword has not been written.
30:0	DATA	R/W	0x7FFFFFFF	User Data. Contains the user data value. This field is initialized to all 1s and can only be written once.



## Register 16: User Register 3 (USER\_REG3), offset 0x1EC

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register provides 31 bits of user-defined data that is non-volatile and can only be written once. Bit 31 indicates that the register is available to be written and is controlled through hardware to ensure that the register is only written once. The write-once characteristics of this register are useful for keeping static information like communication addresses that need to be unique per part and would otherwise require an external EEPROM or other non-volatile device.

### User Register 3 (USER\_REG3)

Base 0x400F.E000

Offset 0x1EC

Type R/W, reset 0xFFFF.FFFF

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

Bit/Field	Name	Type	Reset	Description
31	NW	R/W	1	Not Written. Specifies that this 32-bit dword has not been written.
30:0	DATA	R/W	0x7FFFFFFF	User Data. Contains the user data value. This field is initialized to all 1s and can only be written once.

**Register 17: Flash Memory Protection Read Enable 1 (FMPRE1), offset 0x204**

**Note:** Offset is relative to System Control base address of 0x400FE000.

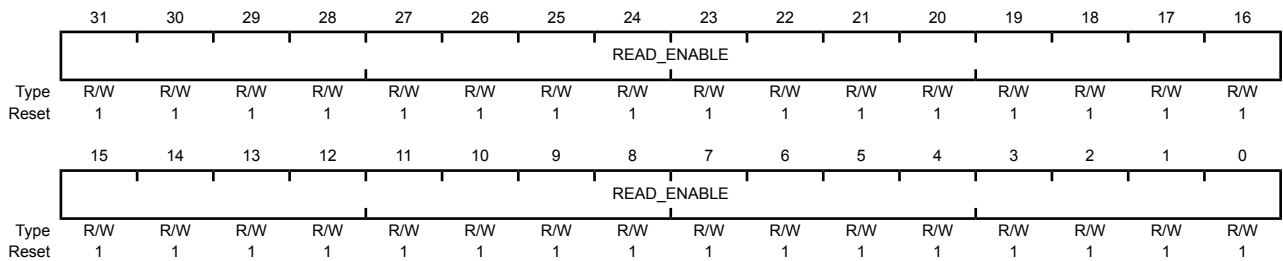
This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). This register is loaded during the power-on reset sequence. The factory settings for the **FMPREN** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. For additional information, see the "Flash Memory Protection" section.

## Flash Memory Protection Read Enable 1 (FMPRE1)

Base 0x400F.E000

Offset 0x204

Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0xFFFFFFFF	Flash Read Enable. Enables 2-KB flash blocks to be executed or read. The policies may be combined as shown in the table "Flash Protection Policy Combinations".
	Value	Description		
	0xFFFFFFFF	Enables 128 KB of flash.		

## Register 18: Flash Memory Protection Read Enable 2 (FMPRE2), offset 0x208

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). This register is loaded during the power-on reset sequence. The factory settings for the **FMPREN** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. For additional information, see the "Flash Memory Protection" section.

### Flash Memory Protection Read Enable 2 (FMPRE2)

Base 0x400F.E000

Offset 0x208

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0x00000000	Flash Read Enable. Enables 2-KB flash blocks to be executed or read. The policies may be combined as shown in the table "Flash Protection Policy Combinations".
	Value	Description		
	0x00000000	Enables 128 KB of flash.		

**Register 19: Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C**

**Note:** Offset is relative to System Control base address of 0x400FE000.

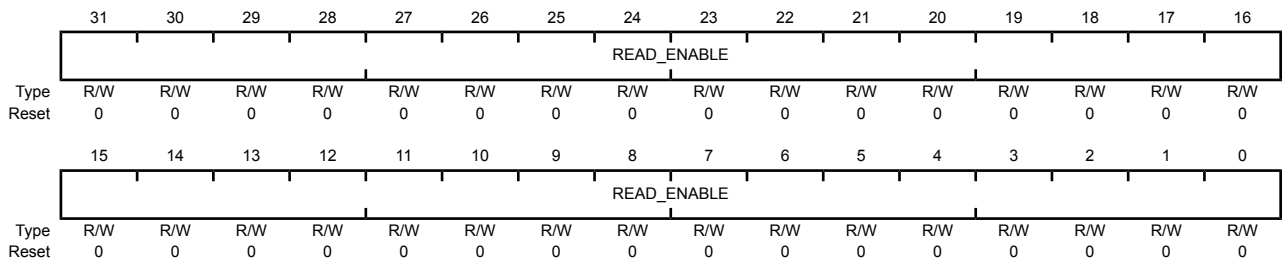
This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). This register is loaded during the power-on reset sequence. The factory settings for the **FMPREN** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. For additional information, see the "Flash Memory Protection" section.

## Flash Memory Protection Read Enable 3 (FMPRE3)

Base 0x400F.E000

Offset 0x20C

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0x00000000	Flash Read Enable. Enables 2-KB flash blocks to be executed or read. The policies may be combined as shown in the table "Flash Protection Policy Combinations".
				Value      Description
				0x00000000 Enables 128 KB of flash.

## Register 20: Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404

**Note:** Offset is relative to System Control base address of 0x400FE000.

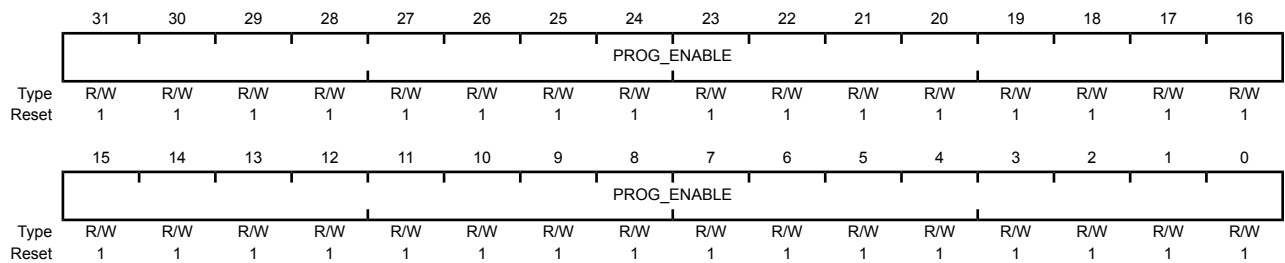
This register stores the execute-only protection bits for each 2-KB flash block (**FMPPE<sub>n</sub>** stores the execute-only bits). This register is loaded during the power-on reset sequence. The factory settings for the **FMPPE<sub>n</sub>** and **FMPPE<sub>n</sub>** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. For additional information, see the "Flash Memory Protection" section.

### Flash Memory Protection Program Enable 1 (FMPPE1)

Base 0x400F.E000

Offset 0x404

Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0xFFFFFFFF	Flash Programming Enable. Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table "Flash Protection Policy Combinations".
				Value      Description
				0xFFFFFFFF Enables 128 KB of flash.

## Register 21: Flash Memory Protection Program Enable 2 (FMPPE2), offset 0x408

**Note:** Offset is relative to System Control base address of 0x400FE000.

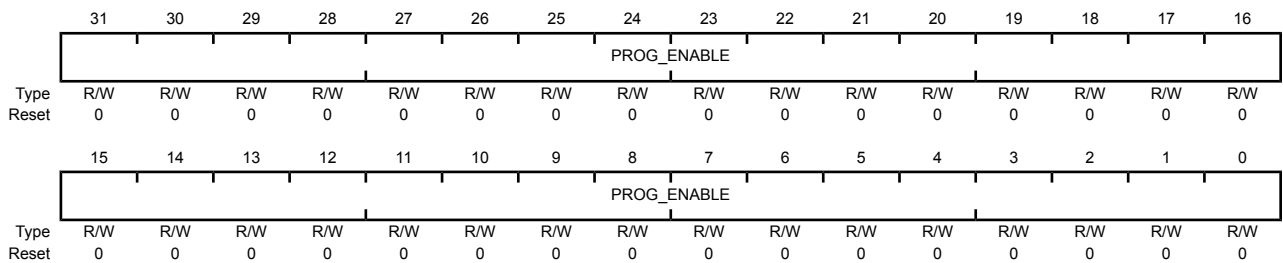
This register stores the execute-only protection bits for each 2-KB flash block (**FMPPE<sub>n</sub>** stores the execute-only bits). This register is loaded during the power-on reset sequence. The factory settings for the **FMPPE<sub>n</sub>** and **FMPPE<sub>n</sub>** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. For additional information, see the "Flash Memory Protection" section.

### Flash Memory Protection Program Enable 2 (FMPPE2)

Base 0x400F.E000

Offset 0x408

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0x00000000	Flash Programming Enable. Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table "Flash Protection Policy Combinations".
				Value      Description
			0x00000000	Enables 128 KB of flash.

## Register 22: Flash Memory Protection Program Enable 3 (FMPPE3), offset 0x40C

**Note:** Offset is relative to System Control base address of 0x400FE000.

This register stores the execute-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). This register is loaded during the power-on reset sequence. The factory settings for the **FMPPEn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. For additional information, see the "Flash Memory Protection" section.

### Flash Memory Protection Program Enable 3 (FMPPE3)

Base 0x400F.E000

Offset 0x40C

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0x00000000	Flash Programming Enable. Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table "Flash Protection Policy Combinations".
				Value      Description
			0x00000000	Enables 128 KB of flash.

## 9 Micro Direct Memory Access ( $\mu$ DMA)

The LM3S1607 microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA ( $\mu$ DMA). The  $\mu$ DMA controller provides a way to offload data transfer tasks from the Cortex-M3 processor, allowing for more efficient use of the processor and the expanded available bus bandwidth. The  $\mu$ DMA controller can perform transfers between memory and peripherals. It has dedicated channels for each supported peripheral and can be programmed to automatically perform transfers between peripherals and memory as the peripheral is ready to transfer more data. The  $\mu$ DMA controller also supports sophisticated transfer modes such as ping-pong and scatter-gather, which allows the processor to set up a list of transfer tasks for the controller.

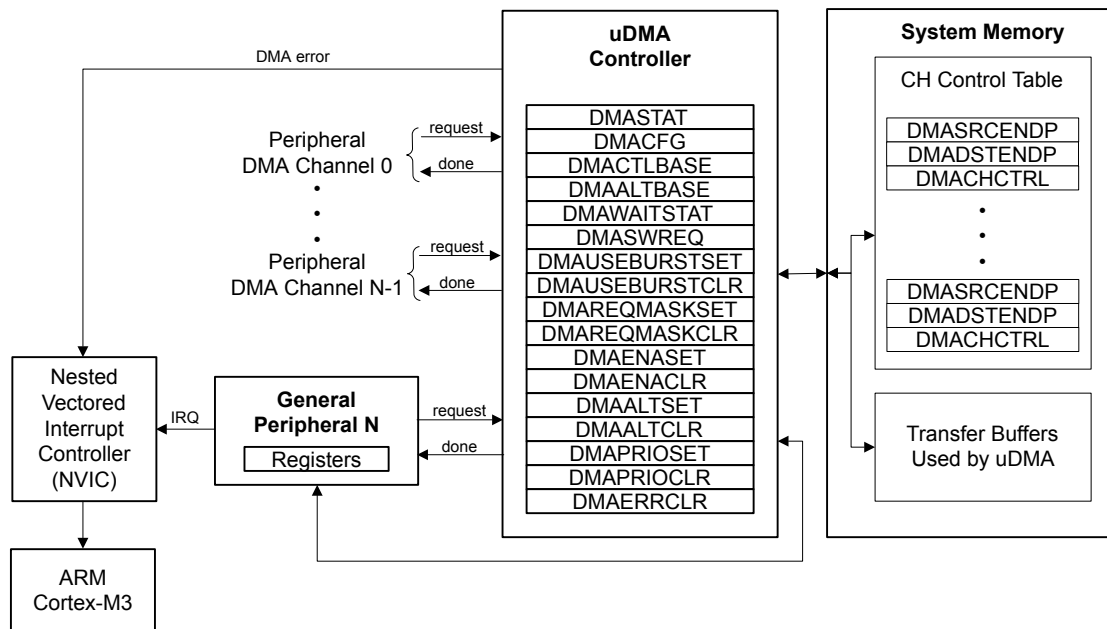
The  $\mu$ DMA controller has the following features:

- ARM PrimeCell® 32-channel configurable  $\mu$ DMA controller
- Support for multiple transfer modes:
  - Basic, for simple transfer scenarios
  - Ping-pong, for continuous data flow to/from peripherals
  - Scatter-gather, from a programmable list of arbitrary transfers initiated from a single request
- Dedicated channels for supported peripherals
- One channel each for receive and transmit path for bidirectional peripherals
- Dedicated channel for software-initiated transfers
- Independently configured and operated channels
- Per-channel configurable bus arbitration scheme
- Two levels of priority
- Design optimizations for improved bus access performance between  $\mu$ DMA controller and the processor core:
  - $\mu$ DMA controller access is subordinate to core access
  - RAM striping
  - Peripheral bus segmentation
- Data sizes of 8, 16, and 32 bits
- Source and destination address increment size of byte, half-word, word, or no increment
- Maskable device requests
- Optional software initiated requests for any channel
- Interrupt on transfer completion, with a separate interrupt per channel



## 9.1 Block Diagram

Figure 9-1.  $\mu$ DMA Block Diagram



## 9.2 Functional Description

The  $\mu$ DMA controller is a flexible and highly configurable DMA controller designed to work efficiently with the microcontroller's Cortex-M3 processor core. It supports multiple data sizes and address increment schemes, multiple levels of priority among DMA channels, and several transfer modes to allow for sophisticated programmed data transfers. The DMA controller's usage of the bus is always subordinate to the processor core, and so it will never hold up a bus transaction by the processor. Because the  $\mu$ DMA controller is only using otherwise-idle bus cycles, the data transfer bandwidth it provides is essentially free, with no impact on the rest of the system. The bus architecture has been optimized to greatly reduce contention between the processor core and the  $\mu$ DMA controller, thus improving performance. The optimizations include RAM striping and peripheral bus segmentation, which in many cases allows both the processor core and the  $\mu$ DMA controller to access the bus and perform simultaneous data transfers.

Each peripheral function that is supported has a dedicated channel on the  $\mu$ DMA controller that can be configured independently.

The  $\mu$ DMA controller makes use of a unique configuration method by using channel control structures that are maintained in system memory by the processor. While simple transfer modes are supported, it is also possible to build up sophisticated "task" lists in memory that allow the controller to perform arbitrary-sized transfers to and from arbitrary locations as part of a single transfer request. The controller also supports the use of ping-pong buffering to accommodate constant streaming of data to or from a peripheral.

Each channel also has a configurable arbitration size. The arbitration size is the number of items that will be transferred in a burst before the controller re-arbitrates for channel priority. Using the arbitration size, it is possible to control exactly how many items are transferred to or from a peripheral each time it makes a DMA service request.

## 9.2.1 Channel Assignments

$\mu$ DMA channels 0-31 are assigned to peripherals according to the following table.

**Note:** Channels that are not listed in the table may be assigned to peripherals in the future. However, they are currently available for software use.

**Table 9-1. DMA Channel Assignments**

DMA Channel	Peripheral Assigned
8	UART0 Receive
9	UART0 Transmit
10	SSI0 Receive
11	SSI0 Transmit
22	UART1 Receive
23	UART1 Transmit
30	Dedicated for software use

## 9.2.2 Priority

The  $\mu$ DMA controller assigns priority to each channel based on the channel number and the priority level bit for the channel. Channel number 0 has the highest priority and as the channel number increases, the priority of a channel decreases. Each channel has a priority level bit to provide two levels of priority: default priority and high priority. If the priority level bit is set, then that channel has higher priority than all other channels at default priority. If multiple channels are set for high priority, then the channel number is used to determine relative priority among all the high priority channels.

The priority bit for a channel can be set using the **DMA Channel Priority Set (DMAPRIOSET)** register, and cleared with the **DMA Channel Priority Clear (DMAPRIOCLR)** register.

## 9.2.3 Arbitration Size

When a  $\mu$ DMA channel requests a transfer, the  $\mu$ DMA controller arbitrates between all the channels making a request and services the DMA channel with the highest priority. Once a transfer begins, it continues for a selectable number of transfers before re-arbitrating among the requesting channels again. The arbitration size can be configured for each channel, ranging from 1 to 1024 item transfers. After the  $\mu$ DMA controller transfers the number of items specified by the arbitration size, it then checks among all the channels making a request and services the channel with the highest priority.

If a lower priority DMA channel uses a large arbitration size, the latency for higher priority channels will be increased because the  $\mu$ DMA controller will complete the lower priority burst before checking for higher priority requests. Therefore, lower priority channels should not use a large arbitration size for best response on high priority channels.

The arbitration size can also be thought of as a burst size. It is the maximum number of items that will be transferred at any one time in a burst. Here, the term arbitration refers to determination of DMA channel priority, not arbitration for the bus. When the  $\mu$ DMA controller arbitrates for the bus, the processor always takes priority. Furthermore, the  $\mu$ DMA controller will be held off whenever the processor needs to perform a bus transaction on the same bus, even in the middle of a burst transfer.

## 9.2.4 Request Types

The  $\mu$ DMA controller responds to two types of requests from a peripheral: single or burst. Each peripheral may support either or both types of requests. A single request means that the peripheral

is ready to transfer one item, while a burst request means that the peripheral is ready to transfer multiple items.

The  $\mu$ DMA controller responds differently depending on whether the peripheral is making a single request or a burst request. If both are asserted and the  $\mu$ DMA channel has been set up for a burst transfer, then the burst request takes precedence. See Table 9-2 on page 179, which shows how each peripheral supports the two request types.

**Table 9-2. Request Type Support**

Peripheral	Single Request Signal	Burst Request Signal
UART TX	TX FIFO Not Full	TX FIFO Level (configurable)
UART RX	RX FIFO Not Empty	RX FIFO Level (configurable)
SSI TX	TX FIFO Not Full	TX FIFO Level (fixed at 4)
SSI RX	RX FIFO Not Empty	RX FIFO Level (fixed at 4)

### 9.2.4.1 Single Request

When a single request is detected, and not a burst request, the  $\mu$ DMA controller will transfer one item, and then stop and wait for another request.

### 9.2.4.2 Burst Request

When a burst request is detected, the  $\mu$ DMA controller will transfer the number of items that is the lesser of the arbitration size or the number of items remaining in the transfer. Therefore, the arbitration size should be the same as the number of data items that the peripheral can accommodate when making a burst request. For example, the UART will generate a burst request based on the FIFO trigger level. In this case, the arbitration size should be set to the amount of data that the FIFO can transfer when the trigger level is reached.

It may be desirable to use only burst transfers and not allow single transfers. For example, perhaps the nature of the data is such that it only makes sense when transferred together as a single unit rather than one piece at a time. The single request can be disabled by using the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register. By setting the bit for a channel in this register, the  $\mu$ DMA controller will only respond to burst requests for that channel.

## 9.2.5 Channel Configuration

The  $\mu$ DMA controller uses an area of system memory to store a set of channel control structures in a table. The control table may have one or two entries for each DMA channel. Each entry in the table structure contains source and destination pointers, transfer size, and transfer mode. The control table can be located anywhere in system memory, but it must be contiguous and aligned on a 1024-byte boundary.

Table 9-3 on page 180 shows the layout in memory of the channel control table. Each channel may have one or two control structures in the control table: a primary control structure and an optional alternate control structure. The table is organized so that all of the primary entries are in the first half of the table and all the alternate structures are in the second half of the table. The primary entry is used for simple transfer modes where transfers can be reconfigured and restarted after each transfer is complete. In this case, the alternate control structures are not used and therefore only the first half of the table needs to be allocated in memory. The second half of the control table is not needed and that memory can be used for something else. If a more complex transfer mode is used such as ping-pong or scatter-gather, then the alternate control structure is also used and memory space should be allocated for the entire table.

Any unused memory in the control table may be used by the application. This includes the control structures for any channels that are unused by the application as well as the unused control word for each channel.

**Table 9-3. Control Structure Memory Map**

Offset	Channel
0x0	0, Primary
0x10	1, Primary
...	...
0x1F0	31, Primary
0x200	0, Alternate
0x210	1, Alternate
...	...
0x3F0	31, Alternate

Table 9-4 on page 180 shows an individual control structure entry in the control table. Each entry has a source and destination *end* pointer. These pointers point to the ending address of the transfer and are inclusive. If the source or destination is non-incrementing (as for a peripheral register), then the pointer should point to the transfer address.

**Table 9-4. Channel Control Structure**

Offset	Description
0x000	Source End Pointer
0x004	Destination End Pointer
0x008	Control Word
0x00C	Unused

The remaining part of the control structure is the control word. The control word contains the following fields:

- Source and destination data sizes
- Source and destination address increment size
- Number of transfers before bus arbitration
- Total number of items to transfer
- Useburst flag
- Transfer mode

The control word and each field are described in detail in “ $\mu$ DMA Channel Control Structure” on page 197. The  $\mu$ DMA controller updates the transfer size and transfer mode fields as the transfer is performed. At the end of a transfer, the transfer size will indicate 0, and the transfer mode will indicate "stopped". Since the control word is modified by the  $\mu$ DMA controller, it must be reconfigured before each new transfer. The source and destination end pointers are not modified so they can be left unchanged if the source or destination addresses remain the same.

Prior to starting a transfer, a  $\mu$ DMA channel must be enabled by setting the appropriate bit in the **DMA Channel Enable Set ((DMAENASET)** register. A channel can be disabled by setting the

channel bit in the **DMA Channel Enable Clear (DMAENACLR)** register. At the end of a complete DMA transfer, the controller will automatically disable the channel.

## 9.2.6 Transfer Modes

The  $\mu$ DMA controller supports several transfer modes. Two of the modes support simple one-time transfers. There are several complex modes that are meant to support a continuous flow of data.

### 9.2.6.1 Stop Mode

While Stop is not actually a transfer mode, it is a valid value for the mode field of the control word. When the mode field has this value, the  $\mu$ DMA controller will not perform a transfer and will disable the channel if it is enabled. At the end of a transfer, the  $\mu$ DMA controller will update the control word to set the mode to Stop.

### 9.2.6.2 Basic Mode

In Basic mode, the  $\mu$ DMA controller will perform transfers as long as there are more items to transfer and a transfer request is present. This mode is used with peripherals that assert a DMA request signal whenever the peripheral is ready for a data transfer. Basic mode should not be used in any situation where the request is momentary but the entire transfer should be completed. For example, for a software initiated transfer, the request is momentary, and if Basic mode is used then only one item will be transferred on a software request.

When all of the items have been transferred using Basic mode, the  $\mu$ DMA controller will set the mode for that channel to Stop.

### 9.2.6.3 Auto Mode

Auto mode is similar to Basic mode, except that once a transfer request is received the transfer will run to completion, even if the DMA request is removed. This mode is suitable for software-triggered transfers. Generally, you would not use Auto mode with a peripheral.

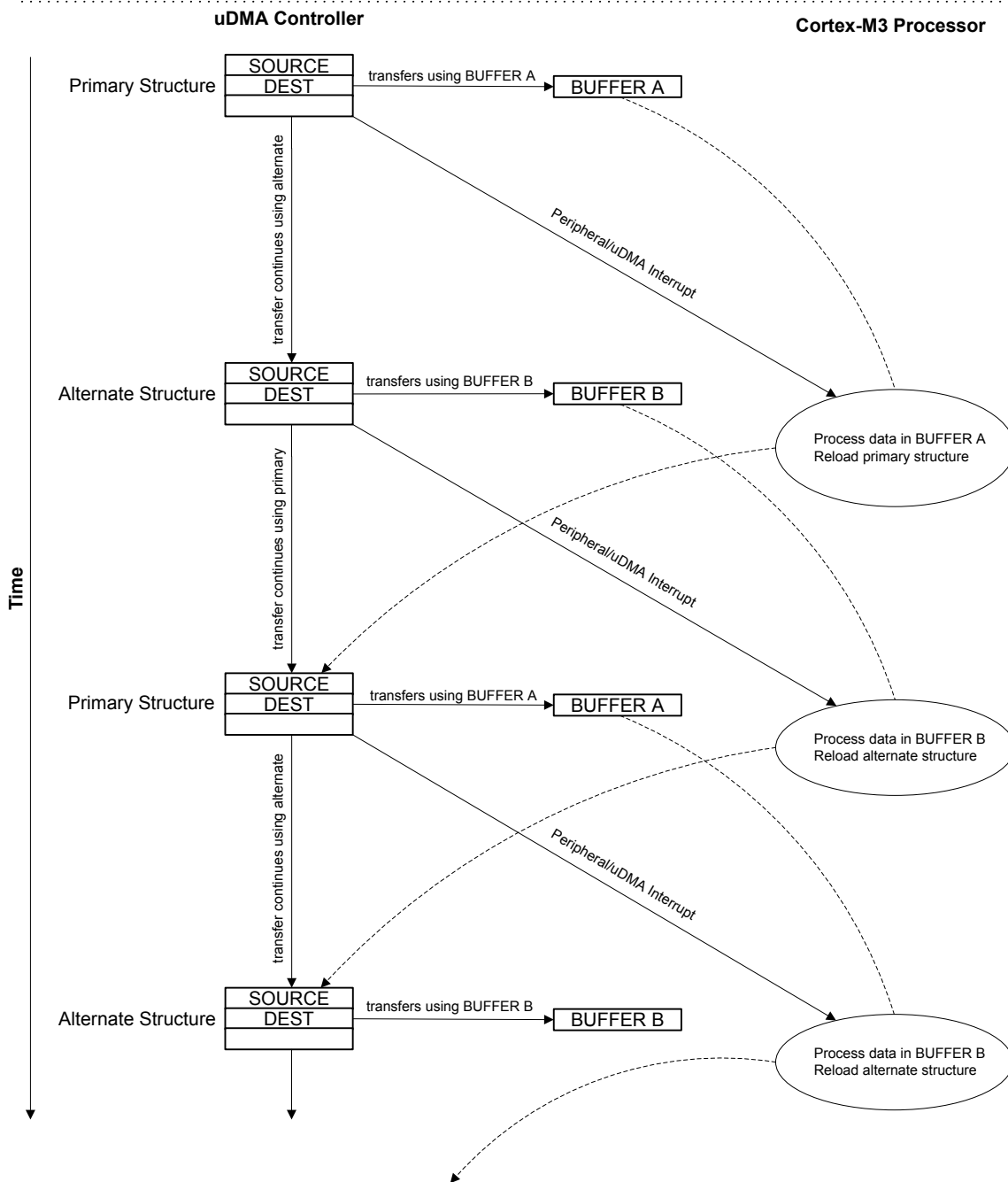
When all the items have been transferred using Auto mode, the  $\mu$ DMA controller will set the mode for that channel to Stop.

### 9.2.6.4 Ping-Pong

Ping-Pong mode is used to support a continuous data flow to or from a peripheral. To use Ping-Pong mode, both the primary and alternate data structures are used. Both are set up by the processor for data transfer between memory and a peripheral. Then the transfer is started using the primary control structure. When the transfer using the primary control structure is complete, the  $\mu$ DMA controller will then read the alternate control structure for that channel to continue the transfer. Each time this happens, an interrupt is generated and the processor can reload the control structure for the just-completed transfer. Data flow can continue indefinitely this way, using the primary and alternate control structures to switch back and forth between buffers as the data flows to or from the peripheral.

Refer to Figure 9-2 on page 182 for an example showing operation in Ping-Pong mode.

Figure 9-2. Example of Ping-Pong DMA Transaction



### 9.2.6.5 Memory Scatter-Gather

Memory Scatter-Gather mode is a complex mode used when data needs to be transferred to or from varied locations in memory instead of a set of contiguous locations in a memory buffer. For example, a gather DMA operation could be used to selectively read the payload of several stored packets of a communication protocol, and store them together in sequence in a memory buffer.

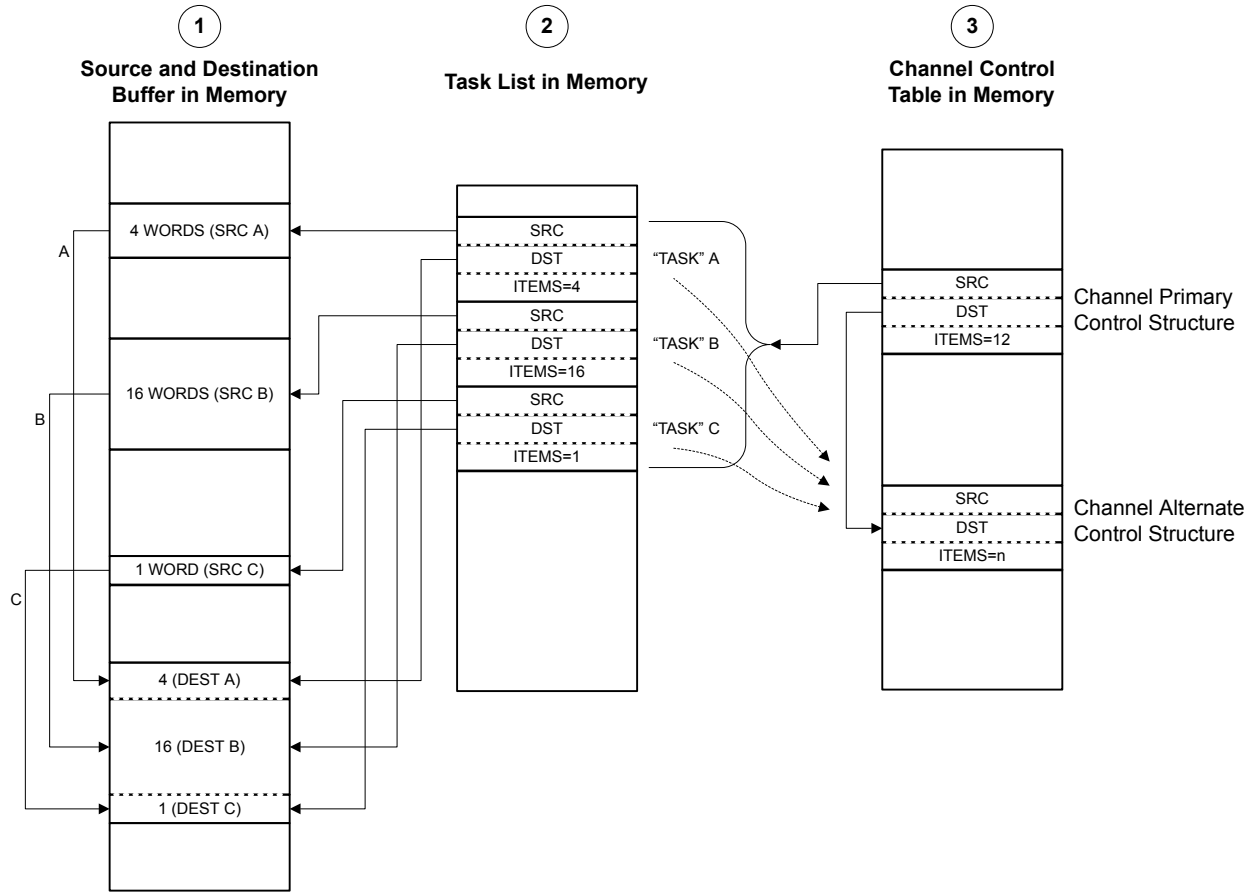
In Memory Scatter-Gather mode, the primary control structure is used to program the alternate control structure from a table in memory. The table is set up by the processor software and contains a list of control structures, each containing the source and destination end pointers, and the control word for a specific transfer. The mode of each control word must be set to Scatter-Gather mode. Each entry in the table is copied in turn to the alternate structure where it is then executed. The  $\mu$ DMA controller alternates between using the primary control structure to copy the next transfer instruction from the list, and then executing the new transfer instruction. The end of the list is marked by setting the control word for the last entry to use Basic transfer mode. Once the last transfer is performed using Basic mode, the  $\mu$ DMA controller will stop. A completion interrupt will only be generated after the last transfer. It is possible to loop the list by having the last entry copy the primary control structure to point back to the beginning of the list (or to a new list). It is also possible to trigger a set of other channels to perform a transfer, either directly by programming a write to the software trigger for another channel, or indirectly by causing a peripheral action that will result in a  $\mu$ DMA request.

By programming the  $\mu$ DMA controller using this method, a set of arbitrary transfers can be performed based on a single DMA request.

Refer to Figure 9-3 on page 184 and Figure 9-4 on page 185, which show an example of operation in Memory Scatter-Gather mode. This example shows a *gather* operation, where data in three separate buffers in memory will be copied together into one buffer. Figure 9-3 on page 184 shows how the application sets up a  $\mu$ DMA *task list* in memory that is used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel that will be used for the operation is configured to copy from the task list to the alternate control structure.

Figure 9-4 on page 185 shows the sequence as the  $\mu$ DMA controller performs the three sets of copy operations. First, using the primary control structure, the  $\mu$ DMA controller loads the alternate control structure with task A. It then performs the copy operation specified by task A, copying the data from the source buffer A to the destination buffer. Next, the  $\mu$ DMA controller again uses the primary control structure to load task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for task C.

Figure 9-3. Memory Scatter-Gather, Setup and Configuration

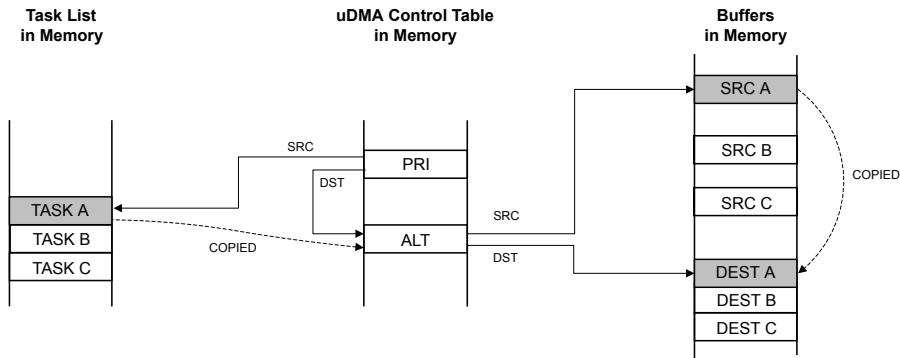


NOTES:

1. Application has a need to copy data items from three separate location in memory into one combined buffer.
2. Application sets up  $\mu$ DMA "task list" in memory, which contains the pointers and control configuration for three  $\mu$ DMA copy "tasks."
3. Application sets up the channel primary control structure to copy each task configuration, one at a time, to the alternate control structure, where it will be executed by the  $\mu$ DMA controller.

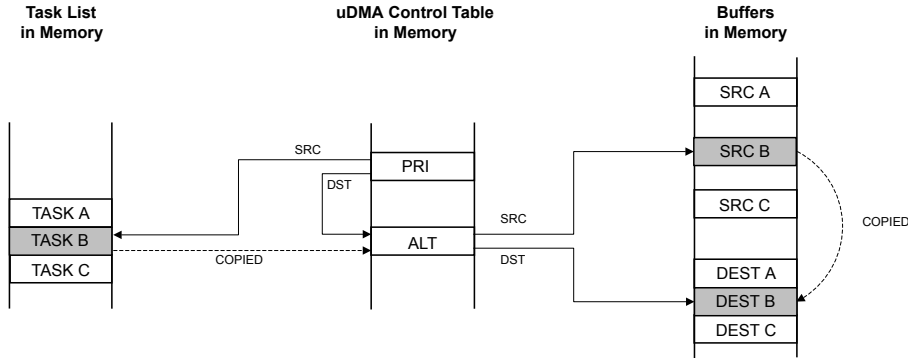


Figure 9-4. Memory Scatter-Gather,  $\mu$ DMA Copy Sequence



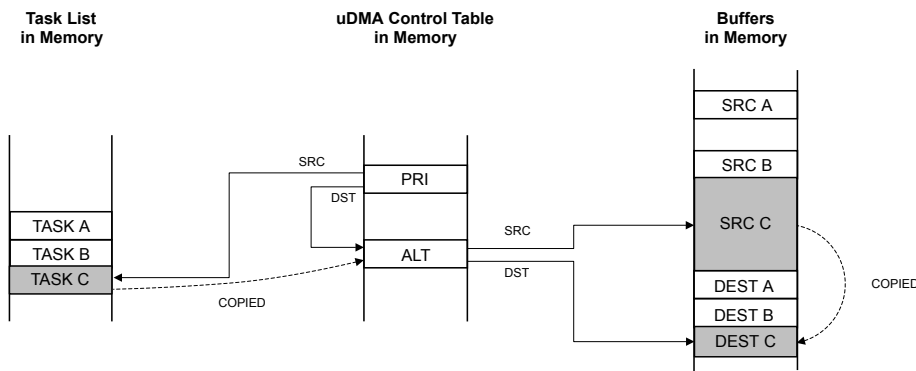
Using the channel's primary control structure, the  $\mu$ DMA controller copies task A configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the  $\mu$ DMA controller copies data from the source buffer A to the destination buffer.



Using the channel's primary control structure, the  $\mu$ DMA controller copies task B configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the  $\mu$ DMA controller copies data from the source buffer B to the destination buffer.



Using the channel's primary control structure, the  $\mu$ DMA controller copies task C configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the  $\mu$ DMA controller copies data from the source buffer C to the destination buffer.

### 9.2.6.6 Peripheral Scatter-Gather

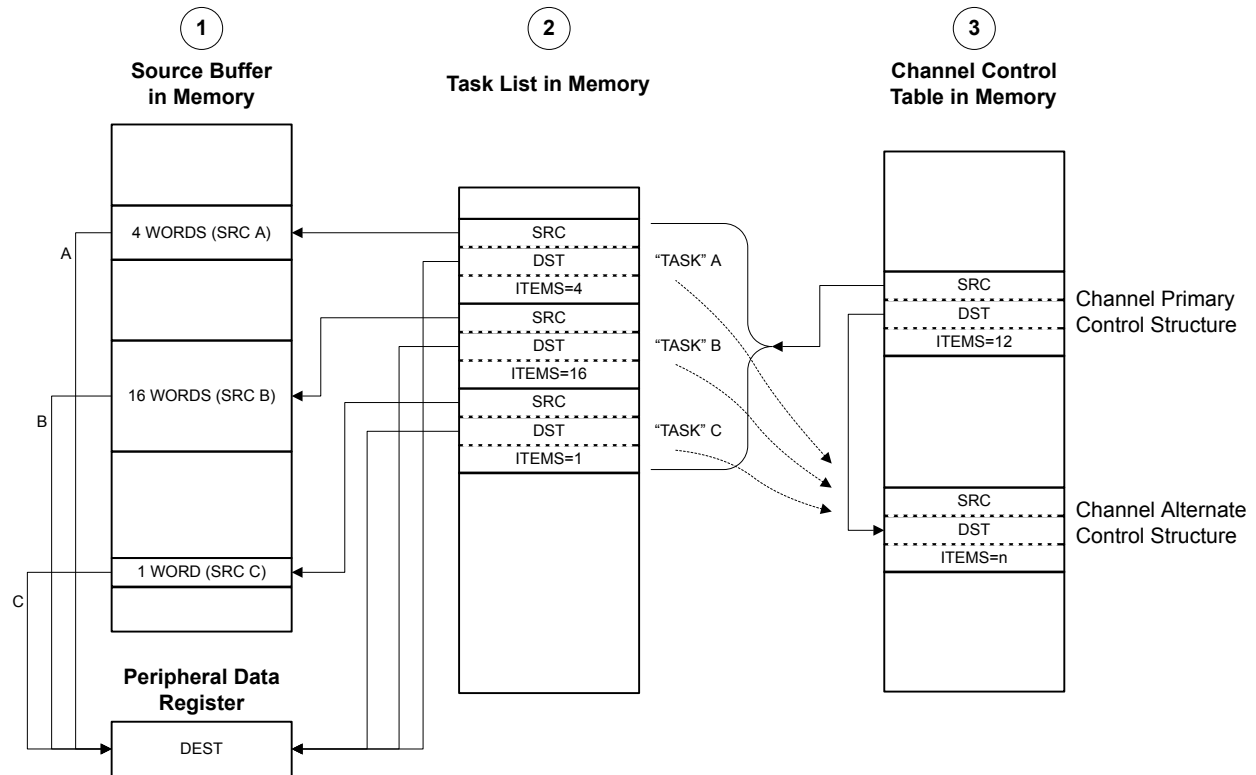
Peripheral Scatter-Gather mode is very similar to Memory Scatter-Gather, except that the transfers are controlled by a peripheral making a DMA request. Upon detecting a DMA request from the peripheral, the  $\mu$ DMA controller will use the primary control structure to copy one entry from the list to the alternate control structure, and then perform the transfer. At the end of this transfer, the next transfer will only be started if the peripheral again asserts a DMA request. The  $\mu$ DMA controller will continue to perform transfers from the list only when the peripheral is making a request, until the last transfer is complete. A completion interrupt will only be generated after the last transfer.

By programming the  $\mu$ DMA controller using this method, data can be transferred to or from a peripheral from a set of arbitrary locations whenever the peripheral is ready to transfer data.

Refer to Figure 9-5 on page 187 and Figure 9-6 on page 188, which show an example of operation in Peripheral Scatter-Gather mode. This example shows a gather operation, where data from three separate buffers in memory will be copied to a single peripheral data register. Figure 9-5 on page 187 shows how the application sets up a  $\mu$ DMA task list in memory that is used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel that will be used for the operation is configured to copy from the task list to the alternate control structure.

Figure 9-6 on page 188 shows the sequence as the  $\mu$ DMA controller performs the three sets of copy operations. First, using the primary control structure, the  $\mu$ DMA controller loads the alternate control structure with task A. It then performs the copy operation specified by task A, copying the data from the source buffer A to the peripheral data register. Next, the  $\mu$ DMA controller again uses the primary control structure to load task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for task C.

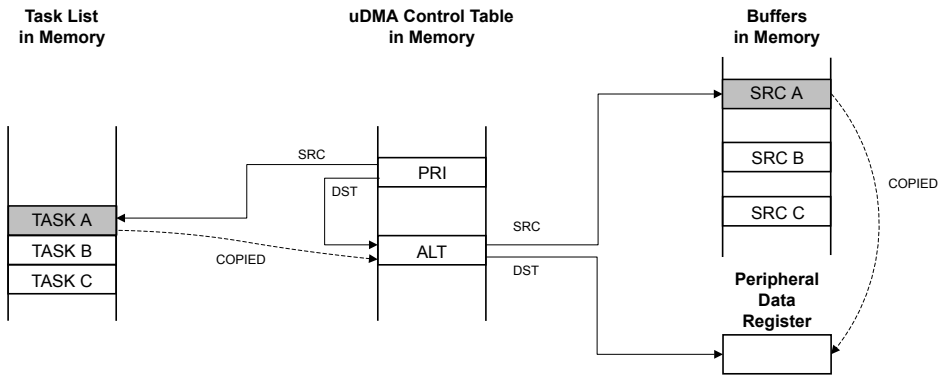
Figure 9-5. Peripheral Scatter-Gather, Setup and Configuration



## NOTES:

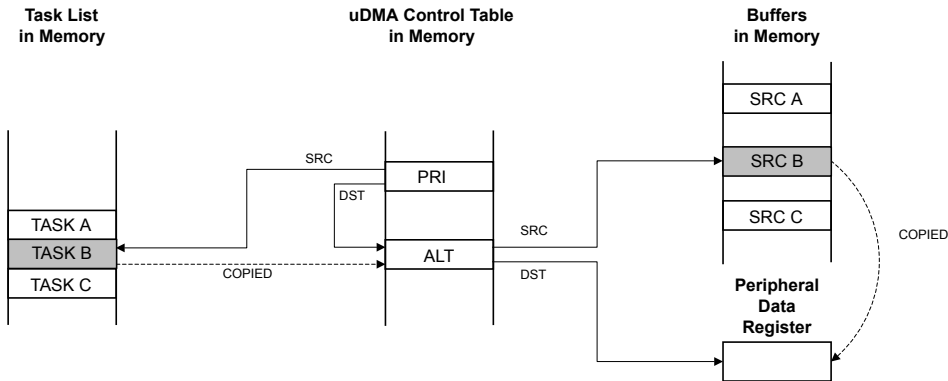
1. Application has a need to copy data items from three separate location in memory into a peripheral data register.
2. Application sets up uDMA "task list" in memory, which contains the pointers and control configuration for three uDMA copy "tasks."
3. Application sets up the channel primary control structure to copy each task configuration, one at a time, to the alternate control structure, where it will be executed by the uDMA controller.

Figure 9-6. Peripheral Scatter-Gather,  $\mu$ DMA Copy Sequence



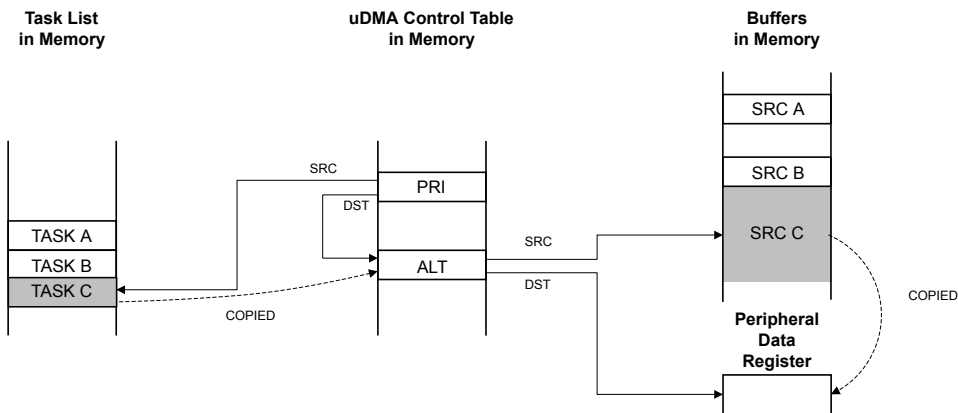
Using the channel's primary control structure, the  $\mu$ DMA controller copies task A configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the  $\mu$ DMA controller copies data from the source buffer A to the peripheral data register.



Using the channel's primary control structure, the  $\mu$ DMA controller copies task B configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the  $\mu$ DMA controller copies data from the source buffer B to the peripheral data register.



Using the channel's primary control structure, the  $\mu$ DMA controller copies task C configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the  $\mu$ DMA controller copies data from the source buffer C to the peripheral data register.

## 9.2.7 Transfer Size and Increment

The  $\mu$ DMA controller supports transfer data sizes of 8, 16, or 32 bits. The source and destination data size must be the same for any given transfer. The source and destination address can be auto-incremented by bytes, half-words, or words, or can be set to no increment. The source and destination address increment values can be set independently, and it is not necessary for the address increment to match the data size as long as the increment is the same or larger than the data size. For example, it is possible to perform a transfer using 8-bit data size, but using an address increment of full words (4 bytes). The data to be transferred must be aligned in memory according to the data size (8, 16, or 32 bits).

Table 9-5 on page 189 shows the configuration to read from a peripheral that supplies 8-bit data.

**Table 9-5.  $\mu$ DMA Read Example: 8-Bit Peripheral**

Field	Configuration
Source data size	8 bits
Destination data size	8 bits
Source address increment	No increment
Destination address increment	Byte
Source end pointer	Peripheral read FIFO register
Destination end pointer	End of the data buffer in memory

## 9.2.8 Peripheral Interface

Each peripheral that supports  $\mu$ DMA has a DMA single request and/or burst request signal that is asserted when the device is ready to transfer data. The request signal can be disabled or enabled by using the **DMA Channel Request Mask Set (DMAREQMASKSET)** and **DMA Channel Request Mask Clear (DMAREQMASKCLR)** registers. The DMA request signal is disabled, or masked, when the channel request mask bit is set. When the request is not masked, the DMA channel is configured correctly and enabled, and the peripheral asserts the DMA request signal, the  $\mu$ DMA controller will begin the transfer.

When a DMA transfer is complete, the  $\mu$ DMA controller asserts a DMA Done signal, which is routed through the interrupt vector of the peripheral. Therefore, if DMA is used to transfer data for a peripheral and interrupts are used, then the interrupt handler for that peripheral must be designed to handle the  $\mu$ DMA transfer completion interrupt. When DMA is enabled for a peripheral, the  $\mu$ DMA controller will mask the normal interrupts for a peripheral. This means that when a large amount of data is transferred using DMA, instead of receiving multiple interrupts from the peripheral as data flows, the processor will only receive one interrupt when the transfer is complete.

The interrupt request from the  $\mu$ DMA controller is automatically cleared when the interrupt handler is activated.

## 9.2.9 Software Request

There is a dedicated  $\mu$ DMA channel for software-initiated transfers. This channel also has a dedicated interrupt to signal completion of a DMA transfer. A transfer is initiated by software by first configuring and enabling the transfer, and then issuing a software request using the **DMA Channel Software Request (DMASWREQ)** register. For software-based transfers, the Auto transfer mode should be used.

It is possible to initiate a transfer on any channel using the **DMASWREQ** register. If a request is initiated by software using a peripheral DMA channel, then the completion interrupt will occur on the interrupt vector for the peripheral instead of the software interrupt vector. This means that any

channel may be used for software requests as long as the corresponding peripheral is not using  $\mu$ DMA.

## 9.2.10 Interrupts and Errors

When a DMA transfer is complete, the  $\mu$ DMA controller will generate a completion interrupt on the interrupt vector of the peripheral. If the transfer uses the software DMA channel, then the completion interrupt will occur on the dedicated software DMA interrupt vector.

If the  $\mu$ DMA controller encounters a bus or memory protection error as it attempts to perform a data transfer, it will disable the DMA channel that caused the error, and generate an interrupt on the  $\mu$ DMA Error interrupt vector. The processor can read the **DMA Bus Error Clear (DMAERRCLR)** register to determine if an error is pending. The `ERRCLR` bit will be set if an error occurred. The error can be cleared by writing a 1 to the `ERRCLR` bit.

Table 9-6 on page 190 shows the dedicated interrupt assignments for the  $\mu$ DMA controller.

**Table 9-6.  $\mu$ DMA Interrupt Assignments**

Interrupt	Assignment
46	$\mu$ DMA Software Channel Transfer
47	$\mu$ DMA Error

## 9.3 Initialization and Configuration

### 9.3.1 Module Initialization

Before the  $\mu$ DMA controller can be used, it must be enabled in the System Control block and in the peripheral. The location of the channel control structure must also be programmed.

The following steps should be performed one time during system initialization:

1. The  $\mu$ DMA peripheral must be enabled in the System Control block. To do this, set the `UDMA` bit of the System Control **RCGC2** register.
2. Enable the  $\mu$ DMA controller by setting the `MASTEREN` bit of the **DMA Configuration (DMACFG)** register.
3. Program the location of the channel control table by writing the base address of the table to the **DMA Channel Control Base Pointer (DMACTLBASE)** register. The base address must be aligned on a 1024-byte boundary.

### 9.3.2 Configuring a Memory-to-Memory Transfer

$\mu$ DMA channel 30 is dedicated for software-initiated transfers. However, any channel can be used for software-initiated, memory-to-memory transfer if the associated peripheral is not being used.

#### 9.3.2.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Set bit 30 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.
2. Set bit 30 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.

3. Set bit 30 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the  $\mu$ DMA controller to respond to single and burst requests.
4. Set bit 30 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the  $\mu$ DMA controller to recognize requests for this channel.

### 9.3.2.2 Configure the Channel Control Structure

Now the channel control structure must be configured.

This example will transfer 256 32-bit words from one memory buffer to another. Channel 30 is used for a software transfer, and the control structure for channel 30 is at offset 0x1E0 of the channel control table. The channel control structure for channel 30 is located at the offsets shown in Table 9-7 on page 191.

**Table 9-7. Channel Control Structure Offsets for Channel 30**

Offset	Description
Control Table Base + 0x1E0	Channel 30 Source End Pointer
Control Table Base + 0x1E4	Channel 30 Destination End Pointer
Control Table Base + 0x1E8	Channel 30 Control Word

#### **Configure the Source and Destination**

The source and destination end pointers must be set to the last address for the transfer (inclusive).

1. Set the source end pointer at offset 0x1E0 to the address of the source buffer + 0x3FC.
2. Set the destination end pointer at offset 0x1E4 to the address of the destination buffer + 0x3FC.

The control word at offset 0x1E8 must be programmed according to Table 9-8 on page 191.

**Table 9-8. Channel Control Word Configuration for Memory Transfer Example**

Field in DMACHCTL	Bits	Value	Description
DSTINC	31:30	2	32-bit destination address increment
DSTSIZE	29:28	2	32-bit destination data size
SRCINC	27:26	2	32-bit source address increment
SRCSIZE	25:24	2	32-bit source data size
reserved	23:18	0	Reserved
ARBSIZE	17:14	3	Arbitrates after 8 transfers
XFERSIZE	13:4	255	Transfer 256 items
NXTUSEBURST	3	0	N/A for this transfer type
XFERMODE	2:0	2	Use Auto-request transfer mode

### 9.3.2.3 Start the Transfer

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 30 of the **DMA Channel Enable Set (DMAENASET)** register.
2. Issue a transfer request by setting bit 30 of the **DMA Channel Software Request (DMASWREQ)** register.

The DMA transfer will now take place. If the interrupt is enabled, then the processor will be notified by interrupt when the transfer is complete. If needed, the status can be checked by reading bit 30 of the **DMAENASET** register. This bit will be automatically cleared when the transfer is complete. The status can also be checked by reading the **XFERMODE** field of the channel control word at offset 0x1E8. This field will automatically be set to 0 at the end of the transfer.

### 9.3.3 Configuring a Peripheral for Simple Transmit

This example will set up the  $\mu$ DMA controller to transmit a buffer of data to a peripheral. The peripheral has a transmit FIFO with a trigger level of 4. The example peripheral will use  $\mu$ DMA channel 7.

#### 9.3.3.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Set bit 7 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.
2. Set bit 7 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.
3. Set bit 7 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the  $\mu$ DMA controller to respond to single and burst requests.
4. Set bit 7 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the  $\mu$ DMA controller to recognize requests for this channel.

#### 9.3.3.2 Configure the Channel Control Structure

Now the channel control structure must be configured. This example will transfer 64 8-bit bytes from a memory buffer to the peripheral's transmit FIFO register. This example uses  $\mu$ DMA channel 7, and the control structure for channel 7 is at offset 0x070 of the channel control table. The channel control structure for channel 7 is located at the offsets shown in Table 9-9 on page 192.

**Table 9-9. Channel Control Structure Offsets for Channel 7**

Offset	Description
Control Table Base + 0x070	Channel 7 Source End Pointer
Control Table Base + 0x074	Channel 7 Destination End Pointer
Control Table Base + 0x078	Channel 7 Control Word

#### **Configure the Source and Destination**

The source and destination end pointers must be set to the last address for the transfer (inclusive). Since the peripheral pointer does not change, it simply points to the peripheral's data register.

1. Set the source end pointer at offset 0x070 to the address of the source buffer + 0x3F.
2. Set the destination end pointer at offset 0x074 to the address of the peripheral's transmit FIFO register.

The control word at offset 0x078 must be programmed according to Table 9-10 on page 193.



**Table 9-10. Channel Control Word Configuration for Peripheral Transmit Example**

Field in DMACHCTL	Bits	Value	Description
DSTINC	31:30	3	Destination address does not increment
DSTSIZE	29:28	0	8-bit destination data size
SRCINC	27:26	0	8-bit source address increment
SRCSIZE	25:24	0	8-bit source data size
reserved	23:18	0	Reserved
ARBSIZE	17:14	2	Arbitrates after 4 transfers
XFERSIZE	13:4	63	Transfer 64 items
NXTUSEBURST	3	0	N/A for this transfer type
XFERMODE	2:0	1	Use Basic transfer mode

**Note:** In this example, it is not important if the peripheral makes a single request or a burst request. Since the peripheral has a FIFO that will trigger at a level of 4, the arbitration size is set to 4. If the peripheral does make a burst request, then 4 bytes will be transferred, which is what the FIFO can accommodate. If the peripheral makes a single request (if there is any space in the FIFO), then one byte will be transferred at a time. If it is important to the application that transfers only be made in bursts, then the channel useburst `SET[n]` bit should be set by writing a 1 to bit 7 of the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register.

### 9.3.3.3 Start the Transfer

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 7 of the **DMA Channel Enable Set (DMAENASET)** register.

The  $\mu$ DMA controller is now configured for transfer on channel 7. The controller will make transfers to the peripheral whenever the peripheral asserts a DMA request. The transfers will continue until the entire buffer of 64 bytes has been transferred. When that happens, the  $\mu$ DMA controller will disable the channel and set the `XFERMODE` field of the channel control word to 0 (Stopped). The status of the transfer can be checked by reading bit 7 of the **DMA Channel Enable Set (DMAENASET)** register. This bit will be automatically cleared when the transfer is complete. The status can also be checked by reading the `XFERMODE` field of the channel control word at offset 0x078. This field will automatically be set to 0 at the end of the transfer.

If peripheral interrupts were enabled, then the peripheral interrupt handler would receive an interrupt when the entire transfer was complete.

## 9.3.4 Configuring a Peripheral for Ping-Pong Receive

This example will set up the  $\mu$ DMA controller to continuously receive 8-bit data from a peripheral into a pair of 64 byte buffers. The peripheral has a receive FIFO with a trigger level of 8. The example peripheral will use  $\mu$ DMA channel 8.

### 9.3.4.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Set bit 7 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.

2. Set bit 7 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.
3. Set bit 7 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the  $\mu$ DMA controller to respond to single and burst requests.
4. Set bit 7 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the  $\mu$ DMA controller to recognize requests for this channel.

### 9.3.4.2 Configure the Channel Control Structure

Now the channel control structure must be configured. This example will transfer 8-bit bytes from the peripheral's receive FIFO register into two memory buffers of 64 bytes each. As data is received, when one buffer is full, the  $\mu$ DMA controller switches to use the other.

To use Ping-Pong buffering, both primary and alternate channel control structures must be used. The primary control structure for channel 8 is at offset 0x080 of the channel control table, and the alternate channel control structure is at offset 0x280. The channel control structures for channel 8 are located at the offsets shown in Table 9-11 on page 194.

**Table 9-11. Primary and Alternate Channel Control Structure Offsets for Channel 8**

Offset	Description
Control Table Base + 0x080	Channel 8 Primary Source End Pointer
Control Table Base + 0x084	Channel 8 Primary Destination End Pointer
Control Table Base + 0x088	Channel 8 Primary Control Word
Control Table Base + 0x280	Channel 8 Alternate Source End Pointer
Control Table Base + 0x284	Channel 8 Alternate Destination End Pointer
Control Table Base + 0x288	Channel 8 Alternate Control Word

#### **Configure the Source and Destination**

The source and destination end pointers must be set to the last address for the transfer (inclusive). Since the peripheral pointer does not change, it simply points to the peripheral's data register. Both the primary and alternate sets of pointers must be configured.

1. Set the primary source end pointer at offset 0x080 to the address of the peripheral's receive buffer.
2. Set the primary destination end pointer at offset 0x084 to the address of ping-pong buffer A + 0x3F.
3. Set the alternate source end pointer at offset 0x280 to the address of the peripheral's receive buffer.
4. Set the alternate destination end pointer at offset 0x284 to the address of ping-pong buffer B + 0x3F.

The primary control word at offset 0x088, and the alternate control word at offset 0x288 must be programmed according to Table 9-10 on page 193. Both control words are initially programmed the same way.

1. Program the primary channel control word at offset 0x088 according to Table 9-12 on page 195.
2. Program the alternate channel control word at offset 0x288 according to Table 9-12 on page 195.

**Table 9-12. Channel Control Word Configuration for Peripheral Ping-Pong Receive Example**

Field in DMACHCTL	Bits	Value	Description
DSTINC	31:30	0	8-bit destination address increment
DSTSIZE	29:28	0	8-bit destination data size
SRCINC	27:26	3	Source address does not increment
SRCSIZE	25:24	0	8-bit source data size
reserved	23:18	0	Reserved
ARBSIZE	17:14	3	Arbitrates after 8 transfers
XFERSIZE	13:4	63	Transfer 64 items
NXTUSEBURST	3	0	N/A for this transfer type
XFERMODE	2:0	3	Use Ping-Pong transfer mode

**Note:** In this example, it is not important if the peripheral makes a single request or a burst request. Since the peripheral has a FIFO that will trigger at a level of 8, the arbitration size is set to 8. If the peripheral does make a burst request, then 8 bytes will be transferred, which is what the FIFO can accommodate. If the peripheral makes a single request (if there is any data in the FIFO), then one byte will be transferred at a time. If it is important to the application that transfers only be made in bursts, then the channel useburst `SET[n]` bit should be set by writing a 1 to bit 8 of the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register.

#### 9.3.4.3 Configure the Peripheral Interrupt

In order to use  $\mu$ DMA Ping-Pong mode, it is best to use an interrupt handler. (It is also possible to use ping-pong mode without interrupts by polling). The interrupt handler will be triggered after each buffer is complete.

1. Configure and enable an interrupt handler for the peripheral.

#### 9.3.4.4 Enable the $\mu$ DMA Channel

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 8 of the **DMA Channel Enable Set (DMAENASET)** register.

#### 9.3.4.5 Process Interrupts

The  $\mu$ DMA controller is now configured and enabled for transfer on channel 8. When the peripheral asserts the DMA request signal, the  $\mu$ DMA controller will make transfers into buffer A using the primary channel control structure. When the primary transfer to buffer A is complete, it will switch to the alternate channel control structure and make transfers into buffer B. At the same time, the primary channel control word mode field will be set to indicate Stopped, and an interrupt will be triggered.

When an interrupt is triggered, the interrupt handler must determine which buffer is complete and process the data, or set a flag that the data needs to be processed by non-interrupt buffer processing code. Then the next buffer transfer must be set up.

In the interrupt handler:

1. Read the primary channel control word at offset 0x088 and check the `XFERMODE` field. If the field is 0, this means buffer A is complete. If buffer A is complete, then:

- a. Process the newly received data in buffer A, or signal the buffer processing code that buffer A has data available.
  - b. Reprogram the primary channel control word at offset 0x88 according to Table 9-12 on page 195.
2. Read the alternate channel control word at offset 0x288 and check the `XFERMODE` field. If the field is 0, this means buffer B is complete. If buffer B is complete, then:
    - a. Process the newly received data in buffer B, or signal the buffer processing code that buffer B has data available.
    - b. Reprogram the alternate channel control word at offset 0x288 according to Table 9-12 on page 195.

## 9.4 Register Map

Table 9-13 on page 196 lists the  $\mu$ DMA channel control structures and registers. The channel control structure shows the layout of one entry in the channel control table. The channel control table is located in system memory, and the location is determined by the application, that is, the base address is n/a (not applicable). In the table below, the offset for the channel control structures is the offset from the entry in the channel control table. See “Channel Configuration” on page 179 and Table 9-3 on page 180 for a description of how the entries in the channel control table are located in memory. The  $\mu$ DMA register addresses are given as a hexadecimal increment, relative to the  $\mu$ DMA base address of 0x400F.F000.

**Table 9-13.  $\mu$ DMA Register Map**

Offset	Name	Type	Reset	Description	See page
<b><math>\mu</math>DMA Channel Control Structure</b>					
0x000	DMASRCENDP	R/W	-	DMA Channel Source Address End Pointer	198
0x004	DMADSTENDP	R/W	-	DMA Channel Destination Address End Pointer	199
0x008	DMACHCTL	R/W	-	DMA Channel Control Word	200
<b><math>\mu</math>DMA Registers</b>					
0x000	DMASTAT	RO	0x001F.0000	DMA Status	204
0x004	DMACFG	WO	-	DMA Configuration	206
0x008	DMACTLBASE	R/W	0x0000.0000	DMA Channel Control Base Pointer	207
0x00C	DMAALTBASE	RO	0x0000.0200	DMA Alternate Channel Control Base Pointer	208
0x010	DMAWAITSTAT	RO	0x0000.0000	DMA Channel Wait on Request Status	209
0x014	DMASWREQ	WO	-	DMA Channel Software Request	210
0x018	DMAUSEBURSTSET	R/W	0x0000.0000	DMA Channel Useburst Set	211
0x01C	DMAUSEBURSTCLR	WO	-	DMA Channel Useburst Clear	213
0x020	DMAREQMASKSET	R/W	0x0000.0000	DMA Channel Request Mask Set	214
0x024	DMAREQMASKCLR	WO	-	DMA Channel Request Mask Clear	216

Offset	Name	Type	Reset	Description	See page
0x028	DMAENASET	R/W	0x0000.0000	DMA Channel Enable Set	217
0x02C	DMAENACLRL	WO	-	DMA Channel Enable Clear	219
0x030	DMAALTSET	R/W	0x0000.0000	DMA Channel Primary Alternate Set	220
0x034	DMAALTCLR	WO	-	DMA Channel Primary Alternate Clear	222
0x038	DMAPRIOSET	R/W	0x0000.0000	DMA Channel Priority Set	223
0x03C	DMAPRIOCLR	WO	-	DMA Channel Priority Clear	225
0x04C	DMAERRCLR	R/W	0x0000.0000	DMA Bus Error Clear	226
0xFD0	DMAPeriphID4	RO	0x0000.0004	DMA Peripheral Identification 4	232
0xFE0	DMAPeriphID0	RO	0x0000.0030	DMA Peripheral Identification 0	228
0xFE4	DMAPeriphID1	RO	0x0000.00B2	DMA Peripheral Identification 1	229
0xFE8	DMAPeriphID2	RO	0x0000.000B	DMA Peripheral Identification 2	230
0xFEC	DMAPeriphID3	RO	0x0000.0000	DMA Peripheral Identification 3	231
0xFF0	DMAPrimeCellID0	RO	0x0000.000D	DMA PrimeCell Identification 0	233
0xFF4	DMAPrimeCellID1	RO	0x0000.00F0	DMA PrimeCell Identification 1	234
0xFF8	DMAPrimeCellID2	RO	0x0000.0005	DMA PrimeCell Identification 2	235
0xFFC	DMAPrimeCellID3	RO	0x0000.00B1	DMA PrimeCell Identification 3	236

## 9.5 $\mu$ DMA Channel Control Structure

The  $\mu$ DMA Channel Control Structure holds the DMA transfer settings for a DMA channel. Each channel has two control structures, which are located in a table in system memory. Refer to “Channel Configuration” on page 179 for an explanation of the Channel Control Table and the Channel Control Structure.

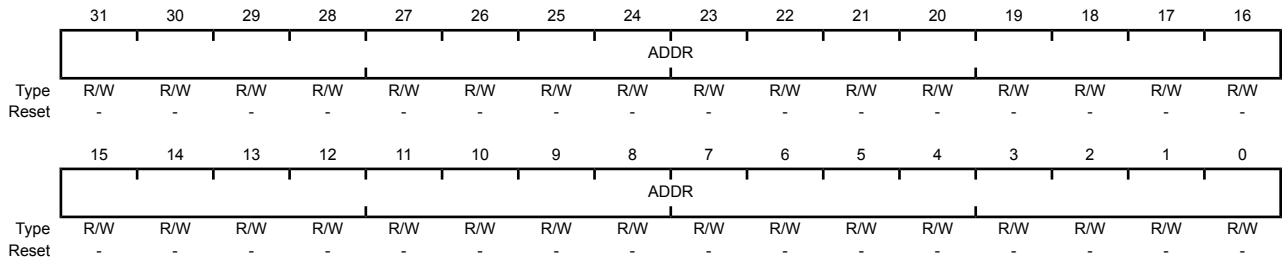
The channel control structure is one entry in the channel control table. There is a primary and alternate structure for each channel. The primary control structures are located at offsets 0x0, 0x10, 0x20 and so on. The alternate control structures are located at offsets 0x200, 0x210, 0x220, and so on.

## Register 1: DMA Channel Source Address End Pointer (DMASRCENDP), offset 0x000

**DMA Channel Source Address End Pointer (DMASRCENDP)** is part of the Channel Control Structure, and is used to specify the source address for a DMA transfer.

### DMA Channel Source Address End Pointer (DMASRCENDP)

Base n/a  
Offset 0x000  
Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31:0	ADDR	R/W	-	Source Address End Pointer  Points to the last address of the DMA transfer source (inclusive). If the source address is not incrementing, then this points at the source location itself (such as a peripheral data register).

## Register 2: DMA Channel Destination Address End Pointer (DMADSTENDP), offset 0x004

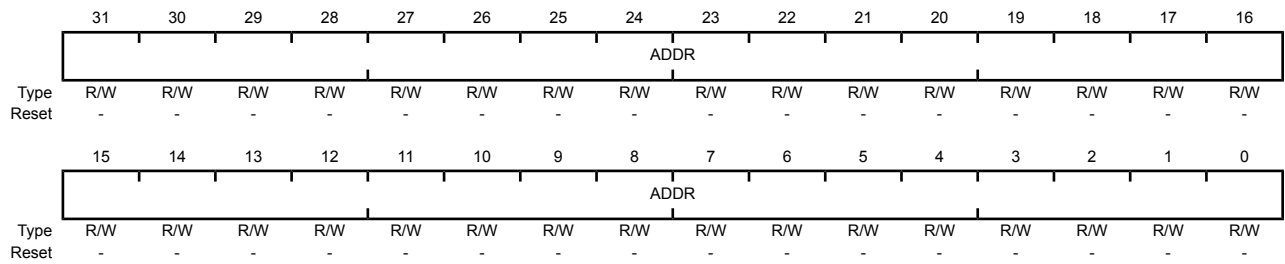
**DMA Channel Destination Address End Pointer (DMADSTENDP)** is part of the Channel Control Structure, and is used to specify the destination address for a DMA transfer.

### DMA Channel Destination Address End Pointer (DMADSTENDP)

Base n/a

Offset 0x004

Type R/W, reset -



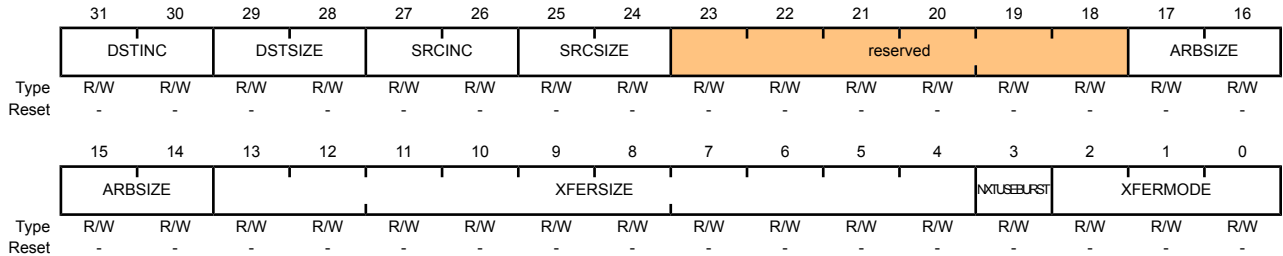
Bit/Field	Name	Type	Reset	Description
31:0	ADDR	R/W	-	Destination Address End Pointer  Points to the last address of the DMA transfer destination (inclusive). If the destination address is not incrementing, then this points at the destination location itself (such as a peripheral data register).

### Register 3: DMA Channel Control Word (DMACHCTL), offset 0x008

DMA Channel Control Word (DMACHCTL) is part of the Channel Control Structure, and is used to specify parameters of a DMA transfer.

#### DMA Channel Control Word (DMACHCTL)

Base n/a  
Offset 0x008  
Type R/W, reset -



Bit/Field	Name	Type	Reset	Description												
31:30	DSTINC	R/W	-	<p>Destination Address Increment</p> <p>Sets the bits to control the destination address increment.</p> <p>The address increment value must be equal or greater than the value of the destination size (DSTSIZE).</p> <p>Value Description</p> <table border="0"> <tr> <td>0x0</td> <td>Byte</td> <td>Increment by 8-bit locations.</td> </tr> <tr> <td>0x1</td> <td>Half-word</td> <td>Increment by 16-bit locations.</td> </tr> <tr> <td>0x2</td> <td>Word</td> <td>Increment by 32-bit locations.</td> </tr> <tr> <td>0x3</td> <td>No increment</td> <td>Address remains set to the value of the Destination Address End Pointer (DMADSTENDP) for the channel.</td> </tr> </table>	0x0	Byte	Increment by 8-bit locations.	0x1	Half-word	Increment by 16-bit locations.	0x2	Word	Increment by 32-bit locations.	0x3	No increment	Address remains set to the value of the Destination Address End Pointer (DMADSTENDP) for the channel.
0x0	Byte	Increment by 8-bit locations.														
0x1	Half-word	Increment by 16-bit locations.														
0x2	Word	Increment by 32-bit locations.														
0x3	No increment	Address remains set to the value of the Destination Address End Pointer (DMADSTENDP) for the channel.														
29:28	DSTSIZE	R/W	-	<p>Destination Data Size</p> <p>Sets the destination item data size.</p> <p><b>Note:</b> You must set DSTSIZE to be the same as SRCSIZE.</p> <p>Value Description</p> <table border="0"> <tr> <td>0x0</td> <td>Byte</td> <td>8-bit data size.</td> </tr> <tr> <td>0x1</td> <td>Half-word</td> <td>16-bit data size.</td> </tr> <tr> <td>0x2</td> <td>Word</td> <td>32-bit data size.</td> </tr> <tr> <td>0x3</td> <td>Reserved</td> <td></td> </tr> </table>	0x0	Byte	8-bit data size.	0x1	Half-word	16-bit data size.	0x2	Word	32-bit data size.	0x3	Reserved	
0x0	Byte	8-bit data size.														
0x1	Half-word	16-bit data size.														
0x2	Word	32-bit data size.														
0x3	Reserved															



Bit/Field	Name	Type	Reset	Description										
27:26	SRCINC	R/W	-	<p>Source Address Increment</p> <p>Sets the bits to control the source address increment.</p> <p>The address increment value must be equal or greater than the value of the source size (SRCSIZE).</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Byte Increment by 8-bit locations.</td> </tr> <tr> <td>0x1</td> <td>Half-word Increment by 16-bit locations.</td> </tr> <tr> <td>0x2</td> <td>Word Increment by 32-bit locations.</td> </tr> <tr> <td>0x3</td> <td>No increment Address remains set to the value of the Source Address End Pointer (DMASRCENDE) for the channel.</td> </tr> </tbody> </table>	Value	Description	0x0	Byte Increment by 8-bit locations.	0x1	Half-word Increment by 16-bit locations.	0x2	Word Increment by 32-bit locations.	0x3	No increment Address remains set to the value of the Source Address End Pointer (DMASRCENDE) for the channel.
Value	Description													
0x0	Byte Increment by 8-bit locations.													
0x1	Half-word Increment by 16-bit locations.													
0x2	Word Increment by 32-bit locations.													
0x3	No increment Address remains set to the value of the Source Address End Pointer (DMASRCENDE) for the channel.													
25:24	SRCSIZE	R/W	-	<p>Source Data Size</p> <p>Sets the source item data size.</p> <p><b>Note:</b> You must set DSTSIZE to be the same as SRCSIZE.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Byte 8-bit data size.</td> </tr> <tr> <td>0x1</td> <td>Half-word 16-bit data size.</td> </tr> <tr> <td>0x2</td> <td>Word 32-bit data size.</td> </tr> <tr> <td>0x3</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Description	0x0	Byte 8-bit data size.	0x1	Half-word 16-bit data size.	0x2	Word 32-bit data size.	0x3	Reserved
Value	Description													
0x0	Byte 8-bit data size.													
0x1	Half-word 16-bit data size.													
0x2	Word 32-bit data size.													
0x3	Reserved													
23:18	reserved	R/W	-	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										

Bit/Field	Name	Type	Reset	Description																										
17:14	ARBSIZE	R/W	-	<p>Arbitration Size</p> <p>Sets the number of DMA transfers that can occur before the controller re-arbitrates. The possible arbitration rate settings represent powers of 2 and are shown below.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>1 Transfer</td> </tr> <tr> <td></td> <td>Arbitrates after each DMA transfer.</td> </tr> <tr> <td>0x1</td> <td>2 Transfers</td> </tr> <tr> <td>0x2</td> <td>4 Transfers</td> </tr> <tr> <td>0x3</td> <td>8 Transfers</td> </tr> <tr> <td>0x4</td> <td>16 Transfers</td> </tr> <tr> <td>0x5</td> <td>32 Transfers</td> </tr> <tr> <td>0x6</td> <td>64 Transfers</td> </tr> <tr> <td>0x7</td> <td>128 Transfers</td> </tr> <tr> <td>0x8</td> <td>256 Transfers</td> </tr> <tr> <td>0x9</td> <td>512 Transfers</td> </tr> <tr> <td>0xA-0xF</td> <td>1024 Transfers</td> </tr> </tbody> </table> <p>This means that no arbitration occurs during the DMA transfer because the maximum transfer size is 1024.</p>	Value	Description	0x0	1 Transfer		Arbitrates after each DMA transfer.	0x1	2 Transfers	0x2	4 Transfers	0x3	8 Transfers	0x4	16 Transfers	0x5	32 Transfers	0x6	64 Transfers	0x7	128 Transfers	0x8	256 Transfers	0x9	512 Transfers	0xA-0xF	1024 Transfers
Value	Description																													
0x0	1 Transfer																													
	Arbitrates after each DMA transfer.																													
0x1	2 Transfers																													
0x2	4 Transfers																													
0x3	8 Transfers																													
0x4	16 Transfers																													
0x5	32 Transfers																													
0x6	64 Transfers																													
0x7	128 Transfers																													
0x8	256 Transfers																													
0x9	512 Transfers																													
0xA-0xF	1024 Transfers																													
13:4	XFERSIZE	R/W	-	<p>Transfer Size (minus 1)</p> <p>Sets the total number of items to transfer. The value of this field is 1 less than the number to transfer (value 0 means transfer 1 item). The maximum value for this 10-bit field is 1023 which represents a transfer size of 1024 items.</p> <p>The transfer size is the number of items, not the number of bytes. If the data size is 32 bits, then this value is the number of 32-bit words to transfer.</p> <p>The controller updates this field immediately prior to it entering the arbitration process, so it contains the number of outstanding DMA items that are necessary to complete the DMA cycle.</p>																										
3	NXTUSEBURST	R/W	-	<p>Next Useburst</p> <p>Controls whether the useburst <code>SET[n]</code> bit is automatically set for the last transfer of a peripheral scatter-gather operation. Normally, for the last transfer, if the number of remaining items to transfer is less than the arbitration size, the controller will use single transfers to complete the transaction. If this bit is set, then the controller will only use a burst transfer to complete the last transfer.</p>																										

Bit/Field	Name	Type	Reset	Description
2:0	XFERMODE	R/W	-	<p>DMA Transfer Mode</p> <p>Since this register is in system RAM, it has no reset value. Therefore, this field should be initialized to 0 before the channel is enabled.</p> <p>The operating mode of the DMA cycle. Refer to "Transfer Modes" on page 181 for a detailed explanation of transfer modes.</p> <p>Value Description</p> <p>0x0 Stop Channel is stopped, or configuration data is invalid.</p> <p>0x1 Basic The controller must receive a new request, prior to it entering the arbitration process, to enable the DMA cycle to complete.</p> <p>0x2 Auto-Request The initial request (software- or peripheral-initiated) is sufficient to complete the entire transfer of <i>XFERSIZE</i> items without any further requests.</p> <p>0x3 Ping-Pong The controller performs a DMA cycle using one of the channel control structures. After the DMA cycle completes, it performs a DMA cycle using the other channel control structure. After the next DMA cycle completes (and provided that the host processor has updated the original channel control data structure), it performs a DMA cycle using the original channel control data structure. The controller continues to perform DMA cycles until it either reads an invalid data structure or the host processor changes this field to 0x1 or 0x2. See "Ping-Pong" on page 181.</p> <p>0x4 Memory Scatter-Gather When the controller operates in Memory Scatter-Gather mode, you must only use this value in the primary channel control data structure. See "Memory Scatter-Gather" on page 182.</p> <p>0x5 Alternate Memory Scatter-Gather When the controller operates in Memory Scatter-Gather mode, you must only use this value in the alternate channel control data structure.</p> <p>0x6 Peripheral Scatter-Gather When the controller operates in Peripheral Scatter-Gather mode, you must only use this value in the primary channel control data structure. See "Peripheral Scatter-Gather" on page 186.</p> <p>0x7 Alternate Peripheral Scatter-Gather When the controller operates in Peripheral Scatter-Gather mode, you must only use this value in the alternate channel control data structure.</p>

## 9.6 $\mu$ DMA Register Descriptions

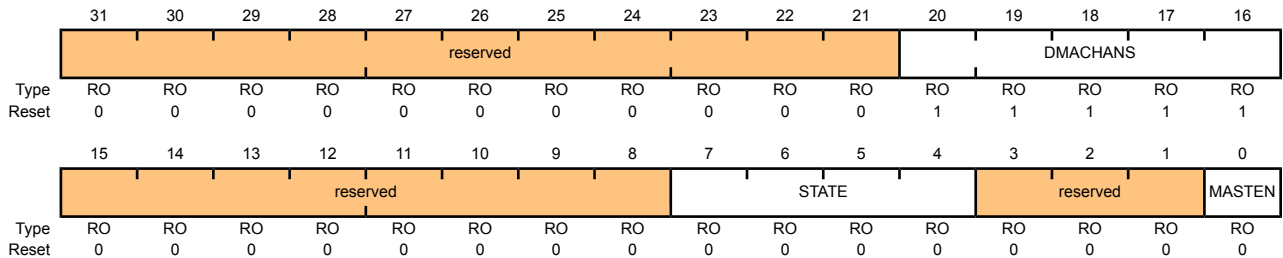
The register addresses given are relative to the  $\mu$ DMA base address of 0x400F.F000.

### Register 4: DMA Status (DMASTAT), offset 0x000

The **DMA Status (DMASTAT)** register returns the status of the controller. You cannot read this register when the controller is in the reset state.

#### DMA Status (DMASTAT)

Base 0x400F.F000  
 Offset 0x000  
 Type RO, reset 0x001F.0000



Bit/Field	Name	Type	Reset	Description
31:21	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
20:16	DMACHANS	RO	0x1F	Available DMA Channels Minus 1  This bit contains a value equal to the number of DMA channels the controller is configured to use, minus one. That is, 32 DMA channels.
15:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description																																						
7:4	STATE	RO	0x00	<p>Control State Machine State</p> <p>Current state of the control state machine. State can be one of the following.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Idle</td> </tr> <tr> <td>0x1</td> <td>Read Chan Control Data</td> </tr> <tr> <td></td> <td>Reading channel controller data.</td> </tr> <tr> <td>0x2</td> <td>Read Source End Ptr</td> </tr> <tr> <td></td> <td>Reading source end pointer.</td> </tr> <tr> <td>0x3</td> <td>Read Dest End Ptr</td> </tr> <tr> <td></td> <td>Reading destination end pointer.</td> </tr> <tr> <td>0x4</td> <td>Read Source Data</td> </tr> <tr> <td></td> <td>Reading source data.</td> </tr> <tr> <td>0x5</td> <td>Write Dest Data</td> </tr> <tr> <td></td> <td>Writing destination data.</td> </tr> <tr> <td>0x6</td> <td>Wait for Req Clear</td> </tr> <tr> <td></td> <td>Waiting for DMA request to clear.</td> </tr> <tr> <td>0x7</td> <td>Write Chan Control Data</td> </tr> <tr> <td></td> <td>Writing channel controller data.</td> </tr> <tr> <td>0x8</td> <td>Stalled</td> </tr> <tr> <td>0x9</td> <td>Done</td> </tr> <tr> <td>0xA-0xF</td> <td>Undefined</td> </tr> </tbody> </table>	Value	Description	0x0	Idle	0x1	Read Chan Control Data		Reading channel controller data.	0x2	Read Source End Ptr		Reading source end pointer.	0x3	Read Dest End Ptr		Reading destination end pointer.	0x4	Read Source Data		Reading source data.	0x5	Write Dest Data		Writing destination data.	0x6	Wait for Req Clear		Waiting for DMA request to clear.	0x7	Write Chan Control Data		Writing channel controller data.	0x8	Stalled	0x9	Done	0xA-0xF	Undefined
Value	Description																																									
0x0	Idle																																									
0x1	Read Chan Control Data																																									
	Reading channel controller data.																																									
0x2	Read Source End Ptr																																									
	Reading source end pointer.																																									
0x3	Read Dest End Ptr																																									
	Reading destination end pointer.																																									
0x4	Read Source Data																																									
	Reading source data.																																									
0x5	Write Dest Data																																									
	Writing destination data.																																									
0x6	Wait for Req Clear																																									
	Waiting for DMA request to clear.																																									
0x7	Write Chan Control Data																																									
	Writing channel controller data.																																									
0x8	Stalled																																									
0x9	Done																																									
0xA-0xF	Undefined																																									
3:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.																																						
0	MASTEN	RO	0x00	<p>Master Enable</p> <p>Returns status of the controller.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disabled</td> </tr> <tr> <td>1</td> <td>Enabled</td> </tr> </tbody> </table>	Value	Description	0	Disabled	1	Enabled																																
Value	Description																																									
0	Disabled																																									
1	Enabled																																									

## Register 5: DMA Configuration (DMACFG), offset 0x004

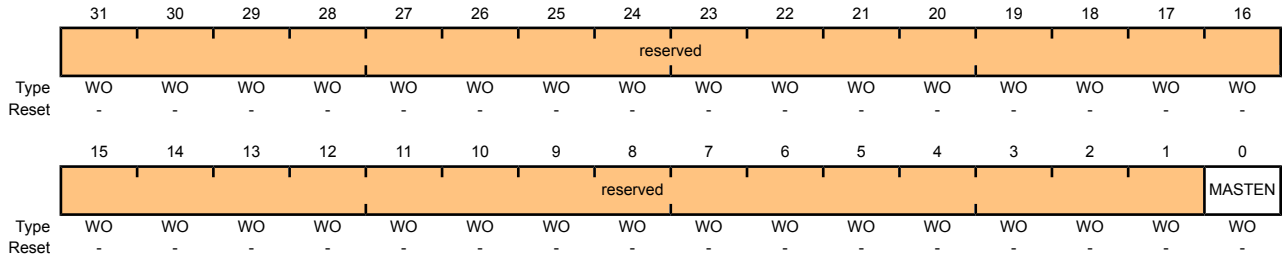
The **DMACFG** register controls the configuration of the controller.

### DMA Configuration (DMACFG)

Base 0x400F.F000

Offset 0x004

Type WO, reset -



Bit/Field	Name	Type	Reset	Description
31:1	reserved	WO	-	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	MASTEN	WO	-	Controller Master Enable Enables the controller.
				Value Description
				0 Disables
				1 Enables

**Register 6: DMA Channel Control Base Pointer (DMACTLBASE), offset 0x008**

The **DMACTLBASE** register must be configured so that the base pointer points to a location in system memory.

The amount of system memory that you must assign to the controller depends on the number of DMA channels used and whether you configure it to use the alternate channel control data structure. See “Channel Configuration” on page 179 for details about the Channel Control Table. The base address must be aligned on a 1024-byte boundary. You cannot read this register when the controller is in the reset state.

**DMA Channel Control Base Pointer (DMACTLBASE)**

Base 0x400F.F000

Offset 0x008

Type R/W, reset 0x0000.0000

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

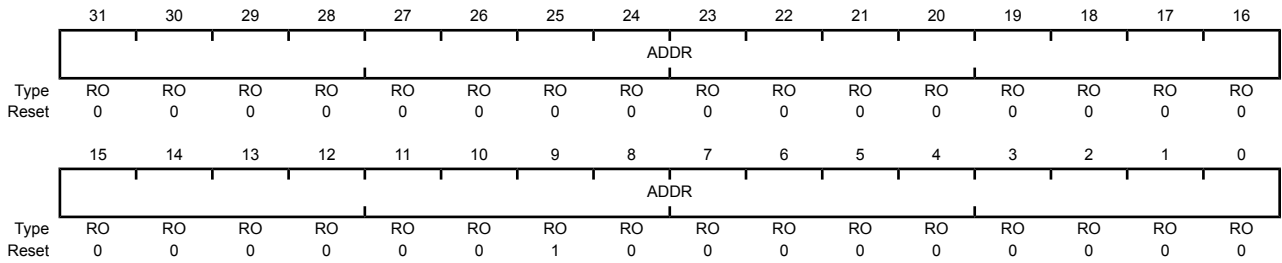
Bit/Field	Name	Type	Reset	Description
31:10	ADDR	R/W	0x00	Channel Control Base Address Pointer to the base address of the channel control table. The base address must be 1024-byte aligned.
9:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

### Register 7: DMA Alternate Channel Control Base Pointer (DMAALTBASE), offset 0x00C

The **DMAALTBASE** register returns the base address of the alternate channel control data. This register removes the necessity for application software to calculate the base address of the alternate channel control structures. You cannot read this register when the controller is in the reset state.

#### DMA Alternate Channel Control Base Pointer (DMAALTBASE)

Base 0x400F.F000  
 Offset 0x00C  
 Type RO, reset 0x0000.0200



Bit/Field	Name	Type	Reset	Description
31:0	ADDR	RO	0x200	Alternate Channel Address Pointer Provides the base address of the alternate channel control structures.



## Register 8: DMA Channel Wait on Request Status (DMAWAITSTAT), offset 0x010

This read-only register indicates that the  $\mu$ DMA channel is waiting on a request. A peripheral can pull this Low to hold off the  $\mu$ DMA from performing a single request until the peripheral is ready for a burst request. The use of this feature is dependent on the design of the peripheral and is used to enhance performance of the  $\mu$ DMA with that peripheral. You cannot read this register when the controller is in the reset state.

### DMA Channel Wait on Request Status (DMAWAITSTAT)

Base 0x400F.F000  
Offset 0x010  
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WAITREQ[n]															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WAITREQ[n]															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	WAITREQ[n]	RO	0x00	Channel [n] Wait Status

Channel wait on request status. For each channel 0 through 31, a 1 in the corresponding bit field indicates that the channel is waiting on a request.

### Register 9: DMA Channel Software Request (DMASWREQ), offset 0x014

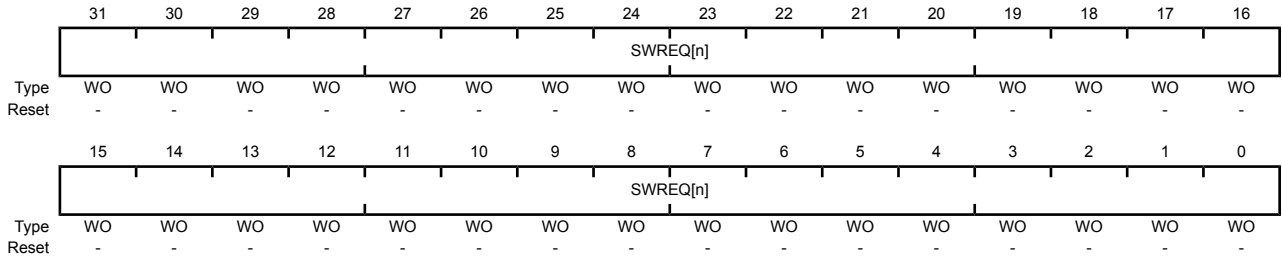
Each bit of the **DMASWREQ** register represents the corresponding DMA channel. When you set a bit, it generates a request for the specified DMA channel.

#### DMA Channel Software Request (DMASWREQ)

Base 0x400F.F000

Offset 0x014

Type WO, reset -



Bit/Field	Name	Type	Reset	Description
-----------	------	------	-------	-------------

31:0	SWREQ[n]	WO	-	Channel [n] Software Request
------	----------	----	---	------------------------------

For each channel 0 through 31, write a 1 to the corresponding bit field to generate a software DMA request for that DMA channel. Writing a 0 does not create a DMA request for the corresponding channel.

## Register 10: DMA Channel Useburst Set (DMAUSEBURSTSET), offset 0x018

Each bit of the **DMAUSEBURSTSET** register represents the corresponding DMA channel. Writing a 1 disables the peripheral's single request input from generating requests, and therefore only the peripheral's burst request generates requests. Reading the register returns the status of useburst.

When there are fewer items remaining to transfer than the arbitration (burst) size, the controller automatically clears the useburst bit to 0. This enables the remaining items to transfer using single requests. This bit should not be set for a peripheral's channel that does not support the burst request model.

Refer to "Request Types" on page 178 for more details about request types.

### DMAUSEBURSTSET Reads

#### DMA Channel Useburst Set (DMAUSEBURSTSET)

Base 0x400F.F000

Offset 0x018

Type RO, reset 0x0000.0000

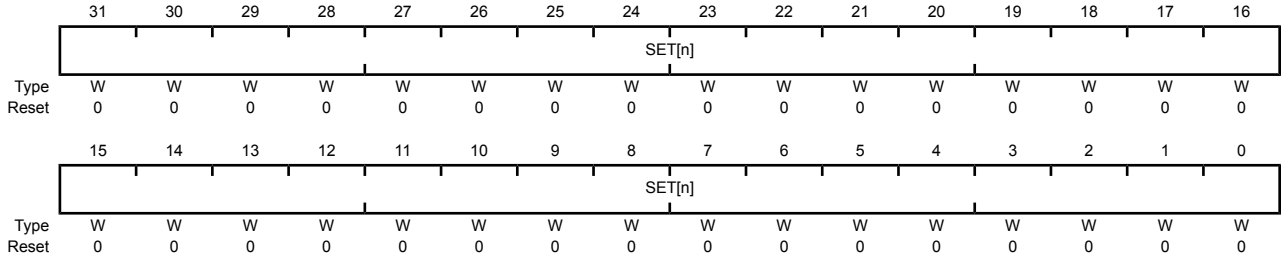
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SET[n]															
Type	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SET[n]															
Type	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R	0x00	Channel [n] Useburst Set Returns the useburst status of channel [n].
				Value Description
				0 Single and Burst DMA channel [n] responds to single or burst requests.
				1 Burst Only DMA channel [n] responds only to burst requests.

**DMAUSEBURSTSET Writes**

DMA Channel Useburst Set (DMAUSEBURSTSET)

Base 0x400F.F000  
 Offset 0x018  
 Type WO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	W	0x00	Channel [n] Useburst Set Sets useburst bit on channel [n].

Value	Description
0	No Effect Use the <b>DMAUSEBURSTCLR</b> register to clear bit [n] to 0.
1	Burst Only DMA channel [n] responds only to burst requests.

**Register 11: DMA Channel Useburst Clear (DMAUSEBURSTCLR), offset 0x01C**

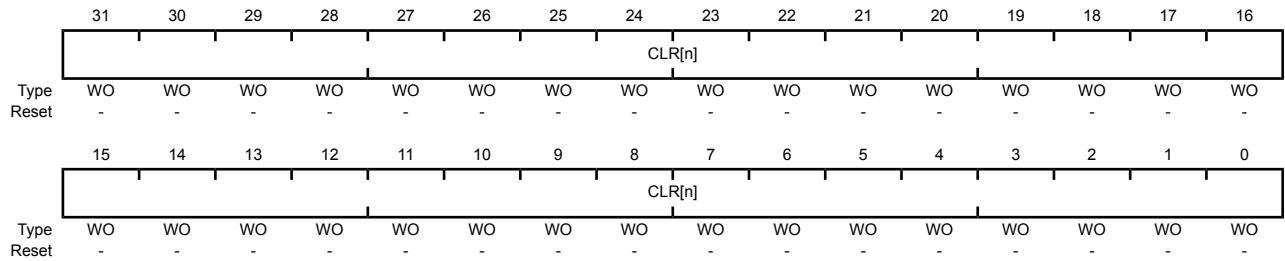
Each bit of the **DMAUSEBURSTCLR** register represents the corresponding DMA channel. Writing a 1 enables `dma_sreq[n]` to generate requests.

## DMA Channel Useburst Clear (DMAUSEBURSTCLR)

Base 0x400F.F000

Offset 0x01C

Type WO, reset -



Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Channel [n] Useburst Clear Clears useburst bit on channel [n].
				Value Description
				0 No Effect
				Use the <b>DMAUSEBURSTSET</b> to set bit [n] to 1.
				1 Single and Burst
				DMA channel [n] responds to single and burst requests.

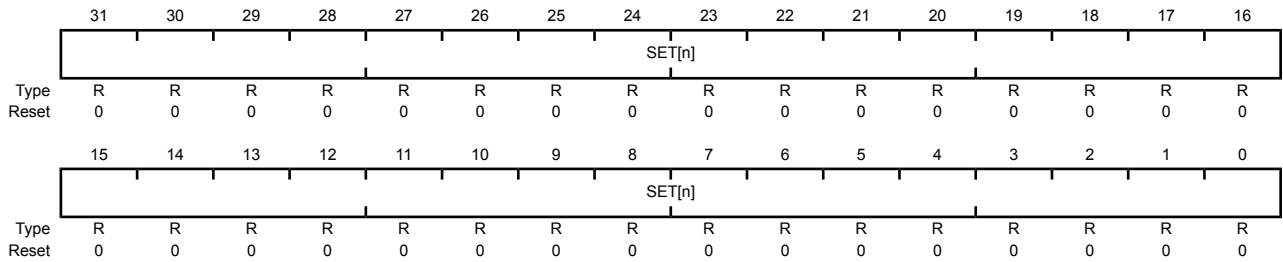
## Register 12: DMA Channel Request Mask Set (DMAREQMASKSET), offset 0x020

Each bit of the **DMAREQMASKSET** register represents the corresponding DMA channel. Writing a 1 disables DMA requests for the channel. Reading the register returns the request mask status. When a  $\mu$ DMA channel's request is masked, that means the peripheral can no longer request  $\mu$ DMA transfers. The channel can then be used for software-initiated transfers.

### DMAREQMASKSET Reads

#### DMA Channel Request Mask Set (DMAREQMASKSET)

Base 0x400F.F000  
Offset 0x020  
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R	0x00	Channel [n] Request Mask Set Returns the channel request mask status.

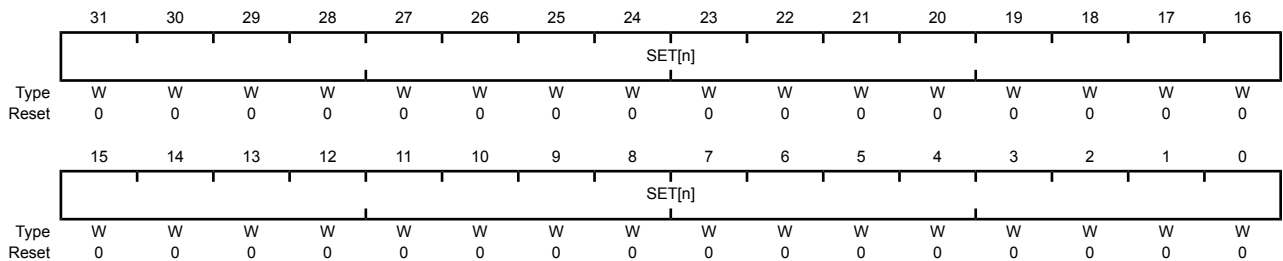
  

Value	Description
0	Enabled External requests are not masked for channel [n].
1	Masked External requests are masked for channel [n].

### DMAREQMASKSET Writes

#### DMA Channel Request Mask Set (DMAREQMASKSET)

Base 0x400F.F000  
Offset 0x020  
Type WO, reset 0x0000.0000



---

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	W	0x00	Channel [n] Request Mask Set Masks (disables) the corresponding channel [n] from generating DMA requests.  Value Description 0 No Effect Use the <b>DMAREQMASKCLR</b> register to clear the request mask. 1 Masked Masks (disables) DMA requests on channel [n].

### Register 13: DMA Channel Request Mask Clear (DMAREQMASKCLR), offset 0x024

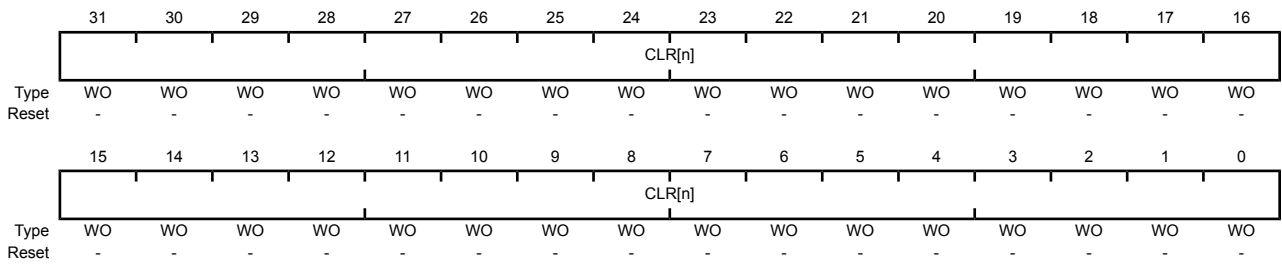
Each bit of the **DMAREQMASKCLR** register represents the corresponding DMA channel. Writing a 1 clears the request mask for the channel, and enables the channel to receive DMA requests.

#### DMA Channel Request Mask Clear (DMAREQMASKCLR)

Base 0x400F.F000

Offset 0x024

Type WO, reset -



Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Channel [n] Request Mask Clear Set the appropriate bit to clear the DMA request mask for channel [n]. This will enable DMA requests for the channel.
				Value Description
				0 No Effect Use the <b>DMAREQMASKSET</b> register to set the request mask.
				1 Clear Mask Clears the request mask for the DMA channel. This enables DMA requests for the channel.



## Register 14: DMA Channel Enable Set (DMAENASET), offset 0x028

Each bit of the **DMAENASET** register represents the corresponding DMA channel. Writing a 1 enables the DMA channel. Reading the register returns the enable status of the channels. If a channel is enabled but the request mask is set (**DMAREQMASKSET**), then the channel can be used for software-initiated transfers.

### DMAENASET Reads

#### DMA Channel Enable Set (DMAENASET)

Base 0x400F.F000  
Offset 0x028  
Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R	0x00	Channel [n] Enable Set Returns the enable status of the channels.
	Value	Description		
	0	Disabled		
	1	Enabled		

### DMAENASET Writes

#### DMA Channel Enable Set (DMAENASET)

Base 0x400F.F000  
Offset 0x028  
Type WO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description										
31:0	CHENSET[n]	W	0x00	<p>Channel [n] Enable Set</p> <p>Enables the corresponding channels.</p> <p><b>Note:</b> The controller disables a channel when it completes the DMA cycle.</p> <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0</td><td>No Effect</td></tr><tr><td></td><td>Use the <b>DMAENACL</b>R register to disable a channel.</td></tr><tr><td>1</td><td>Enable</td></tr><tr><td></td><td>Enables channel [n].</td></tr></tbody></table>	Value	Description	0	No Effect		Use the <b>DMAENACL</b> R register to disable a channel.	1	Enable		Enables channel [n].
Value	Description													
0	No Effect													
	Use the <b>DMAENACL</b> R register to disable a channel.													
1	Enable													
	Enables channel [n].													

## Register 15: DMA Channel Enable Clear (DMAENACLRL), offset 0x02C

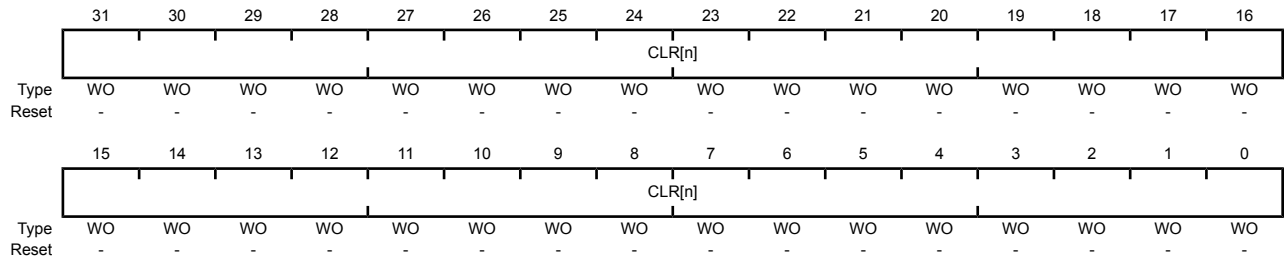
Each bit of the **DMAENACLRL** register represents the corresponding DMA channel. Writing a 1 disables the specified DMA channel.

### DMA Channel Enable Clear (DMAENACLRL)

Base 0x400F.F000

Offset 0x02C

Type WO, reset -



Bit/Field	Name	Type	Reset	Description
-----------	------	------	-------	-------------

31:0	CLR[n]	WO	-	Clear Channel [n] Enable
------	--------	----	---	--------------------------

Set the appropriate bit to disable the corresponding DMA channel.

**Note:** The controller disables a channel when it completes the DMA cycle.

Value	Description
-------	-------------

0	No Effect
---	-----------

Use the **DMAENASET** register to enable DMA channels.

1	Disable
---	---------

Disables channel [n].

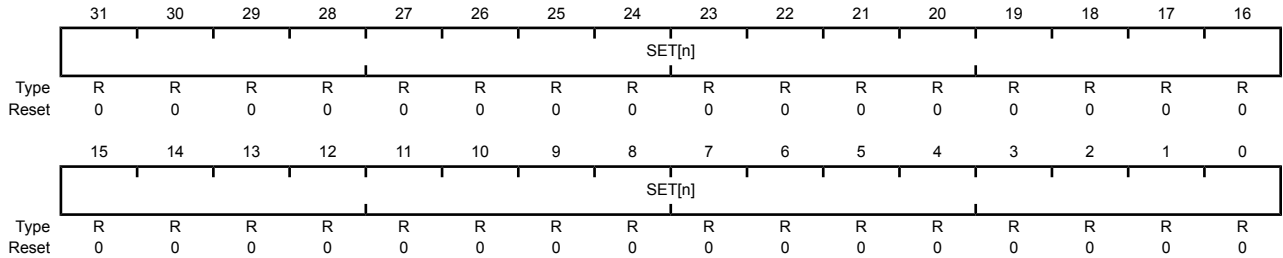
## Register 16: DMA Channel Primary Alternate Set (DMAALTSET), offset 0x030

Each bit of the **DMAALTSET** register represents the corresponding DMA channel. Writing a 1 configures the DMA channel to use the alternate control data structure. Reading the register returns the status of which control data structure is in use for the corresponding DMA channel.

### DMAALTSET Reads

#### DMA Channel Primary Alternate Set (DMAALTSET)

Base 0x400F.F000  
Offset 0x030  
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
-----------	------	------	-------	-------------

31:0	SET[n]	R	0x00	Channel [n] Alternate Set
------	--------	---	------	---------------------------

Returns the channel control data structure status.

Value Description

0 Primary

DMA channel [n] is using the primary control structure.

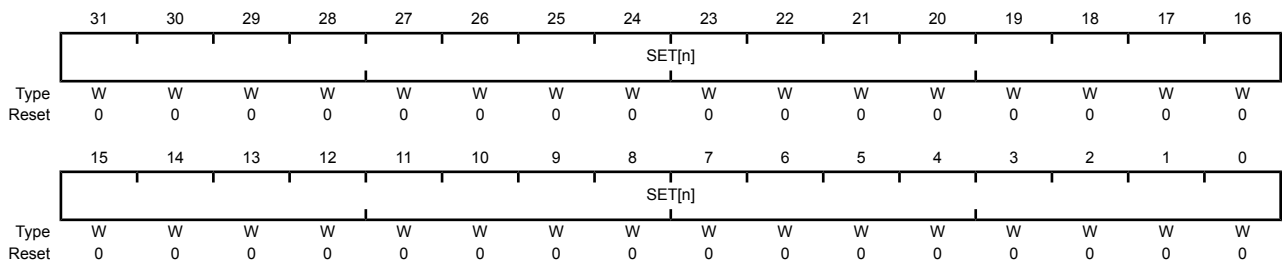
1 Alternate

DMA channel [n] is using the alternate control structure.

### DMAALTSET Writes

#### DMA Channel Primary Alternate Set (DMAALTSET)

Base 0x400F.F000  
Offset 0x030  
Type WO, reset 0x0000.0000



---

Bit/Field	Name	Type	Reset	Description										
31:0	SET[n]	W	0x00	<p>Channel [n] Alternate Set</p> <p>Selects the alternate channel control data structure for the corresponding DMA channel.</p> <p><b>Note:</b> For Ping-Pong and Scatter-Gather DMA cycle types, the controller automatically sets these bits to select the alternate channel control data structure.</p> <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0</td><td>No Effect</td></tr><tr><td></td><td>Use the <b>DMAALTCLR</b> register to set bit [n] to 0.</td></tr><tr><td>1</td><td>Alternate</td></tr><tr><td></td><td>Selects the alternate control data structure for channel [n].</td></tr></tbody></table>	Value	Description	0	No Effect		Use the <b>DMAALTCLR</b> register to set bit [n] to 0.	1	Alternate		Selects the alternate control data structure for channel [n].
Value	Description													
0	No Effect													
	Use the <b>DMAALTCLR</b> register to set bit [n] to 0.													
1	Alternate													
	Selects the alternate control data structure for channel [n].													

## Register 17: DMA Channel Primary Alternate Clear (DMAALTCLR), offset 0x034

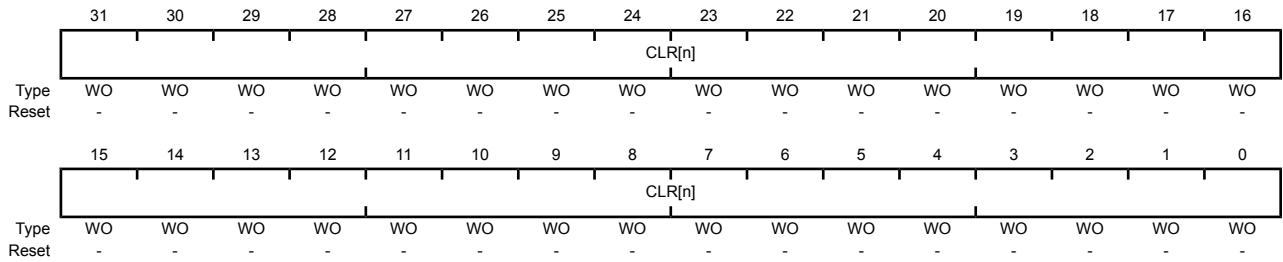
Each bit of the **DMAALTCLR** register represents the corresponding DMA channel. Writing a 1 configures the DMA channel to use the primary control data structure.

### DMA Channel Primary Alternate Clear (DMAALTCLR)

Base 0x400F.F000

Offset 0x034

Type WO, reset -



Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Channel [n] Alternate Clear

Set the appropriate bit to select the primary control data structure for the corresponding DMA channel.

**Note:** For Ping-Pong and Scatter-Gather DMA cycle types, the controller sets these bits to select the primary channel control data structure.

Value	Description
0	No Effect Use the <b>DMAALTSET</b> register to select the alternate control data structure.
1	Primary Selects the primary control data structure for channel [n].

## Register 18: DMA Channel Priority Set (DMAPRIOSET), offset 0x038

Each bit of the the **DMAPRIOSET** register represents the corresponding DMA channel. Writing a 1 configures the DMA channel to have a high priority level. Reading the register returns the status of the channel priority mask.

### DMAPRIOSET Reads

#### DMA Channel Priority Set (DMAPRIOSET)

Base 0x400F.F000  
Offset 0x038  
Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R	0x00	Channel [n] Priority Set Returns the channel priority status.

Value	Description
0	Default Priority DMA channel [n] is using the default priority level.
1	High Priority DMA channel [n] is using a High Priority level.

### DMAPRIOSET Writes

#### DMA Channel Priority Set (DMAPRIOSET)

Base 0x400F.F000  
Offset 0x038  
Type WO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	W	0x00	Channel [n] Priority Set Sets the channel priority to high.
				Value Description
				0 No Effect Use the <b>DMAPRIOCLR</b> register to set channel [n] to the default priority level.
				1 High Priority Sets DMA channel [n] to a High Priority level.



## Register 19: DMA Channel Priority Clear (DMAPRIOCLR), offset 0x03C

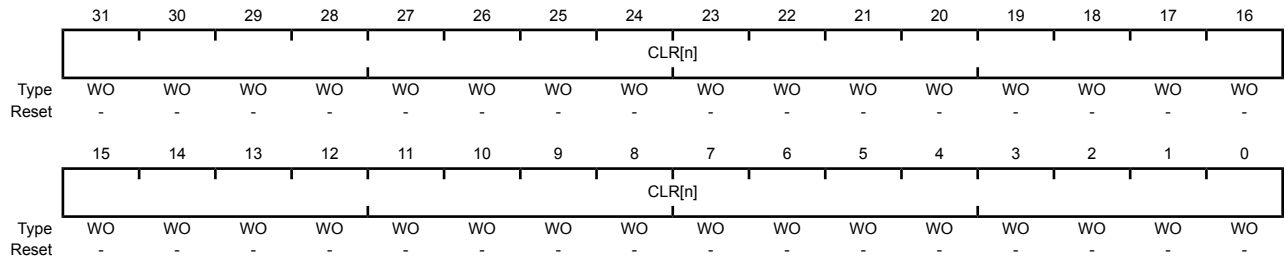
Each bit of the **DMAPRIOCLR** register represents the corresponding DMA channel. Writing a 1 configures the DMA channel to have the default priority level.

### DMA Channel Priority Clear (DMAPRIOCLR)

Base 0x400F.F000

Offset 0x03C

Type WO, reset -



Bit/Field	Name	Type	Reset	Description						
31:0	CLR[n]	WO	-	<p>Channel [n] Priority Clear</p> <p>Set the appropriate bit to clear the high priority level for the specified DMA channel.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No Effect</td> </tr> <tr> <td>1</td> <td>Default Priority</td> </tr> </tbody> </table> <p>Sets DMA channel [n] to a Default priority level.</p>	Value	Description	0	No Effect	1	Default Priority
Value	Description									
0	No Effect									
1	Default Priority									

## Register 20: DMA Bus Error Clear (DMAERRCLR), offset 0x04C

The **DMAERRCLR** register is used to read and clear the DMA bus error status. The error status will be set if the  $\mu$ DMA controller encountered a bus error while performing a DMA transfer. If a bus error occurs on a channel, that channel will be automatically disabled by the  $\mu$ DMA controller. The other channels are unaffected.

### DMAERRCLR Reads

#### DMA Bus Error Clear (DMAERRCLR)

Base 0x400F.F000  
Offset 0x04C  
Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	ERRCLR	R	0	DMA Bus Error Status
Value Description				
0	Low	No bus error is pending.		
1	High	Bus error is pending.		

### DMAERRCLR Writes

#### DMA Bus Error Clear (DMAERRCLR)

Base 0x400F.F000  
Offset 0x04C  
Type WO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description						
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
0	ERRCLR	W	0	<p>DMA Bus Error Status</p> <p>Clears the bus error.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No Effect Bus error status is unchanged.</td> </tr> <tr> <td>1</td> <td>Clear Clears a pending bus error.</td> </tr> </tbody> </table>	Value	Description	0	No Effect Bus error status is unchanged.	1	Clear Clears a pending bus error.
Value	Description									
0	No Effect Bus error status is unchanged.									
1	Clear Clears a pending bus error.									

## Register 21: DMA Peripheral Identification 0 (DMAPeriphID0), offset 0xFE0

The **DMAPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

### DMA Peripheral Identification 0 (DMAPeriphID0)

Base 0x400F.F000

Offset 0xFE0

Type RO, reset 0x0000.0030

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x30	DMA Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

## Register 22: DMA Peripheral Identification 1 (DMAPeriphID1), offset 0xFE4

The **DMAPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

### DMA Peripheral Identification 1 (DMAPeriphID1)

Base 0x400F.F000

Offset 0xFE4

Type RO, reset 0x0000.00B2

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	0

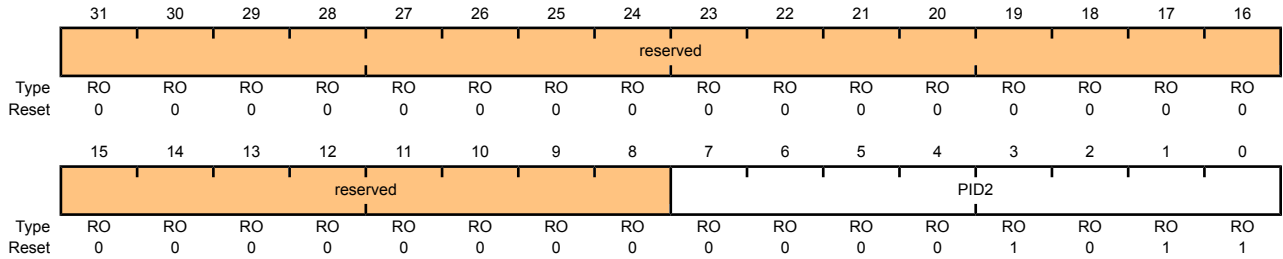
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0xB2	DMA Peripheral ID Register[15:8] Can be used by software to identify the presence of this peripheral.

## Register 23: DMA Peripheral Identification 2 (DMAPeriphID2), offset 0xFE8

The **DMAPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

### DMA Peripheral Identification 2 (DMAPeriphID2)

Base 0x400F.F000  
 Offset 0xFE8  
 Type RO, reset 0x0000.000B



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x0B	DMA Peripheral ID Register[23:16] Can be used by software to identify the presence of this peripheral.

## Register 24: DMA Peripheral Identification 3 (DMAPeriphID3), offset 0xFEC

The **DMAPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

### DMA Peripheral Identification 3 (DMAPeriphID3)

Base 0x400F.F000

Offset 0xFEC

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID3							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x00	DMA Peripheral ID Register[31:24] Can be used by software to identify the presence of this peripheral.

## Register 25: DMA Peripheral Identification 4 (DMAPeriphID4), offset 0xFD0

The **DMAPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

### DMA Peripheral Identification 4 (DMAPeriphID4)

Base 0x400F.F000

Offset 0xFD0

Type RO, reset 0x0000.0004

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x04	DMA Peripheral ID Register Can be used by software to identify the presence of this peripheral.



## Register 26: DMA PrimeCell Identification 0 (DMAPCellID0), offset 0xFF0

The **DMAPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

### DMA PrimeCell Identification 0 (DMAPCellID0)

Base 0x400F.F000

Offset 0xFF0

Type RO, reset 0x0000.000D

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	DMA PrimeCell ID Register[7:0] Provides software a standard cross-peripheral identification system.

### Register 27: DMA PrimeCell Identification 1 (DMAPCellID1), offset 0xFF4

The **DMA PrimeCell IDn** registers are hard-coded and the fields within the registers determine the reset values.

#### DMA PrimeCell Identification 1 (DMAPCellID1)

Base 0x400F.F000

Offset 0xFF4

Type RO, reset 0x0000.00F0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	DMA PrimeCell ID Register[15:8] Provides software a standard cross-peripheral identification system.

## Register 28: DMA PrimeCell Identification 2 (DMAPCellID2), offset 0xFF8

The **DMA PrimeCell IDn** registers are hard-coded and the fields within the registers determine the reset values.

### DMA PrimeCell Identification 2 (DMAPCellID2)

Base 0x400F.F000

Offset 0xFF8

Type RO, reset 0x0000.0005

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	DMA PrimeCell ID Register[23:16] Provides software a standard cross-peripheral identification system.

### Register 29: DMA PrimeCell Identification 3 (DMAPCellID3), offset 0xFFC

The **DMA PrimeCell IDn** registers are hard-coded and the fields within the registers determine the reset values.

#### DMA PrimeCell Identification 3 (DMAPCellID3)

Base 0x400F.F000

Offset 0xFFC

Type RO, reset 0x0000.00B1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID3							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	DMA PrimeCell ID Register[31:24] Provides software a standard cross-peripheral identification system.

## 10 General-Purpose Input/Outputs (GPIOs)

The GPIO module is composed of five physical GPIO blocks, each corresponding to an individual GPIO port (Port A, Port B, Port C, Port D, and Port E, ). The GPIO module supports 0-33 programmable input/output pins, depending on the peripherals being used.

The GPIO module has the following features:

- Two means of port access: either high speed (for single-cycle writes), or legacy for backwards-compatibility with existing code
- Programmable control for GPIO interrupts
  - Interrupt generation masking
  - Edge-triggered on rising, falling, or both
  - Level-sensitive on High or Low values
- 5-V-tolerant input/outputs
- Bit masking in both read and write operations through address lines
- Pins configured as digital inputs are Schmitt-triggered.
- Programmable control for GPIO pad configuration:
  - Weak pull-up or pull-down resistors
  - 2-mA, 4-mA, and 8-mA pad drive for digital communication; up to four pads can be configured with an 18-mA pad drive for high-current applications
  - Slew rate control for the 8-mA drive
  - Open drain enables
  - Digital input enables

### 10.1 Functional Description

---

**Important:** All GPIO pins are tri-stated by default (**GPIOAFSEL=0**, **GPIODEN=0**, **GPIOPDR=0**, and **GPIOPUR=0**), with the exception of the four JTAG/SWD pins ( $PC[3:0]$ ). The JTAG/SWD pins default to their JTAG/SWD functionality (**GPIOAFSEL=1**, **GPIODEN=1** and **GPIOPUR=1**). A Power-On-Reset ( $\overline{POR}$ ) or asserting  $\overline{RST}$  puts both groups of pins back to their default state.

---

Each GPIO port is a separate hardware instantiation of the same physical block(see Figure 10-1 on page 238 and Figure 10-2 on page 239). The LM3S1607 microcontroller contains five ports and thus five of these physical GPIO blocks.

Figure 10-1. Digital I/O Pads

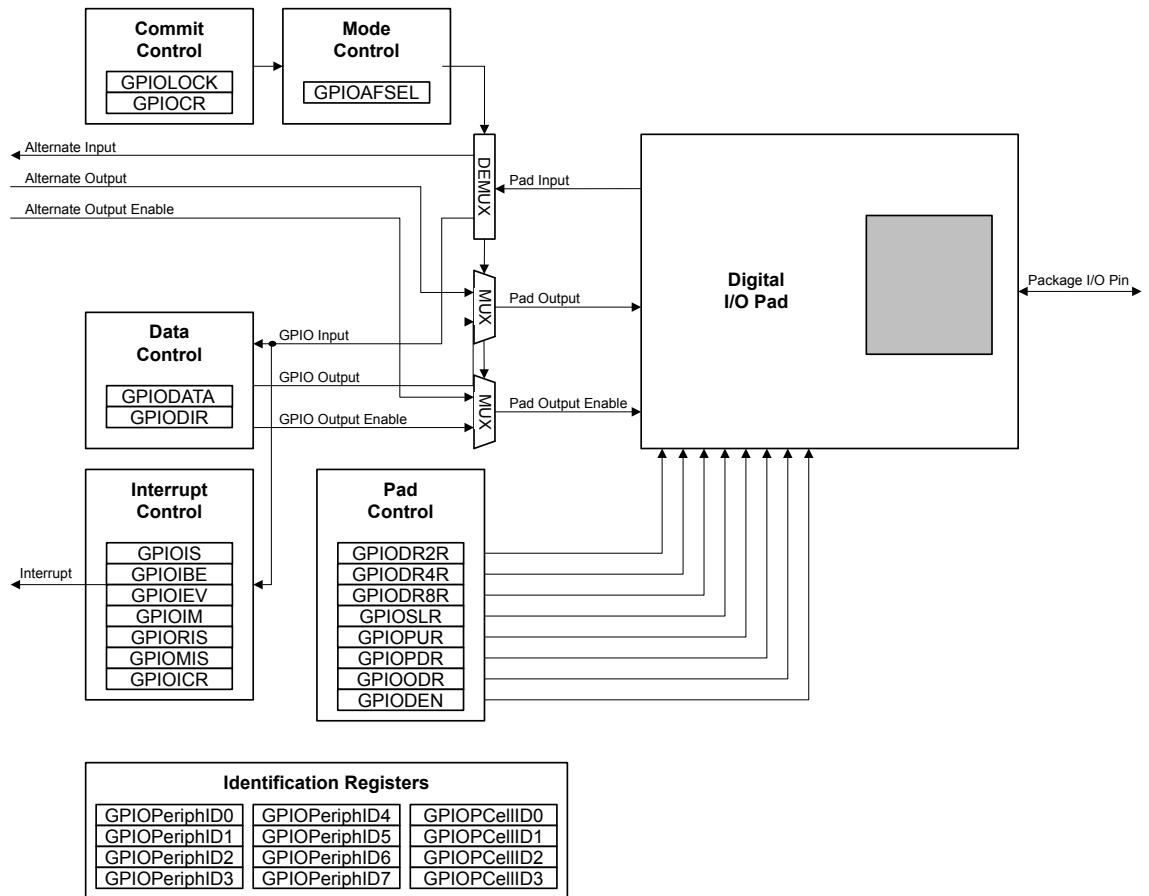
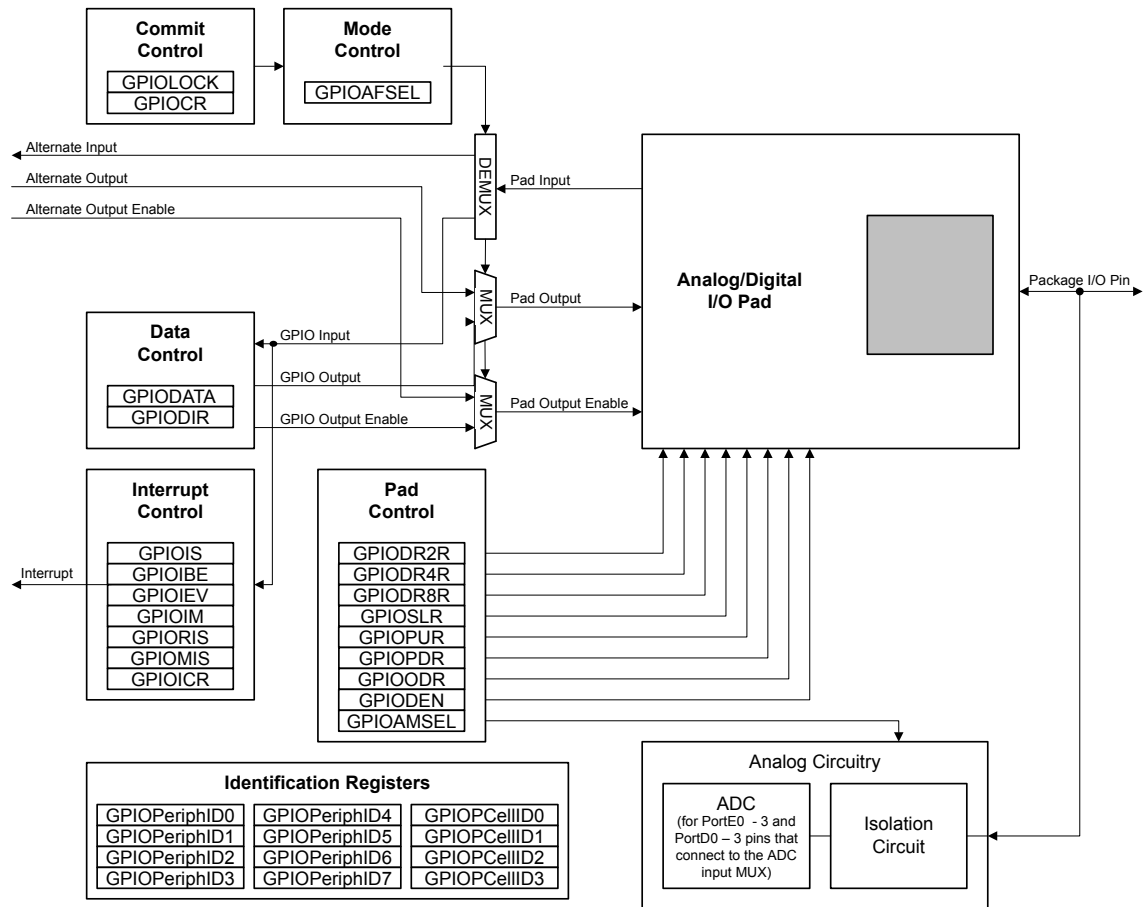


Figure 10-2. Analog/Digital I/O Pads



### 10.1.1 Data Control

The data control registers allow software to configure the operational modes of the GPIOs. The data direction register configures the GPIO as an input or an output while the data register either captures incoming data or drives it out to the pads.

#### 10.1.1.1 Data Direction Operation

The **GPIO Direction (GPIODIR)** register (see page 247) is used to configure each individual pin as an input or output. When the data direction bit is set to 0, the GPIO is configured as an input and the corresponding data register bit will capture and store the value on the GPIO port. When the data direction bit is set to 1, the GPIO is configured as an output and the corresponding data register bit will be driven out on the GPIO port.

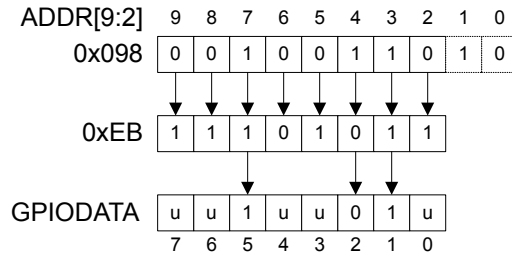
#### 10.1.1.2 Data Register Operation

To aid in the efficiency of software, the GPIO ports allow for the modification of individual bits in the **GPIO Data (GPIODATA)** register (see page 246) by using bits [9:2] of the address bus as a mask. This allows software drivers to modify individual GPIO pins in a single instruction, without affecting the state of the other pins. This is in contrast to the "typical" method of doing a read-modify-write operation to set or clear an individual GPIO pin. To accommodate this feature, the **GPIODATA** register covers 256 locations in the memory map.

During a write, if the address bit associated with that data bit is set to 1, the value of the **GPIODATA** register is altered. If it is cleared to 0, it is left unchanged.

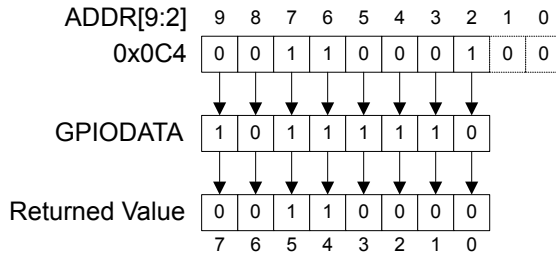
For example, writing a value of 0xEB to the address **GPIODATA + 0x098** would yield as shown in Figure 10-3 on page 240, where *u* is data unchanged by the write.

**Figure 10-3. GPIODATA Write Example**



During a read, if the address bit associated with the data bit is set to 1, the value is read. If the address bit associated with the data bit is set to 0, it is read as a zero, regardless of its actual value. For example, reading address **GPIODATA + 0x0C4** yields as shown in Figure 10-4 on page 240.

**Figure 10-4. GPIODATA Read Example**



### 10.1.2 Interrupt Control

The interrupt capabilities of each GPIO port are controlled by a set of seven registers. With these registers, it is possible to select the source of the interrupt, its polarity, and the edge properties. When one or more GPIO inputs cause an interrupt, a single interrupt output is sent to the interrupt controller for the entire GPIO port. For edge-triggered interrupts, software must clear the interrupt to enable any further interrupts. For a level-sensitive interrupt, it is assumed that the external source holds the level constant for the interrupt to be recognized by the controller.

Three registers are required to define the edge or sense that causes interrupts:

- **GPIO Interrupt Sense (GPIOIS)** register (see page 248)
- **GPIO Interrupt Both Edges (GPIOIBE)** register (see page 249)
- **GPIO Interrupt Event (GPIOIEV)** register (see page 250)

Interrupts are enabled/disabled via the **GPIO Interrupt Mask (GPIOIM)** register (see page 251).

When an interrupt condition occurs, the state of the interrupt signal can be viewed in two locations: the **GPIO Raw Interrupt Status (GPIO RIS)** and **GPIO Masked Interrupt Status (GPIO MIS)** registers (see page 252 and page 253). As the name implies, the **GPIO MIS** register only shows interrupt



conditions that are allowed to be passed to the controller. The **GPIORIS** register indicates that a GPIO pin meets the conditions for an interrupt, but has not necessarily been sent to the controller.

In addition to providing GPIO functionality,  $PB4$  can also be used as an external trigger for the ADC. If  $PB4$  is configured as a non-masked interrupt pin (the appropriate bit of **GPIOIM** is set to 1), not only is an interrupt for PortB generated, but an external trigger signal is sent to the ADC. If the **ADC Event Multiplexer Select (ADCEMUX)** register is configured to use the external trigger, an ADC conversion is initiated.

If no other PortB pins are being used to generate interrupts, the ARM Integrated Nested Vectored Interrupt Controller (NVIC) Interrupt Set Enable (SETNA) register can disable the PortB interrupts and the ADC interrupt can be used to read back the converted data. Otherwise, the PortB interrupt handler needs to ignore and clear interrupts on  $B4$ , and wait for the ADC interrupt or the ADC interrupt needs to be disabled in the SETNA register and the PortB interrupt handler polls the ADC registers until the conversion is completed.

Interrupts are cleared by writing a 1 to the appropriate bit of the **GPIO Interrupt Clear (GPIOICR)** register (see page 254).

When programming the following interrupt control registers, the interrupts should be masked (**GPIOIM** set to 0). Writing any value to an interrupt control register (**GPIOIS**, **GPIOIBE**, or **GPIOIEV**) can generate a spurious interrupt if the corresponding bits are enabled.

### 10.1.3 Mode Control

The GPIO pins can be controlled by either hardware or software. When hardware control is enabled via the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 255), the pin state is controlled by its alternate function (that is, the peripheral). Software control corresponds to GPIO mode, where the **GPIODATA** register is used to read/write the corresponding pins.

**Note:** If any pin is to be used as an ADC input, the appropriate bit in **GPIOAMSEL** must be written to 1 to disable the analog isolation circuit.

### 10.1.4 Commit Control

The commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 255), **GPIO Pull-Up Select (GPIOPUR)** register (see page 261), and **GPIO Digital Enable (GPIODEN)** register (see page 264) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 266) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 267) have been set to 1.

### 10.1.5 Pad Control

The pad control registers allow for GPIO pad configuration by software based on the application requirements. The pad control registers include the **GPIODR2R**, **GPIODR4R**, **GPIODR8R**, **GPIODR**, **GPIOPUR**, **GPIOPDR**, **GPIOSLR**, and **GPIODEN** registers. These registers control drive strength, open-drain configuration, pull-up and pull-down resistors, slew-rate control and digital input enable.

For special high-current applications, the GPIO output buffers may be used with the following restrictions. With the GPIO pins configured as 8-mA output drivers, a total of four GPIO outputs may be used to sink current loads up to 18 mA each. At 18-mA sink current loading, the  $V_{OL}$  value is specified as 1.2 V. The high-current GPIO package pins must be selected such that there are only a maximum of two per side of the physical package with the total number of high-current GPIO outputs not exceeding four for the entire package.

### 10.1.6 Identification

The identification registers configured at reset allow software to detect and identify the module as a GPIO block. The identification registers include the **GPIOPeriphID0-GPIOPeriphID7** registers as well as the **GPIOCellID0-GPIOCellID3** registers.

## 10.2 Initialization and Configuration

The GPIO modules may be accessed via two different memory apertures. The legacy aperture is backwards-compatible with previous Stellaris parts and offers two-cycle access time to all GPIO registers. The high-speed aperture offers the same register map but provides single-cycle access times. These apertures are mutually exclusive. The aperture enabled for a given GPIO port is controlled by the appropriate bit in the **GPIOHCTL** register (see page 84).

To use the GPIO, the peripheral clock must be enabled by setting the appropriate GPIO Port bit field (**GPIO<sub>n</sub>**) in the **RCGC2** register.

On reset, all GPIO pins (except for the four JTAG pins) are configured out of reset to be undriven (tristate): **GPIOAFSEL=0**, **GPIO DEN=0**, **GPIO PDR=0**, and **GPIO PUR=0**. Table 10-1 on page 242 shows all possible configurations of the GPIO pads and the control register settings required to achieve them. Table 10-2 on page 243 shows how a rising edge interrupt would be configured for pin 2 of a GPIO port.

**Table 10-1. GPIO Pad Configuration Examples**

Configuration	GPIO Register Bit Value <sup>a</sup>									
	AFSEL	DIR	ODR	DEN	PUR	PDR	DR2R	DR4R	DR8R	SLR
Digital Input (GPIO)	0	0	0	1	?	?	X	X	X	X
Digital Output (GPIO)	0	1	0	1	?	?	?	?	?	?
Open Drain Input (GPIO)	0	0	1	1	X	X	X	X	X	X
Open Drain Output (GPIO)	0	1	1	1	X	X	?	?	?	?
Open Drain Input/Output (I <sup>2</sup> C)	1	X	1	1	X	X	?	?	?	?
Digital Input (Timer CCP)	1	X	0	1	?	?	X	X	X	X
Digital Output (Timer PWM)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (SSI)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (UART)	1	X	0	1	?	?	?	?	?	?

a. X=Ignored (don't care bit)

?=Can be either 0 or 1, depending on the configuration

Table 10-2. GPIO Interrupt Configuration Example

Register	Desired Interrupt Event Trigger	Pin 2 Bit Value <sup>a</sup>							
		7	6	5	4	3	2	1	0
GPIOIS	0=edge 1=level	X	X	X	X	X	0	X	X
GPIOIBE	0=single edge 1=both edges	X	X	X	X	X	0	X	X
GPIOIEV	0=Low level, or negative edge 1=High level, or positive edge	X	X	X	X	X	1	X	X
GPIOIM	0=masked 1=not masked	0	0	0	0	0	1	0	0

a. X=Ignored (don't care bit)

### 10.3 Register Map

Table 10-3 on page 244 lists the GPIO registers. The offset listed is a hexadecimal increment to the register's address, relative to that GPIO port's base address:

- GPIO Port A (legacy): 0x4000.4000
- GPIO Port A (high-speed): 0x4005.8000
- GPIO Port B (legacy): 0x4000.5000
- GPIO Port B (high-speed): 0x4005.9000
- GPIO Port C (legacy): 0x4000.6000
- GPIO Port C (high-speed): 0x4005.A000
- GPIO Port D (legacy): 0x4000.7000
- GPIO Port D (high-speed): 0x4005.B000
- GPIO Port E (legacy): 0x4002.4000
- GPIO Port E (high-speed): 0x4005.C000

**Important:** The GPIO registers in this chapter are duplicated in each GPIO block, however, depending on the block, all eight bits may not be connected to a GPIO pad. In those cases, writing to those unconnected bits has no effect and reading those unconnected bits returns no meaningful data.

**Note:** The default reset value for the **GPIOAFSEL**, **GIOPUR**, and **GIODEN** registers are 0x0000.0000 for all GPIO pins, with the exception of the four JTAG/SWD pins ( $PC[3:0]$ ). These four pins default to JTAG/SWD functionality. Because of this, the default reset value of these registers for Port C is 0x0000.000F.

The default register type for the **GPIOCR** register is RO for all GPIO pins, with the exception of the **NMI** pin and the four JTAG/SWD pins ( $PB7$  and  $PC[3:0]$ ). These five pins are currently the only GPIOs that are protected by the **GPIOCR** register. Because of this, the register type for GPIO Port B7 and GPIO Port C[3:0] is R/W.

The default reset value for the **GPIOCR** register is 0x0000.00FF for all GPIO pins, with the exception of the **NMI** pin and the four JTAG/SWD pins ( $PB7$  and  $PC[3:0]$ ). To ensure that the JTAG port is not accidentally programmed as a GPIO, these four pins default to non-committable. To ensure that the **NMI** pin is not accidentally programmed as the non-maskable interrupt pin, it defaults to non-committable. Because of this, the default reset value of **GPIOCR** for GPIO Port B is 0x0000.007F while the default reset value of GPIOCR for Port C is 0x0000.00F0.

**Table 10-3. GPIO Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	GPIODATA	R/W	0x0000.0000	GPIO Data	246
0x400	GPIODIR	R/W	0x0000.0000	GPIO Direction	247
0x404	GPIOIS	R/W	0x0000.0000	GPIO Interrupt Sense	248
0x408	GPIOIBE	R/W	0x0000.0000	GPIO Interrupt Both Edges	249
0x40C	GPIOIEV	R/W	0x0000.0000	GPIO Interrupt Event	250
0x410	GPIOIM	R/W	0x0000.0000	GPIO Interrupt Mask	251
0x414	GIORIS	RO	0x0000.0000	GPIO Raw Interrupt Status	252
0x418	GIOMIS	RO	0x0000.0000	GPIO Masked Interrupt Status	253
0x41C	GPIOICR	W1C	0x0000.0000	GPIO Interrupt Clear	254
0x420	GPIOAFSEL	R/W	-	GPIO Alternate Function Select	255
0x500	GIODR2R	R/W	0x0000.00FF	GPIO 2-mA Drive Select	257
0x504	GIODR4R	R/W	0x0000.0000	GPIO 4-mA Drive Select	258
0x508	GIODR8R	R/W	0x0000.0000	GPIO 8-mA Drive Select	259
0x50C	GPIOODR	R/W	0x0000.0000	GPIO Open Drain Select	260
0x510	GIOPUR	R/W	-	GPIO Pull-Up Select	261
0x514	GIOPDR	R/W	0x0000.0000	GPIO Pull-Down Select	262
0x518	GPIOSLR	R/W	0x0000.0000	GPIO Slew Rate Control Select	263
0x51C	GIODEN	R/W	-	GPIO Digital Enable	264
0x520	GPIOLOCK	R/W	0x0000.0001	GPIO Lock	266
0x524	GPIOCR	-	-	GPIO Commit	267
0x528	GPIOAMSEL	R/W	0x0000.0000	GPIO Analog Mode Select	269

Offset	Name	Type	Reset	Description	See page
0xFD0	GPIOPeriphID4	RO	0x0000.0000	GPIO Peripheral Identification 4	270
0xFD4	GPIOPeriphID5	RO	0x0000.0000	GPIO Peripheral Identification 5	271
0xFD8	GPIOPeriphID6	RO	0x0000.0000	GPIO Peripheral Identification 6	272
0xFDC	GPIOPeriphID7	RO	0x0000.0000	GPIO Peripheral Identification 7	273
0xFE0	GPIOPeriphID0	RO	0x0000.0061	GPIO Peripheral Identification 0	274
0xFE4	GPIOPeriphID1	RO	0x0000.0000	GPIO Peripheral Identification 1	275
0xFE8	GPIOPeriphID2	RO	0x0000.0018	GPIO Peripheral Identification 2	276
0xFEC	GPIOPeriphID3	RO	0x0000.0001	GPIO Peripheral Identification 3	277
0xFF0	GPIOPrimeCellID0	RO	0x0000.000D	GPIO PrimeCell Identification 0	278
0xFF4	GPIOPrimeCellID1	RO	0x0000.00F0	GPIO PrimeCell Identification 1	279
0xFF8	GPIOPrimeCellID2	RO	0x0000.0005	GPIO PrimeCell Identification 2	280
0xFFC	GPIOPrimeCellID3	RO	0x0000.00B1	GPIO PrimeCell Identification 3	281

## 10.4 Register Descriptions

The remainder of this section lists and describes the GPIO registers, in numerical order by address offset.

### Register 1: GPIO Data (GPIODATA), offset 0x000

The **GPIODATA** register is the data register. In software control mode, values written in the **GPIODATA** register are transferred onto the GPIO port pins if the respective pins have been configured as outputs through the **GPIO Direction (GPIODIR)** register (see page 247).

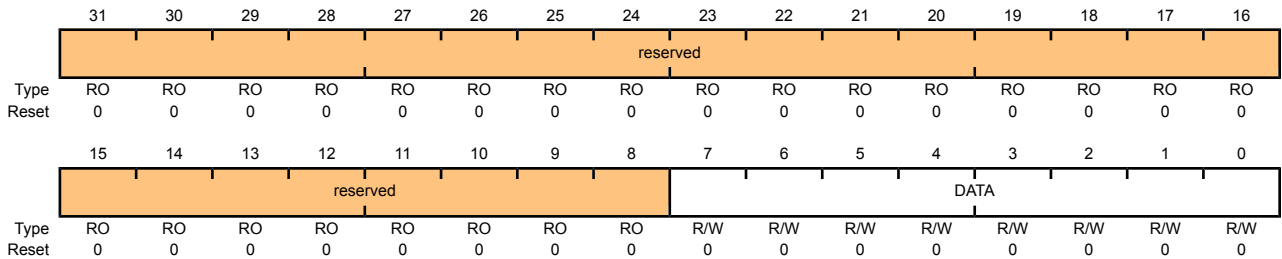
In order to write to **GPIODATA**, the corresponding bits in the mask, resulting from the address bus bits [9:2], must be High. Otherwise, the bit values remain unchanged by the write.

Similarly, the values read from this register are determined for each bit by the mask bit derived from the address used to access the data register, bits [9:2]. Bits that are 1 in the address mask cause the corresponding bits in **GPIODATA** to be read, and bits that are 0 in the address mask cause the corresponding bits in **GPIODATA** to be read as 0, regardless of their value.

A read from **GPIODATA** returns the last bit value written if the respective pins are configured as outputs, or it returns the value on the corresponding input pin when these are configured as inputs. All bits are cleared by a reset.

#### GPIO Data (GPIODATA)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x000  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x00	GPIO Data

This register is virtually mapped to 256 locations in the address space. To facilitate the reading and writing of data to these registers by independent drivers, the data read from and the data written to the registers are masked by the eight address lines `ipaddr[9:2]`. Reads from this register return its current state. Writes to this register only affect bits that are not masked by `ipaddr[9:2]` and are configured as outputs. See "Data Register Operation" on page 239 for examples of reads and writes.

## Register 2: GPIO Direction (GPIODIR), offset 0x400

The **GPIODIR** register is the data direction register. Bits set to 1 in the **GPIODIR** register configure the corresponding pin to be an output, while bits set to 0 configure the pins to be inputs. All bits are cleared by a reset, meaning all GPIO pins are inputs by default.

### GPIO Direction (GPIODIR)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x400  
 Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DIR	R/W	0x00	GPIO Data Direction

The **DIR** values are defined as follows:

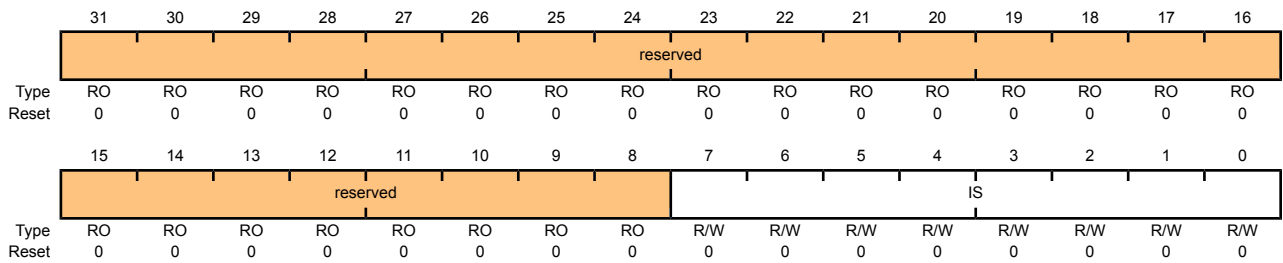
Value	Description
0	Pins are inputs.
1	Pins are outputs.

### Register 3: GPIO Interrupt Sense (GPIOIS), offset 0x404

The **GPIOIS** register is the interrupt sense register. Bits set to 1 in **GPIOIS** configure the corresponding pins to detect levels, while bits set to 0 configure the pins to detect edges. All bits are cleared by a reset.

#### GPIO Interrupt Sense (GPIOIS)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x404  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IS	R/W	0x00	GPIO Interrupt Sense

The **IS** values are defined as follows:

Value	Description
0	Edge on corresponding pin is detected (edge-sensitive).
1	Level on corresponding pin is detected (level-sensitive).



## Register 4: GPIO Interrupt Both Edges (GPIOIBE), offset 0x408

The **GPIOIBE** register is the interrupt both-edges register. When the corresponding bit in the **GPIO Interrupt Sense (GPIOIS)** register (see page 248) is set to detect edges, bits set to High in **GPIOIBE** configure the corresponding pin to detect both rising and falling edges, regardless of the corresponding bit in the **GPIO Interrupt Event (GPIOIEV)** register (see page 250). Clearing a bit configures the pin to be controlled by **GPIOIEV**. All bits are cleared by a reset.

### GPIO Interrupt Both Edges (GPIOIBE)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x408  
 Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IBE	R/W	0x00	GPIO Interrupt Both Edges

The **IBE** values are defined as follows:

#### Value Description

- 0 Interrupt generation is controlled by the **GPIO Interrupt Event (GPIOIEV)** register (see page 250).
- 1 Both edges on the corresponding pin trigger an interrupt.

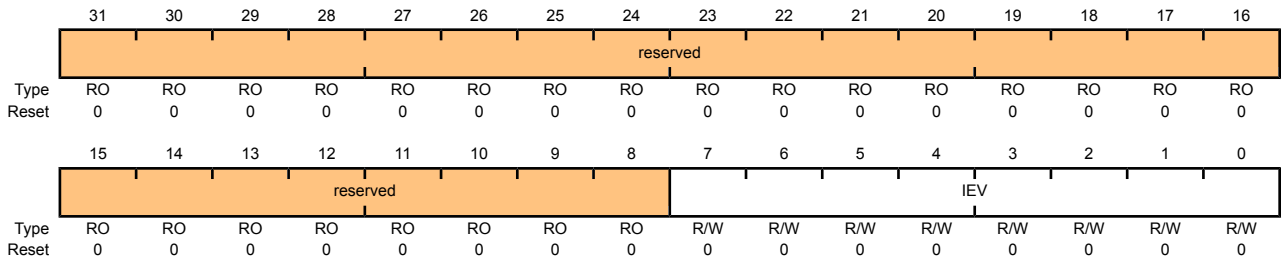
**Note:** Single edge is determined by the corresponding bit in **GPIOIEV**.

### Register 5: GPIO Interrupt Event (GPIOIEV), offset 0x40C

The **GPIOIEV** register is the interrupt event register. Bits set to High in **GPIOIEV** configure the corresponding pin to detect rising edges or high levels, depending on the corresponding bit value in the **GPIO Interrupt Sense (GPIOIS)** register (see page 248). Clearing a bit configures the pin to detect falling edges or low levels, depending on the corresponding bit value in **GPIOIS**. All bits are cleared by a reset.

#### GPIO Interrupt Event (GPIOIEV)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x40C  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IEV	R/W	0x00	GPIO Interrupt Event

The **IEV** values are defined as follows:

Value	Description
0	Falling edge or Low levels on corresponding pins trigger interrupts.
1	Rising edge or High levels on corresponding pins trigger interrupts.

## Register 6: GPIO Interrupt Mask (GPIOIM), offset 0x410

The **GPIOIM** register is the interrupt mask register. Bits set to High in **GPIOIM** allow the corresponding pins to trigger their individual interrupts and the combined **GPIOINTR** line. Clearing a bit disables interrupt triggering on that pin. All bits are cleared by a reset.

### GPIO Interrupt Mask (GPIOIM)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x410  
 Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IME	R/W	0x00	GPIO Interrupt Mask Enable

The **IME** values are defined as follows:

Value	Description
0	Corresponding pin interrupt is masked.
1	Corresponding pin interrupt is not masked.

### Register 7: GPIO Raw Interrupt Status (GPIORIS), offset 0x414

The **GPIORIS** register is the raw interrupt status register. Bits read High in **GPIORIS** reflect the status of interrupt trigger conditions detected (raw, prior to masking), indicating that all the requirements have been met, before they are finally allowed to trigger by the **GPIO Interrupt Mask (GPIOIM)** register (see page 251). Bits read as zero indicate that corresponding input pins have not initiated an interrupt. All bits are cleared by a reset.

#### GPIO Raw Interrupt Status (GPIORIS)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x414  
 Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	RIS	RO	0x00	GPIO Interrupt Raw Status Reflects the status of interrupt trigger condition detection on pins (raw, prior to masking).

The RIS values are defined as follows:

Value	Description
0	Corresponding pin interrupt requirements not met.
1	Corresponding pin interrupt has met requirements.

## Register 8: GPIO Masked Interrupt Status (GPIOMIS), offset 0x418

The **GPIOMIS** register is the masked interrupt status register. Bits read High in **GPIOMIS** reflect the status of input lines triggering an interrupt. Bits read as Low indicate that either no interrupt has been generated, or the interrupt is masked.

In addition to providing GPIO functionality, **PB4** can also be used as an external trigger for the ADC. If **PB4** is configured as a non-masked interrupt pin (the appropriate bit of **GPIOIM** is set to 1), not only is an interrupt for PortB generated, but an external trigger signal is sent to the ADC. If the **ADC Event Multiplexer Select (ADCEMUX)** register is configured to use the external trigger, an ADC conversion is initiated.

If no other PortB pins are being used to generate interrupts, the ARM Integrated Nested Vectored Interrupt Controller (NVIC) Interrupt Set Enable (SETNA) register can disable the PortB interrupts and the ADC interrupt can be used to read back the converted data. Otherwise, the PortB interrupt handler needs to ignore and clear interrupts on B4, and wait for the ADC interrupt or the ADC interrupt needs to be disabled in the SETNA register and the PortB interrupt handler polls the ADC registers until the conversion is completed.

**GPIOMIS** is the state of the interrupt after masking.

### GPIO Masked Interrupt Status (GPIOMIS)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x418  
 Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

7:0	MIS	RO	0x00	GPIO Masked Interrupt Status
-----	-----	----	------	------------------------------

Masked value of interrupt due to corresponding pin.

The MIS values are defined as follows:

Value Description

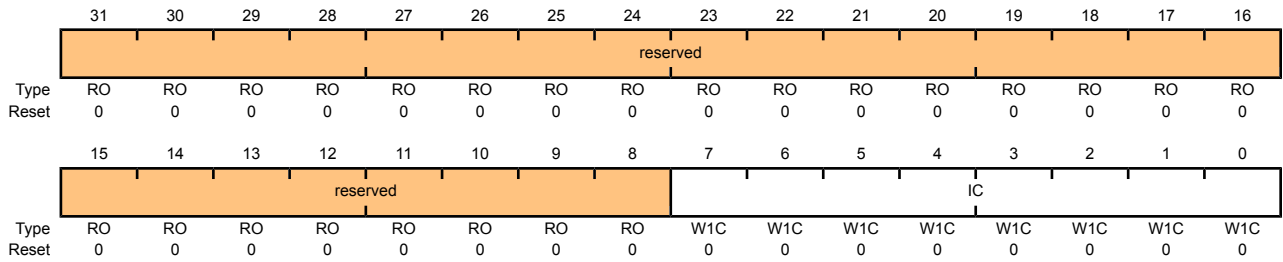
- 0 Corresponding GPIO line interrupt not active.
- 1 Corresponding GPIO line asserting interrupt.

### Register 9: GPIO Interrupt Clear (GPIOICR), offset 0x41C

The **GPIOICR** register is the interrupt clear register. Writing a 1 to a bit in this register clears the corresponding interrupt edge detection logic register. Writing a 0 has no effect.

#### GPIO Interrupt Clear (GPIOICR)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x41C  
 Type W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IC	W1C	0x00	GPIO Interrupt Clear

The IC values are defined as follows:

Value	Description
0	Corresponding interrupt is unaffected.
1	Corresponding interrupt is cleared.

## Register 10: GPIO Alternate Function Select (GPIOAFSEL), offset 0x420

The **GPIOAFSEL** register is the mode control select register. Writing a 1 to any bit in this register selects the hardware control for the corresponding GPIO line. All bits are cleared by a reset, therefore no GPIO line is set to hardware control by default.

The commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 255), **GPIO Pull-Up Select (GPIOPUR)** register (see page 261), and **GPIO Digital Enable (GPIODEN)** register (see page 264) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 266) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 267) have been set to 1.

**Important:** All GPIO pins are tri-stated by default (**GPIOAFSEL=0**, **GPIODEN=0**, **GPIOPDR=0**, and **GPIOPUR=0**), with the exception of the four JTAG/SWD pins ( $PC[3:0]$ ). The JTAG/SWD pins default to their JTAG/SWD functionality (**GPIOAFSEL=1**, **GPIODEN=1** and **GPIOPUR=1**). A Power-On-Reset ( $\overline{POR}$ ) or asserting  $\overline{RST}$  puts both groups of pins back to their default state.

**Caution – It is possible to create a software sequence that prevents the debugger from connecting to the Stellaris® microcontroller. If the program code loaded into flash immediately changes the JTAG pins to their GPIO functionality, the debugger may not have enough time to connect and halt the controller before the JTAG pin functionality switches. This may lock the debugger out of the part. This can be avoided with a software routine that restores JTAG functionality based on an external or software trigger.**

### GPIO Alternate Function Select (GPIOAFSEL)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000

Offset 0x420  
 Type R/W, reset -

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
7:0	AFSEL	R/W	-	GPIO Alternate Function Select The AFSEL values are defined as follows:  Value Description 0 Software control of corresponding GPIO line (GPIO mode). 1 Hardware control of corresponding GPIO line (alternate hardware function).  <b>Note:</b> The default reset value for the <b>GPIOAFSEL</b> , <b>GPIOPUR</b> , and <b>GPIODEN</b> registers are 0x0000.0000 for all GPIO pins, with the exception of the four JTAG/SWD pins (PC[3:0]). These four pins default to JTAG/SWD functionality. Because of this, the default reset value of these registers for Port C is 0x0000.000F.

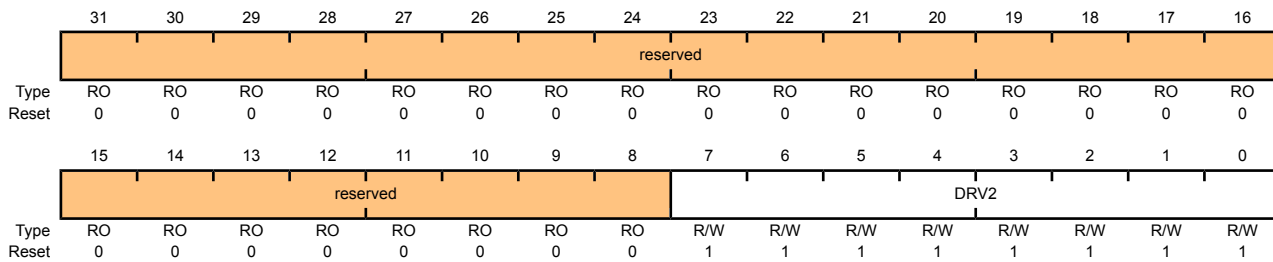


### Register 11: GPIO 2-mA Drive Select (GPIODR2R), offset 0x500

The **GPIODR2R** register is the 2-mA drive control register. It allows for each GPIO signal in the port to be individually configured without affecting the other pads. When writing a **DRV2** bit for a GPIO signal, the corresponding **DRV4** bit in the **GPIODR4R** register and the **DRV8** bit in the **GPIODR8R** register are automatically cleared by hardware.

#### GPIO 2-mA Drive Select (GPIODR2R)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x500  
 Type R/W, reset 0x0000.00FF



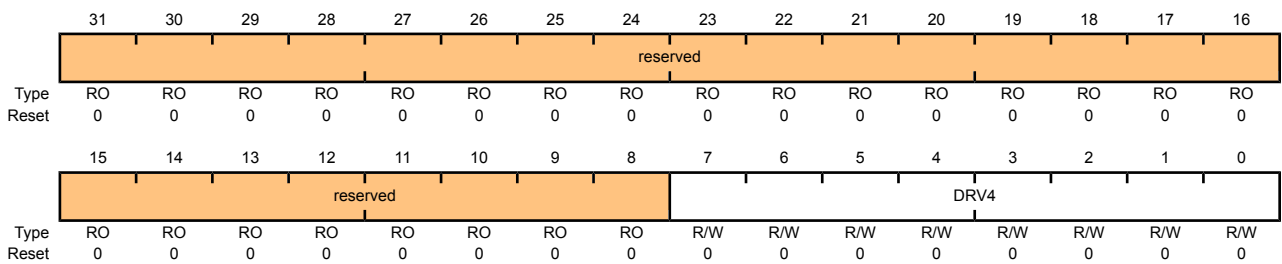
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV2	R/W	0xFF	Output Pad 2-mA Drive Enable  A write of 1 to either <b>GPIODR4[n]</b> or <b>GPIODR8[n]</b> clears the corresponding 2-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the legacy memory aperture. If using high-speed access, the change is effective on the next clock cycle.

### Register 12: GPIO 4-mA Drive Select (GPIODR4R), offset 0x504

The **GPIODR4R** register is the 4-mA drive control register. It allows for each GPIO signal in the port to be individually configured without affecting the other pads. When writing the **DRV4** bit for a GPIO signal, the corresponding **DRV2** bit in the **GPIODR2R** register and the **DRV8** bit in the **GPIODR8R** register are automatically cleared by hardware.

#### GPIO 4-mA Drive Select (GPIODR4R)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x504  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV4	R/W	0x00	Output Pad 4-mA Drive Enable  A write of 1 to either <b>GPIODR2[n]</b> or <b>GPIODR8[n]</b> clears the corresponding 4-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the legacy memory aperture. If using high-speed access, the change is effective on the next clock cycle.

### Register 13: GPIO 8-mA Drive Select (GPIODR8R), offset 0x508

The **GPIODR8R** register is the 8-mA drive control register. It allows for each GPIO signal in the port to be individually configured without affecting the other pads. When writing the **DRV8** bit for a GPIO signal, the corresponding **DRV2** bit in the **GPIODR2R** register and the **DRV4** bit in the **GPIODR4R** register are automatically cleared by hardware. The 8-mA setting is also used for high-current operation.

**Note:** There is no configuration difference between 8-mA and high-current operation. The additional current capacity results from a shift in the  $V_{OH}/V_{OL}$  levels. See “Recommended DC Operating Conditions” on page 500 for further information.

#### GPIO 8-mA Drive Select (GPIODR8R)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x508  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DRV8							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV8	R/W	0x00	Output Pad 8-mA Drive Enable  A write of 1 to either <b>GPIODR2[n]</b> or <b>GPIODR4[n]</b> clears the corresponding 8-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the legacy memory aperture. If using high-speed access, the change is effective on the next clock cycle.

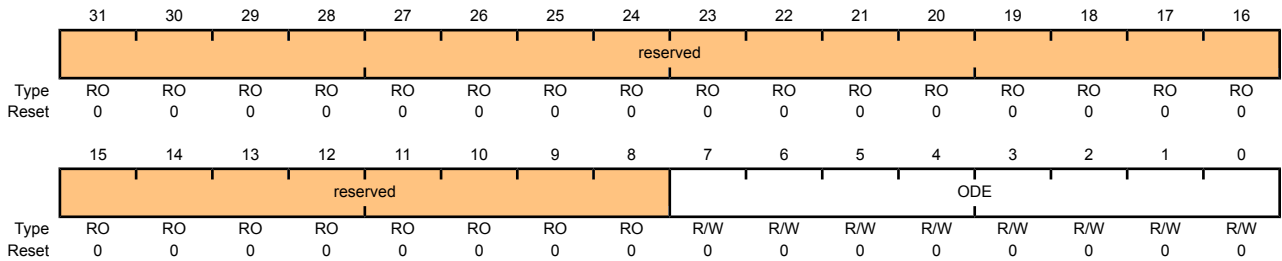
### Register 14: GPIO Open Drain Select (GPIOODR), offset 0x50C

The **GPIOODR** register is the open drain control register. Setting a bit in this register enables the open drain configuration of the corresponding GPIO pad. When open drain mode is enabled, the corresponding bit should also be set in the **GPIO Digital Input Enable (GPIODEN)** register (see page 264). Corresponding bits in the drive strength registers (**GPIODR2R**, **GPIODR4R**, **GPIODR8R**, and **GPIOSLR**) can be set to achieve the desired rise and fall times. The GPIO acts as an open drain input if the corresponding bit in the **GPIODIR** register is set to 0; and as an open drain output when set to 1.

When using the I<sup>2</sup>C module, in addition to configuring the pin to open drain, the **GPIO Alternate Function Select (GPIOAFSEL)** register bit for the I<sup>2</sup>C clock and data pins should be set to 1 (see examples in “Initialization and Configuration” on page 242).

#### GPIO Open Drain Select (GPIOODR)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x50C  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ODE	R/W	0x00	Output Pad Open Drain Enable

The ODE values are defined as follows:

Value	Description
0	Open drain configuration is disabled.
1	Open drain configuration is enabled.

## Register 15: GPIO Pull-Up Select (GPIOPUR), offset 0x510

The **GPIOPUR** register is the pull-up control register. When a bit is set to 1, it enables a weak pull-up resistor on the corresponding GPIO signal. Setting a bit in **GPIOPUR** automatically clears the corresponding bit in the **GPIO Pull-Down Select (GPIOPDR)** register (see page 262). Write access to this register is protected with the **GPIOCR** register. Bits in **GPIOCR** that are set to 0 will prevent writes to the equivalent bit in this register.

**Note:** The commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 255), **GPIO Pull-Up Select (GPIOPUR)** register (see page 261), and **GPIO Digital Enable (GPIODEN)** register (see page 264) are not committed to storage unless the **GPIO Lock (GPIOLCK)** register (see page 266) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 267) have been set to 1.

### GPIO Pull-Up Select (GPIOPUR)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x510  
 Type R/W, reset -

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PUE	R/W	-	Pad Weak Pull-Up Enable

A write of 1 to **GPIOPDR[n]** clears the corresponding **GPIOPUR[n]** enables. The change is effective on the second clock cycle after the write.

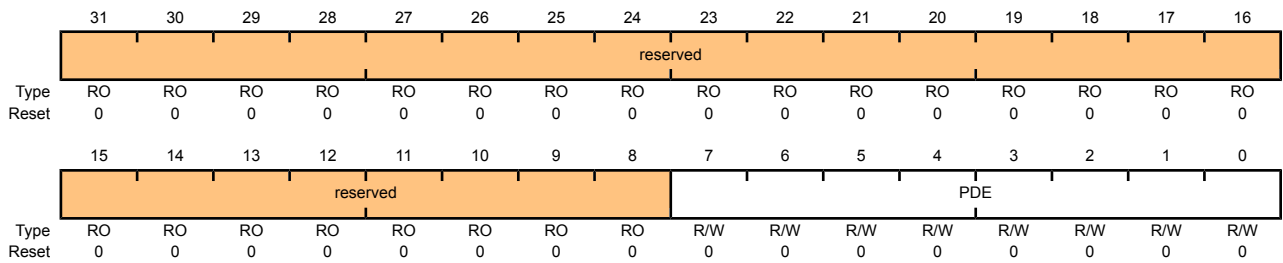
**Note:** The default reset value for the **GPIOAFSEL**, **GPIOPUR**, and **GPIODEN** registers are 0x0000.0000 for all GPIO pins, with the exception of the four JTAG/SWD pins (**PC[3:0]**). These four pins default to JTAG/SWD functionality. Because of this, the default reset value of these registers for Port C is 0x0000.000F.

### Register 16: GPIO Pull-Down Select (GPIOPDR), offset 0x514

The **GPIOPDR** register is the pull-down control register. When a bit is set to 1, it enables a weak pull-down resistor on the corresponding GPIO signal. Setting a bit in **GPIOPDR** automatically clears the corresponding bit in the **GPIO Pull-Up Select (GPIOPUR)** register (see page 261).

#### GPIO Pull-Down Select (GPIOPDR)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x514  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PDE	R/W	0x00	Pad Weak Pull-Down Enable  A write of 1 to <b>GPIOPUR[n]</b> clears the corresponding <b>GPIOPDR[n]</b> enables. The change is effective on the second clock cycle after the write.

## Register 17: GPIO Slew Rate Control Select (GPIOSLR), offset 0x518

The **GPIOSLR** register is the slew rate control register. Slew rate control is only available when using the 8-mA drive strength option via the **GPIO 8-mA Drive Select (GPIO8R)** register (see page 259).

### GPIO Slew Rate Control Select (GPIOSLR)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x518  
 Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	SRL	R/W	0x00	Slew Rate Limit Enable (8-mA drive only)

The **SRL** values are defined as follows:

Value	Description
0	Slew rate control disabled.
1	Slew rate control enabled.

### Register 18: GPIO Digital Enable (GPIODEN), offset 0x51C

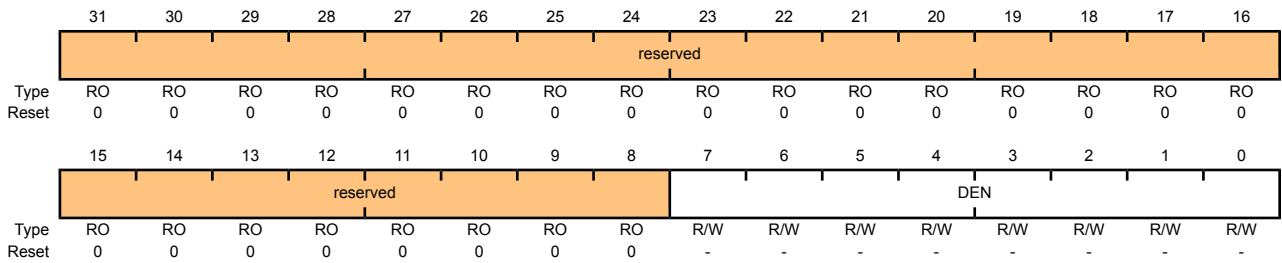
**Note:** Pins configured as digital inputs are Schmitt-triggered.

The **GPIODEN** register is the digital enable register. By default, with the exception of the GPIO signals used for JTAG/SWD function, all other GPIO signals are configured out of reset to be undriven (tristate). Their digital function is disabled; they do not drive a logic value on the pin and they do not allow the pin voltage into the GPIO receiver. To use the pin in a digital function (either GPIO or alternate function), the corresponding **GPIODEN** bit must be set.

**Note:** The commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 255), **GPIO Pull-Up Select (GPIOPUR)** register (see page 261), and **GPIO Digital Enable (GPIODEN)** register (see page 264) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 266) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 267) have been set to 1.

#### GPIO Digital Enable (GPIODEN)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x51C  
 Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.



---

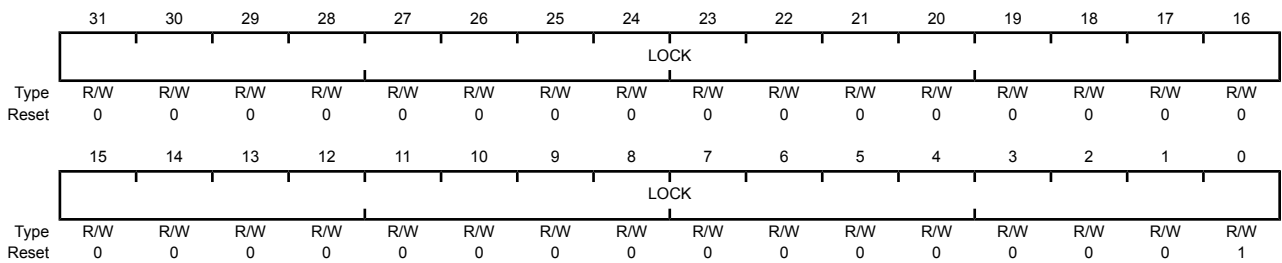
Bit/Field	Name	Type	Reset	Description
7:0	DEN	R/W	-	Digital Enable
				The DEN values are defined as follows:
				Value Description
				0 Digital functions disabled.
				1 Digital functions enabled.
				<b>Note:</b> The default reset value for the <b>GPIOAFSEL</b> , <b>GPIOPUR</b> , and <b>GPIODEN</b> registers are 0x0000.0000 for all GPIO pins, with the exception of the four JTAG/SWD pins ( $PC[3:0]$ ). These four pins default to JTAG/SWD functionality. Because of this, the default reset value of these registers for Port C is 0x0000.000F.

### Register 19: GPIO Lock (GPIOLOCK), offset 0x520

The **GPIOLOCK** register enables write access to the **GPIOCR** register (see page 267). Writing 0x0x4C4F.434B to the **GPIOLOCK** register will unlock the **GPIOCR** register. Writing any other value to the **GPIOLOCK** register re-enables the locked state. Reading the **GPIOLOCK** register returns the lock status rather than the 32-bit value that was previously written. Therefore, when write accesses are disabled, or locked, reading the **GPIOLOCK** register returns 0x00000001. When write accesses are enabled, or unlocked, reading the **GPIOLOCK** register returns 0x00000000.

#### GPIO Lock (GPIOLOCK)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x520  
 Type R/W, reset 0x0000.0001



Bit/Field	Name	Type	Reset	Description
31:0	LOCK	R/W	0x0000.0001	GPIO Lock

A write of the value 0x4C4F.434B unlocks the **GPIO Commit (GPIOCR)** register for write access.

A write of any other value or a write to the **GPIOCR** register reapplies the lock, preventing any register updates. A read of this register returns the following values:

Value	Description
0x0000.0001	locked
0x0000.0000	unlocked

## Register 20: GPIO Commit (GPIOCR), offset 0x524

The **GPIOCR** register is the commit register. The value of the **GPIOCR** register determines which bits of the **GPIOAFSEL**, **GPIOPUR**, and **GIODEN** registers are committed when a write to these registers is performed. If a bit in the **GPIOCR** register is zero, the data being written to the corresponding bit in the **GPIOAFSEL**, **GPIOPUR**, or **GIODEN** registers cannot be committed and retains its previous value. If a bit in the **GPIOCR** register is set, the data being written to the corresponding bit of the **GPIOAFSEL**, **GPIOPUR**, or **GIODEN** registers is committed to the register and reflects the new value.

The contents of the **GPIOCR** register can only be modified if the **GPIOLOCK** register is unlocked. Writes to the **GPIOCR** register are ignored if the **GPIOLOCK** register is locked.

**Important:** This register is designed to prevent accidental programming of the registers that control connectivity to the NMI and JTAG/SWD debug hardware. By initializing the bits of the **GPIOCR** register to 0 for **PB7** and **PC[3:0]**, the NMI and JTAG/SWD debug port can only be converted to GPIOs through a deliberate set of writes to the **GPIOLOCK**, **GPIOCR**, and the corresponding registers.

Because this protection is currently only implemented on the NMI and JTAG/SWD pins on **PB7** and **PC[3:0]**, all of the other bits in the **GPIOCR** registers cannot be written with 0x0. These bits are hardwired to 0x1, ensuring that it is always possible to commit new values to the **GPIOAFSEL**, **GPIOPUR**, or **GIODEN** register bits of these other pins.

### GPIO Commit (GPIOCR)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x524  
 Type -, reset -

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
7:0	CR	-	-	<p>GPIO Commit</p> <p>On a bit-wise basis, any bit set allows the corresponding <code>GPIOAFSEL</code> bit to be set to its alternate function.</p> <p><b>Note:</b> The default register type for the <b>GPIOCR</b> register is RO for all GPIO pins, with the exception of the <code>NMI</code> pin and the four JTAG/SWD pins (<code>PB7</code> and <code>PC[3:0]</code>). These five pins are currently the only GPIOs that are protected by the <b>GPIOCR</b> register. Because of this, the register type for GPIO Port B7 and GPIO Port C[3:0] is R/W.</p> <p>The default reset value for the <b>GPIOCR</b> register is 0x0000.00FF for all GPIO pins, with the exception of the <code>NMI</code> pin and the four JTAG/SWD pins (<code>PB7</code> and <code>PC[3:0]</code>). To ensure that the JTAG port is not accidentally programmed as a GPIO, these four pins default to non-committable. To ensure that the <code>NMI</code> pin is not accidentally programmed as the non-maskable interrupt pin, it defaults to non-committable. Because of this, the default reset value of <b>GPIOCR</b> for GPIO Port B is 0x0000.007F while the default reset value of <b>GPIOCR</b> for Port C is 0x0000.00F0.</p>

## Register 21: GPIO Analog Mode Select (GPIOAMSEL), offset 0x528

**Note:** If any pin is to be used as an ADC input, the appropriate bit in **GPIOAMSEL** must be written to 1 to disable the analog isolation circuit.

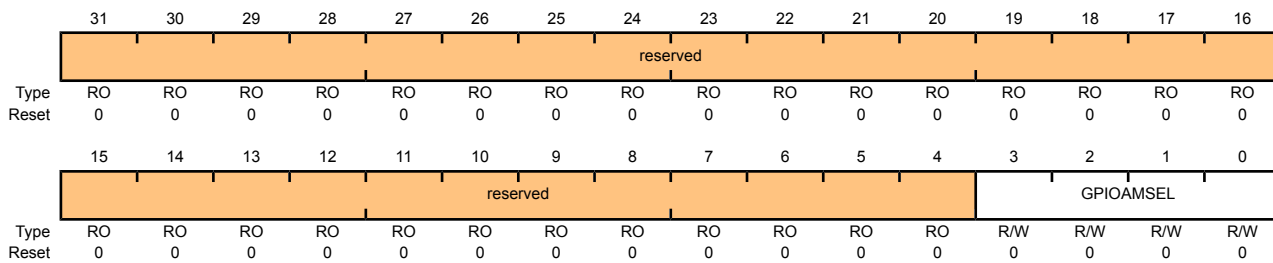
The **GPIOAMSEL** register controls isolation circuits to the analog side of a unified I/O pad. Because the GPIOs may be driven by a 5V source and affect analog operation, analog circuitry requires isolation from the pins when not used in their analog function.

Each bit of this register controls the isolation circuitry for circuits that share the same pin as the GPIO bit lane.

**Note:** This register is only valid for ports D and E.

### GPIO Analog Mode Select (GPIOAMSEL)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0x528  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	GPIOAMSEL	R/W	0x00	GPIO Analog Mode Select

#### Value Description

- 0 Analog function of the pin is disabled, the isolation is enabled, and the pin is capable of digital functions as specified by the other GPIO configuration registers.
- 1 Analog function of the pin is enabled, the isolation is disabled, and the pin is capable of analog functions.

**Note:** This register and bits are required only for GPIO bit lanes that share analog function through a unified I/O pad.

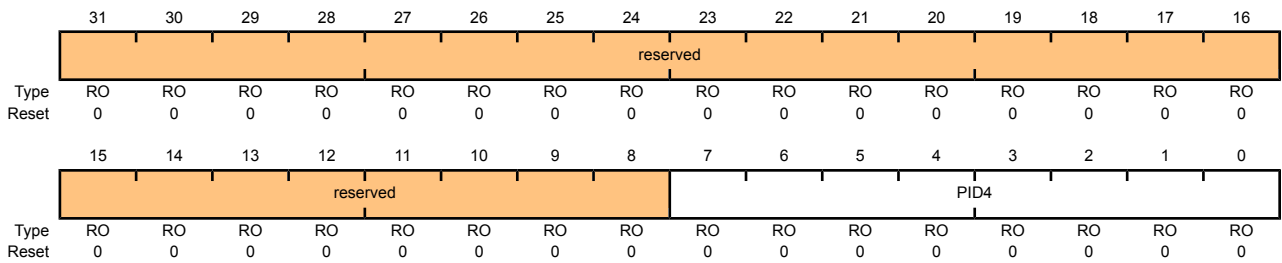
The reset state of this register is 0 for all bit lanes.

### Register 22: GPIO Peripheral Identification 4 (GPIOPeriphID4), offset 0xFD0

The **GPIOPeriphID4**, **GPIOPeriphID5**, **GPIOPeriphID6**, and **GPIOPeriphID7** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

#### GPIO Peripheral Identification 4 (GPIOPeriphID4)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0xFD0  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	GPIO Peripheral ID Register[7:0]

## Register 23: GPIO Peripheral Identification 5 (GPIOPeriphID5), offset 0xFD4

The **GPIOPeriphID4**, **GPIOPeriphID5**, **GPIOPeriphID6**, and **GPIOPeriphID7** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

### GPIO Peripheral Identification 5 (GPIOPeriphID5)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0xFD4  
 Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID5							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

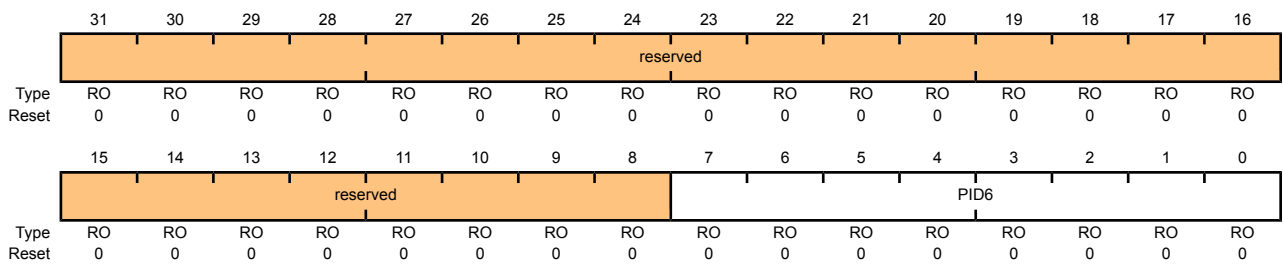
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	GPIO Peripheral ID Register[15:8]

### Register 24: GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8

The **GPIOPeriphID4**, **GPIOPeriphID5**, **GPIOPeriphID6**, and **GPIOPeriphID7** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

#### GPIO Peripheral Identification 6 (GPIOPeriphID6)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0xFD8  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	GPIO Peripheral ID Register[23:16]



**Register 25: GPIO Peripheral Identification 7 (GPIOPeriphID7), offset 0xFDC**

The **GPIOPeriphID4**, **GPIOPeriphID5**, **GPIOPeriphID6**, and **GPIOPeriphID7** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

## GPIO Peripheral Identification 7 (GPIOPeriphID7)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0xFDC  
 Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID7							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

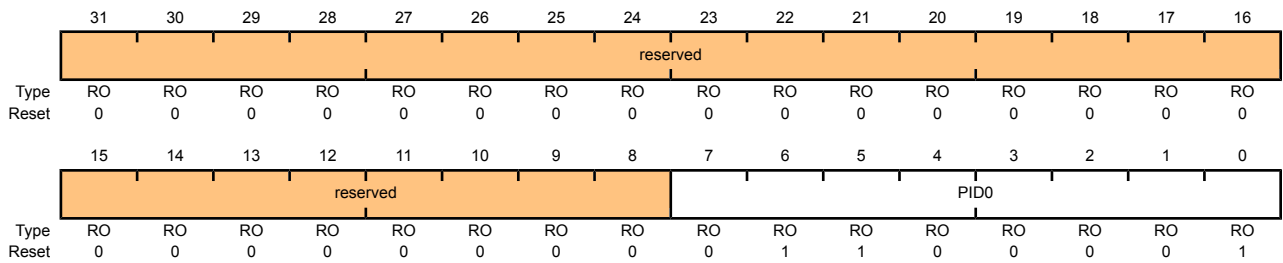
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	GPIO Peripheral ID Register[31:24]

### Register 26: GPIO Peripheral Identification 0 (GPIOPeriphID0), offset 0xFE0

The **GPIOPeriphID0**, **GPIOPeriphID1**, **GPIOPeriphID2**, and **GPIOPeriphID3** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

#### GPIO Peripheral Identification 0 (GPIOPeriphID0)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0xFE0  
 Type RO, reset 0x0000.0061



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x61	GPIO Peripheral ID Register[7:0]  Can be used by software to identify the presence of this peripheral.

## Register 27: GPIO Peripheral Identification 1 (GPIOPeriphID1), offset 0xFE4

The **GPIOPeriphID0**, **GPIOPeriphID1**, **GPIOPeriphID2**, and **GPIOPeriphID3** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

### GPIO Peripheral Identification 1 (GPIOPeriphID1)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0xFE4  
 Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

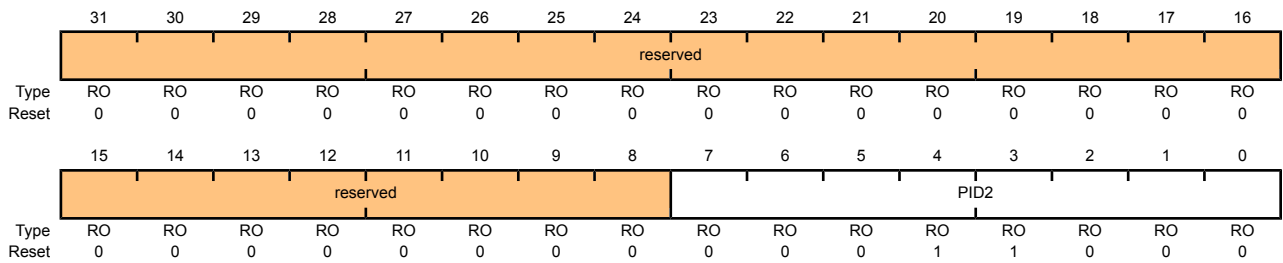
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	GPIO Peripheral ID Register[15:8]  Can be used by software to identify the presence of this peripheral.

### Register 28: GPIO Peripheral Identification 2 (GPIOPeriphID2), offset 0xFE8

The **GPIOPeriphID0**, **GPIOPeriphID1**, **GPIOPeriphID2**, and **GPIOPeriphID3** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

#### GPIO Peripheral Identification 2 (GPIOPeriphID2)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0xFE8  
 Type RO, reset 0x0000.0018



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	GPIO Peripheral ID Register[23:16]  Can be used by software to identify the presence of this peripheral.

## Register 29: GPIO Peripheral Identification 3 (GPIOPeriphID3), offset 0xFEC

The **GPIOPeriphID0**, **GPIOPeriphID1**, **GPIOPeriphID2**, and **GPIOPeriphID3** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

### GPIO Peripheral Identification 3 (GPIOPeriphID3)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0xFEC  
 Type RO, reset 0x0000.0001

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID3							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	GPIO Peripheral ID Register[31:24]  Can be used by software to identify the presence of this peripheral.

### Register 30: GPIO PrimeCell Identification 0 (GPIOCellID0), offset 0xFF0

The **GPIOCellID0**, **GPIOCellID1**, **GPIOCellID2**, and **GPIOCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

#### GPIO PrimeCell Identification 0 (GPIOCellID0)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0xFF0  
 Type RO, reset 0x0000.000D

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	GPIO PrimeCell ID Register[7:0]  Provides software a standard cross-peripheral identification system.

## Register 31: GPIO PrimeCell Identification 1 (GPIOCellID1), offset 0xFF4

The **GPIOCellID0**, **GPIOCellID1**, **GPIOCellID2**, and **GPIOCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

### GPIO PrimeCell Identification 1 (GPIOCellID1)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0xFF4  
 Type RO, reset 0x0000.00F0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

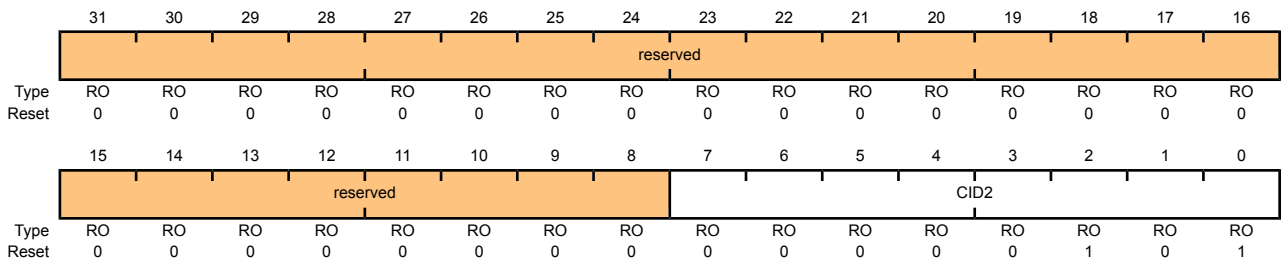
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	GPIO PrimeCell ID Register[15:8] Provides software a standard cross-peripheral identification system.

### Register 32: GPIO PrimeCell Identification 2 (GPIOCellID2), offset 0xFF8

The **GPIOCellID0**, **GPIOCellID1**, **GPIOCellID2**, and **GPIOCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

#### GPIO PrimeCell Identification 2 (GPIOCellID2)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0xFF8  
 Type RO, reset 0x0000.0005



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	GPIO PrimeCell ID Register[23:16]  Provides software a standard cross-peripheral identification system.



### Register 33: GPIO PrimeCell Identification 3 (GPIOCellID3), offset 0xFFC

The **GPIOCellID0**, **GPIOCellID1**, **GPIOCellID2**, and **GPIOCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

#### GPIO PrimeCell Identification 3 (GPIOCellID3)

GPIO Port A (legacy) base: 0x4000.4000  
 GPIO Port A (high-speed) base: 0x4005.8000  
 GPIO Port B (legacy) base: 0x4000.5000  
 GPIO Port B (high-speed) base: 0x4005.9000  
 GPIO Port C (legacy) base: 0x4000.6000  
 GPIO Port C (high-speed) base: 0x4005.A000  
 GPIO Port D (legacy) base: 0x4000.7000  
 GPIO Port D (high-speed) base: 0x4005.B000  
 GPIO Port E (legacy) base: 0x4002.4000  
 GPIO Port E (high-speed) base: 0x4005.C000  
 Offset 0xFFC  
 Type RO, reset 0x0000.00B1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID3							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	GPIO PrimeCell ID Register[31:24] Provides software a standard cross-peripheral identification system.

## 11 General-Purpose Timers

Programmable timers can be used to count or time external events that drive the Timer input pins. The Stellaris<sup>®</sup> General-Purpose Timer Module (GPTM) contains four GPTM blocks (Timer0, Timer1, Timer 2, and Timer 3). Each GPTM block provides two 16-bit timers/counters (referred to as TimerA and TimerB) that can be configured to operate independently as timers or event counters, or configured to operate as one 32-bit timer or one 32-bit Real-Time Clock (RTC). Timers can also be used to trigger analog-to-digital (ADC) conversions. The trigger signals from all of the general-purpose timers are ORed together before reaching the ADC module, so only one timer should be used to trigger ADC events.

The General-Purpose Timer Module is one timing resource available on the Stellaris<sup>®</sup> microcontrollers. Other timer resources include the System Timer (SysTick) (see “System Timer (SysTick)” on page 39).

The following modes are supported:

- 32-bit Timer modes
  - Programmable one-shot timer
  - Programmable periodic timer
  - Real-Time Clock using 32.768-KHz input clock
  - Software-controlled event stalling (excluding RTC mode)
- 16-bit Timer modes
  - General-purpose timer function with an 8-bit prescaler (for one-shot and periodic modes only)
  - Programmable one-shot timer
  - Programmable periodic timer
  - Software-controlled event stalling
- 16-bit Input Capture modes
  - Input edge count capture
  - Input edge time capture
- 16-bit PWM mode
  - Simple PWM mode with software-programmable output inversion of the PWM signal

### 11.1 Block Diagram

**Note:** In Figure 11-1 on page 283, the specific CCP pins available depend on the Stellaris<sup>®</sup> device. See Table 11-1 on page 283 for the available CCPs.

Figure 11-1. GPTM Module Block Diagram

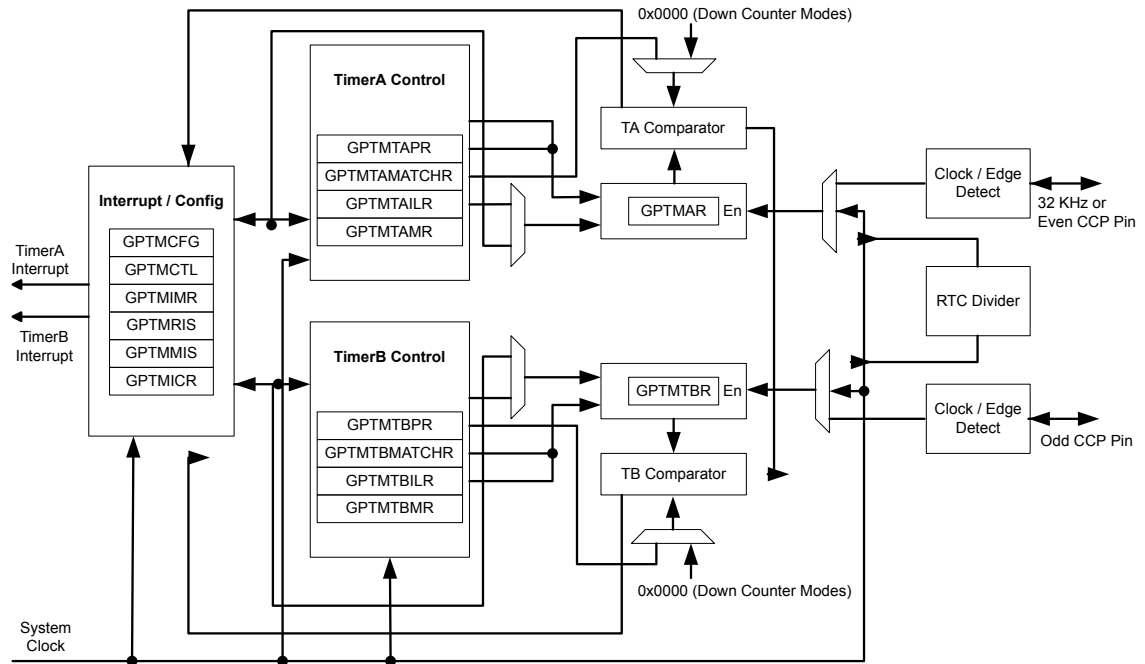


Table 11-1. Available CCP Pins

Timer	16-Bit Up/Down Counter	Even CCP Pin	Odd CCP Pin
Timer 0	TimerA	CCP0	-
	TimerB	-	CCP1
Timer 1	TimerA	CCP2	-
	TimerB	-	CCP3
Timer 2	TimerA	CCP4	-
	TimerB	-	CCP5
Timer 3	TimerA	-	-
	TimerB	-	-

## 11.2 Functional Description

The main components of each GPTM block are two free-running 16-bit up/down counters (referred to as TimerA and TimerB), two 16-bit match registers, and two 16-bit load/initialization registers and their associated control functions. The exact functionality of each GPTM is controlled by software and configured through the register interface.

Software configures the GPTM using the **GPTM Configuration (GPTMCFG)** register (see page 294), the **GPTM TimerA Mode (GPTMTAMR)** register (see page 295), and the **GPTM TimerB Mode (GPTMTBMR)** register (see page 297). When in one of the 32-bit modes, the timer can only act as a 32-bit timer. However, when configured in 16-bit mode, the GPTM can have its two 16-bit timers configured in any combination of the 16-bit modes.

### 11.2.1 GPTM Reset Conditions

After reset has been applied to the GPTM module, the module is in an inactive state, and all control registers are cleared and in their default states. Counters TimerA and TimerB are initialized to 0xFFFF, along with their corresponding load registers: the **GPTM TimerA Interval Load (GPTMTAILR)** register (see page 308) and the **GPTM TimerB Interval Load (GPTMTBILR)** register (see page 309). The prescale counters are initialized to 0x00: the **GPTM TimerA Prescale (GPTMTAPR)** register (see page 312) and the **GPTM TimerB Prescale (GPTMTBPR)** register (see page 313).

### 11.2.2 32-Bit Timer Operating Modes

This section describes the three GPTM 32-bit timer modes (One-Shot, Periodic, and RTC) and their configuration.

The GPTM is placed into 32-bit mode by writing a 0 (One-Shot/Periodic 32-bit timer mode) or a 1 (RTC mode) to the **GPTM Configuration (GPTMCFG)** register. In both configurations, certain GPTM registers are concatenated to form pseudo 32-bit registers. These registers include:

- **GPTM TimerA Interval Load (GPTMTAILR)** register [15:0], see page 308
- **GPTM TimerB Interval Load (GPTMTBILR)** register [15:0], see page 309
- **GPTM TimerA (GPTMTAR)** register [15:0], see page 314
- **GPTM TimerB (GPTMTBR)** register [15:0], see page 315

In the 32-bit modes, the GPTM translates a 32-bit write access to **GPTMTAILR** into a write access to both **GPTMTAILR** and **GPTMTBILR**. The resulting word ordering for such a write operation is:

```
GPTMTBILR[15:0]:GPTMTAILR[15:0]
```

Likewise, a read access to **GPTMTAR** returns the value:

```
GPTMTBR[15:0]:GPTMTAR[15:0]
```

#### 11.2.2.1 32-Bit One-Shot/Periodic Timer Mode

In 32-bit one-shot and periodic timer modes, the concatenated versions of the TimerA and TimerB registers are configured as a 32-bit down-counter. The selection of one-shot or periodic mode is determined by the value written to the **TAMR** field of the **GPTM TimerA Mode (GPTMTAMR)** register (see page 295), and there is no need to write to the **GPTM TimerB Mode (GPTMTBMR)** register.

When software writes the **TAEN** bit in the **GPTM Control (GPTMCTL)** register (see page 299), the timer begins counting down from its preloaded value. Once the 0x0000.0000 state is reached, the timer reloads its start value from the concatenated **GPTMTAILR** on the next cycle. If configured to be a one-shot timer, the timer stops counting and clears the **TAEN** bit in the **GPTMCTL** register. If configured as a periodic timer, it continues counting.

In addition to reloading the count value, the GPTM generates interrupts and triggers when it reaches the 0x000.0000 state. The GPTM sets the **TATORIS** bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register (see page 304), and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register (see page 306). If the time-out interrupt is enabled in the **GPTM Interrupt Mask (GPTIMR)** register (see page 302), the GPTM also sets the **TATOMIS** bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register (see page 305). The trigger is enabled by setting the **TAOTE** bit in **GPTMCTL**, and can trigger SoC-level events such as ADC conversions.

If software reloads the **GPTMTAILR** register while the counter is running, the counter loads the new value on the next clock cycle and continues counting from the new value.

If the **TASTALL** bit in the **GPTMCTL** register is asserted, the timer freezes counting until the signal is deasserted.

### 11.2.2.2 32-Bit Real-Time Clock Timer Mode

In Real-Time Clock (RTC) mode, the concatenated versions of the TimerA and TimerB registers are configured as a 32-bit up-counter. When RTC mode is selected for the first time, the counter is loaded with a value of 0x0000.0001. All subsequent load values must be written to the **GPTM TimerA Match (GPTMTAMATCHR)** register (see page 310) by the controller.

The input clock on the **CCP0**, **CCP2**, or **CCP4** pins is required to be 32.768 KHz in RTC mode. The clock signal is then divided down to a 1 Hz rate and is passed along to the input of the 32-bit counter.

When software writes the **TAEN** bit in the **GPTMCTL** register, the counter starts counting up from its preloaded value of 0x0000.0001. When the current count value matches the preloaded value in the **GPTMTAMATCHR** register, it rolls over to a value of 0x0000.0000 and continues counting until either a hardware reset, or it is disabled by software (clearing the **TAEN** bit). When a match occurs, the GPTM asserts the **RTCRES** bit in **GPTMRIS**. If the RTC interrupt is enabled in **GPTIMR**, the GPTM also sets the **RTCMIS** bit in **GPTMISR** and generates a controller interrupt. The status flags are cleared by writing the **RTCCINT** bit in **GPTMICR**.

If the **TASTALL** and/or **TBSTALL** bits in the **GPTMCTL** register are set, the timer does not freeze if the **RTCEN** bit is set in **GPTMCTL**.

### 11.2.3 16-Bit Timer Operating Modes

The GPTM is placed into global 16-bit mode by writing a value of 0x4 to the **GPTM Configuration (GPTMCFG)** register (see page 294). This section describes each of the GPTM 16-bit modes of operation. TimerA and TimerB have identical modes, so a single description is given using an *n* to reference both.

#### 11.2.3.1 16-Bit One-Shot/Periodic Timer Mode

In 16-bit one-shot and periodic timer modes, the timer is configured as a 16-bit down-counter with an optional 8-bit prescaler that effectively extends the counting range of the timer to 24 bits. The selection of one-shot or periodic mode is determined by the value written to the **TnMR** field of the **GPTMTnMR** register. The optional prescaler is loaded into the **GPTM Timern Prescale (GPTMTnPR)** register.

When software writes the **TnEN** bit in the **GPTMCTL** register, the timer begins counting down from its preloaded value. Once the 0x0000 state is reached, the timer reloads its start value from **GPTMTnILR** and **GPTMTnPR** on the next cycle. If configured to be a one-shot timer, the timer stops counting and clears the **TnEN** bit in the **GPTMCTL** register. If configured as a periodic timer, it continues counting.

In addition to reloading the count value, the timer generates interrupts and triggers when it reaches the 0x0000 state. The GPTM sets the **TnTORIS** bit in the **GPTMRIS** register, and holds it until it is cleared by writing the **GPTMICR** register. If the time-out interrupt is enabled in **GPTIMR**, the GPTM also sets the **TnTOMIS** bit in **GPTMISR** and generates a controller interrupt. The trigger is enabled by setting the **TnOTE** bit in the **GPTMCTL** register, and can trigger SoC-level events such as ADC conversions.

If software reloads the **GPTMTAILR** register while the counter is running, the counter loads the new value on the next clock cycle and continues counting from the new value.

If the  $T_{nSTALL}$  bit in the **GPTMCTL** register is enabled, the timer freezes counting until the signal is deasserted.

The following example shows a variety of configurations for a 16-bit free running timer while using the prescaler. All values assume a 50-MHz clock with  $T_c=20$  ns (clock period).

**Table 11-2. 16-Bit Timer With Prescaler Configurations**

Prescale	#Clock (T c) <sup>a</sup>	Max Time	Units
00000000	1	1.3107	mS
00000001	2	2.6214	mS
00000010	3	3.9321	mS
-----	--	--	--
11111100	254	332.9229	mS
11111110	255	334.2336	mS
11111111	256	335.5443	mS

a.  $T_c$  is the clock period.

### 11.2.3.2 16-Bit Input Edge Count Mode

**Note:** For rising-edge detection, the input signal must be High for at least two system clock periods following the rising edge. Similarly, for falling-edge detection, the input signal must be Low for at least two system clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is 1/4 of the system frequency.

**Note:** The prescaler is not available in 16-Bit Input Edge Count mode.

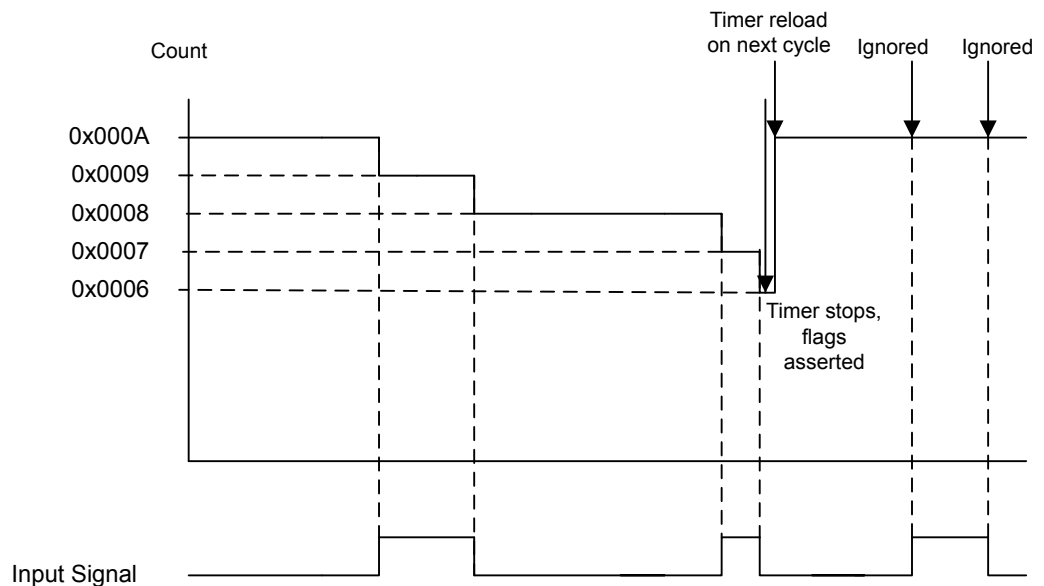
In Edge Count mode, the timer is configured as a down-counter capable of capturing three types of events: rising edge, falling edge, or both. To place the timer in Edge Count mode, the  $T_{nCMR}$  bit of the **GPTMTnMR** register must be set to 0. The type of edge that the timer counts is determined by the  $T_{nEVENT}$  fields of the **GPTMCTL** register. During initialization, the **GPTM Timern Match (GPTMTnMATCHR)** register is configured so that the difference between the value in the **GPTMTnILR** register and the **GPTMTnMATCHR** register equals the number of edge events that must be counted.

When software writes the  $T_{nEN}$  bit in the **GPTM Control (GPTMCTL)** register, the timer is enabled for event capture. Each input event on the CCP pin decrements the counter by 1 until the event count matches **GPTMTnMATCHR**. When the counts match, the GPTM asserts the  $C_{nMRIS}$  bit in the **GPTMRIS** register (and the  $C_{nMMIS}$  bit, if the interrupt is not masked). The counter is then reloaded using the value in **GPTMTnILR**, and stopped since the GPTM automatically clears the  $T_{nEN}$  bit in the **GPTMCTL** register. Once the event count has been reached, all further events are ignored until  $T_{nEN}$  is re-enabled by software.

Figure 11-2 on page 287 shows how input edge count mode works. In this case, the timer start value is set to **GPTMTnILR** = 0x000A and the match value is set to **GPTMTnMATCHR** = 0x0006 so that four edge events are counted. The counter is configured to detect both edges of the input signal.

Note that the last two edges are not counted since the timer automatically clears the  $T_{nEN}$  bit after the current count matches the value in the **GPTMTnMR** register.

Figure 11-2. 16-Bit Input Edge Count Mode Example



### 11.2.3.3 16-Bit Input Edge Time Mode

**Note:** For rising-edge detection, the input signal must be High for at least two system clock periods following the rising edge. Similarly, for falling edge detection, the input signal must be Low for at least two system clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is 1/4 of the system frequency.

**Note:** The prescaler is not available in 16-Bit Input Edge Time mode.

In Edge Time mode, the timer is configured as a free-running down-counter initialized to the value loaded in the **GPTMTnILR** register (or 0xFFFF at reset). This mode allows for event capture of either rising or falling edges, but not both. The timer is placed into Edge Time mode by setting the **TnCMR** bit in the **GPTMTnMR** register, and the type of event that the timer captures is determined by the **TnEVENT** fields of the **GPTMCnTL** register.

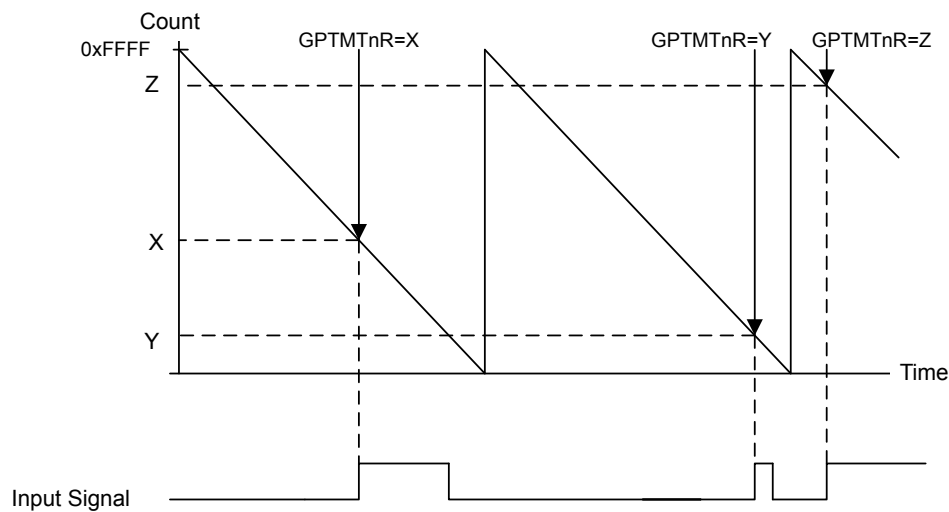
When software writes the **TnEN** bit in the **GPTMCTL** register, the timer is enabled for event capture. When the selected input event is detected, the current **Tn** counter value is captured in the **GPTMTnR** register and is available to be read by the controller. The GPTM then asserts the **CnERIS** bit (and the **CnEMIS** bit, if the interrupt is not masked).

After an event has been captured, the timer does not stop counting. It continues to count until the **TnEN** bit is cleared. When the timer reaches the 0x0000 state, it is reloaded with the value from the **GPTMnILR** register.

Figure 11-3 on page 288 shows how input edge timing mode works. In the diagram, it is assumed that the start value of the timer is the default value of 0xFFFF, and the timer is configured to capture rising edge events.

Each time a rising edge event is detected, the current count value is loaded into the **GPTMTnR** register, and is held there until another rising edge is detected (at which point the new count value is loaded into **GPTMTnR**).

Figure 11-3. 16-Bit Input Edge Time Mode Example



#### 11.2.3.4 16-Bit PWM Mode

**Note:** The prescaler is not available in 16-Bit PWM mode.

The GPTM supports a simple PWM generation mode. In PWM mode, the timer is configured as a down-counter with a start value (and thus period) defined by **GPTMTnILR**. PWM mode is enabled with the **GPTMTnMR** register by setting the  $T_nAMS$  bit to 0x1, the  $T_nCMR$  bit to 0x0, and the  $T_nMR$  field to 0x2.

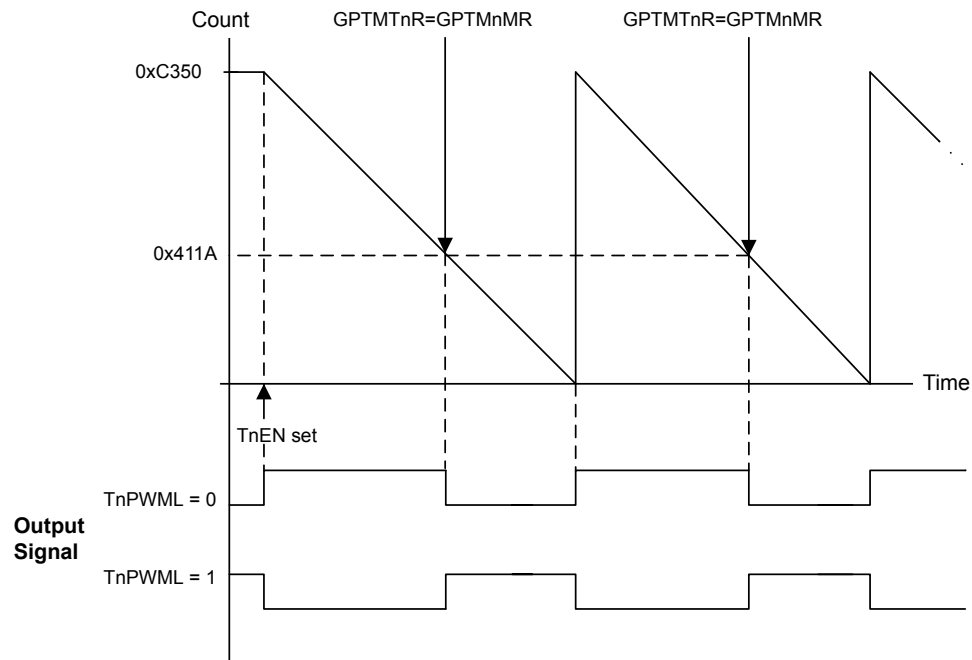
When software writes the  $T_nEN$  bit in the **GPTMCTL** register, the counter begins counting down until it reaches the 0x0000 state. On the next counter cycle, the counter reloads its start value from **GPTMTnILR** and continues counting until disabled by software clearing the  $T_nEN$  bit in the **GPTMCTL** register. No interrupts or status bits are asserted in PWM mode.

The output PWM signal asserts when the counter is at the value of the **GPTMTnILR** register (its start state), and is deasserted when the counter value equals the value in the **GPTM Timern Match Register (GPTMnMATCHR)**. Software has the capability of inverting the output PWM signal by setting the  $T_nPWML$  bit in the **GPTMCTL** register.

Figure 11-4 on page 289 shows how to generate an output PWM with a 1-ms period and a 66% duty cycle assuming a 50-MHz input clock and  $T_nPWML = 0$  (duty cycle would be 33% for the  $T_nPWML = 1$  configuration). For this example, the start value is **GPTMnILR=0xC350** and the match value is **GPTMnMR=0x411A**.



Figure 11-4. 16-Bit PWM Mode Example



## 11.3 Initialization and Configuration

To use the general-purpose timers, the peripheral clock must be enabled by setting the `TIMER0`, `TIMER1`, `TIMER2`, and `TIMER3` bits in the `RCGC1` register.

This section shows module initialization and configuration examples for each of the supported timer modes.

### 11.3.1 32-Bit One-Shot/Periodic Timer Mode

The GPTM is configured for 32-bit One-Shot and Periodic modes by the following sequence:

1. Ensure the timer is disabled (the `TAEN` bit in the `GPTMCTL` register is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x0.
3. Set the `TAMR` field in the **GPTM TimerA Mode Register (GPTMTAMR)**:
  - a. Write a value of 0x1 for One-Shot mode.
  - b. Write a value of 0x2 for Periodic mode.
4. Load the start value into the **GPTM TimerA Interval Load Register (GPTMTAILR)**.
5. If interrupts are required, set the `TATOIM` bit in the **GPTM Interrupt Mask Register (GPTMIMR)**.
6. Set the `TAEN` bit in the `GPTMCTL` register to enable the timer and start counting.

7. Poll the `TATORIS` bit in the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the `TATOCINT` bit of the **GPTM Interrupt Clear Register (GPTMICR)**.

In One-Shot mode, the timer stops counting after step 7 on page 290. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode does not stop counting after it times out.

### 11.3.2 32-Bit Real-Time Clock (RTC) Mode

To use the RTC mode, the timer must have a 32.768-KHz input signal on its `CCP0`, `CCP2`, or `CCP4` pins. To enable the RTC feature, follow these steps:

1. Ensure the timer is disabled (the `TAEN` bit is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x1.
3. Write the desired match value to the **GPTM TimerA Match Register (GPTMTAMATCHR)**.
4. Set/clear the `RTCEN` bit in the **GPTM Control Register (GPTMCTL)** as desired.
5. If interrupts are required, set the `RTCIM` bit in the **GPTM Interrupt Mask Register (GPTMIMR)**.
6. Set the `TAEN` bit in the **GPTMCTL** register to enable the timer and start counting.

When the timer count equals the value in the **GPTMTAMATCHR** register, the counter is re-loaded with 0x0000.0000 and begins counting. If an interrupt is enabled, it does not have to be cleared.

### 11.3.3 16-Bit One-Shot/Periodic Timer Mode

A timer is configured for 16-bit One-Shot and Periodic modes by the following sequence:

1. Ensure the timer is disabled (the `TnEN` bit is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x4.
3. Set the `TnMR` field in the **GPTM Timer Mode (GPTMTnMR)** register:
  - a. Write a value of 0x1 for One-Shot mode.
  - b. Write a value of 0x2 for Periodic mode.
4. If a prescaler is to be used, write the prescale value to the **GPTM Timern Prescale Register (GPTMTnPR)**.
5. Load the start value into the **GPTM Timer Interval Load Register (GPTMTnILR)**.
6. If interrupts are required, set the `TnTOIM` bit in the **GPTM Interrupt Mask Register (GPTMIMR)**.
7. Set the `TnEN` bit in the **GPTM Control Register (GPTMCTL)** to enable the timer and start counting.
8. Poll the `TnTORIS` bit in the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the `TnTOCINT` bit of the **GPTM Interrupt Clear Register (GPTMICR)**.

In One-Shot mode, the timer stops counting after step 8 on page 290. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode does not stop counting after it times out.

### 11.3.4 16-Bit Input Edge Count Mode

A timer is configured to Input Edge Count mode by the following sequence:

1. Ensure the timer is disabled (the  $TnEN$  bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x4.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the  $TnCMR$  field to 0x0 and the  $TnMR$  field to 0x3.
4. Configure the type of event(s) that the timer captures by writing the  $TnEVENT$  field of the **GPTM Control (GPTMCTL)** register.
5. Load the timer start value into the **GPTM Timern Interval Load (GPTMTnILR)** register.
6. Load the desired event count into the **GPTM Timern Match (GPTMTnMATCHR)** register.
7. If interrupts are required, set the  $CnMIM$  bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
8. Set the  $TnEN$  bit in the **GPTMCTL** register to enable the timer and begin waiting for edge events.
9. Poll the  $CnMRIS$  bit in the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the  $CnMCINT$  bit of the **GPTM Interrupt Clear (GPTMICR)** register.

In Input Edge Count Mode, the timer stops after the desired number of edge events has been detected. To re-enable the timer, ensure that the  $TnEN$  bit is cleared and repeat step 4 on page 291 through step 9 on page 291.

### 11.3.5 16-Bit Input Edge Timing Mode

A timer is configured to Input Edge Timing mode by the following sequence:

1. Ensure the timer is disabled (the  $TnEN$  bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x4.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the  $TnCMR$  field to 0x1 and the  $TnMR$  field to 0x3.
4. Configure the type of event that the timer captures by writing the  $TnEVENT$  field of the **GPTM Control (GPTMCTL)** register.
5. Load the timer start value into the **GPTM Timern Interval Load (GPTMTnILR)** register.
6. If interrupts are required, set the  $CnEIM$  bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
7. Set the  $TnEN$  bit in the **GPTM Control (GPTMCTL)** register to enable the timer and start counting.
8. Poll the  $CnERIS$  bit in the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the  $CnECINT$  bit of the **GPTM**

**Interrupt Clear (GPTMICR)** register. The time at which the event happened can be obtained by reading the **GPTM Timern (GPTMTnR)** register.

In Input Edge Timing mode, the timer continues running after an edge event has been detected, but the timer interval can be changed at any time by writing the **GPTMTnILR** register. The change takes effect at the next cycle after the write.

### 11.3.6 16-Bit PWM Mode

A timer is configured to PWM mode using the following sequence:

1. Ensure the timer is disabled (the  $TnEN$  bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x4.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, set the  $TnAMS$  bit to 0x1, the  $TnCMR$  bit to 0x0, and the  $TnMR$  field to 0x2.
4. Configure the output state of the PWM signal (whether or not it is inverted) in the  $TnEVENT$  field of the **GPTM Control (GPTMCTL)** register.
5. Load the timer start value into the **GPTM Timern Interval Load (GPTMTnILR)** register.
6. Load the **GPTM Timern Match (GPTMTnMATCHR)** register with the desired value.
7. Set the  $TnEN$  bit in the **GPTM Control (GPTMCTL)** register to enable the timer and begin generation of the output PWM signal.

In PWM Timing mode, the timer continues running after the PWM signal has been generated. The PWM period can be adjusted at any time by writing the **GPTMTnILR** register, and the change takes effect at the next cycle after the write.

## 11.4 Register Map

Table 11-3 on page 292 lists the GPTM registers. The offset listed is a hexadecimal increment to the register's address, relative to that timer's base address:

- Timer0: 0x4003.0000
- Timer1: 0x4003.1000
- Timer2: 0x4003.2000
- Timer3: 0x4003.3000

**Table 11-3. Timers Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	GPTMCFG	R/W	0x0000.0000	GPTM Configuration	294
0x004	GPTMTAMR	R/W	0x0000.0000	GPTM TimerA Mode	295
0x008	GPTMTBMR	R/W	0x0000.0000	GPTM TimerB Mode	297
0x00C	GPTMCTL	R/W	0x0000.0000	GPTM Control	299

Offset	Name	Type	Reset	Description	See page
0x018	GPTMIMR	R/W	0x0000.0000	GPTM Interrupt Mask	302
0x01C	GPTMRIS	RO	0x0000.0000	GPTM Raw Interrupt Status	304
0x020	GPTMMIS	RO	0x0000.0000	GPTM Masked Interrupt Status	305
0x024	GPTMICR	W1C	0x0000.0000	GPTM Interrupt Clear	306
0x028	GPTMTAILR	R/W	0x0000.FFFF (16-bit mode) 0xFFFF.FFFF (32-bit mode)	GPTM TimerA Interval Load	308
0x02C	GPTMTBILR	R/W	0x0000.FFFF	GPTM TimerB Interval Load	309
0x030	GPTMTAMATCHR	R/W	0x0000.FFFF (16-bit mode) 0xFFFF.FFFF (32-bit mode)	GPTM TimerA Match	310
0x034	GPTMTBMATCHR	R/W	0x0000.FFFF	GPTM TimerB Match	311
0x038	GPTMTAPR	R/W	0x0000.0000	GPTM TimerA Prescale	312
0x03C	GPTMTBPR	R/W	0x0000.0000	GPTM TimerB Prescale	313
0x048	GPTMTAR	RO	0x0000.FFFF (16-bit mode) 0xFFFF.FFFF (32-bit mode)	GPTM TimerA	314
0x04C	GPTMTBR	RO	0x0000.FFFF	GPTM TimerB	315

## 11.5 Register Descriptions

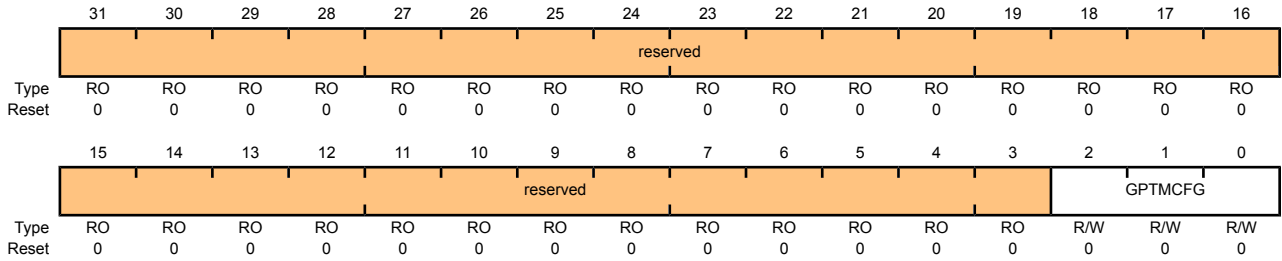
The remainder of this section lists and describes the GPTM registers, in numerical order by address offset.

### Register 1: GPTM Configuration (GPTMCFG), offset 0x000

This register configures the global operation of the GPTM module. The value written to this register determines whether the GPTM is in 32- or 16-bit mode.

#### GPTM Configuration (GPTMCFG)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x000  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	GPTMCFG	R/W	0x0	GPTM Configuration

The GPTMCFG values are defined as follows:

Value	Description
0x0	32-bit timer configuration.
0x1	32-bit real-time clock (RTC) counter configuration.
0x2	Reserved
0x3	Reserved
0x4-0x7	16-bit timer configuration, function is controlled by bits 1:0 of <b>GPTMTAMR</b> and <b>GPTMTBMR</b> .

## Register 2: GPTM TimerA Mode (GPTMTAMR), offset 0x004

This register configures the GPTM based on the configuration selected in the **GPTMCFG** register. When in 16-bit PWM mode, set the **TAAMS** bit to 0x1, the **TACMR** bit to 0x0, and the **TAMR** field to 0x2.

### GPTM TimerA Mode (GPTMTAMR)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x004  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												TAAMS	TACMR	TAMR	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description						
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
3	TAAMS	R/W	0	GPTM TimerA Alternate Mode Select  The <b>TAAMS</b> values are defined as follows:  <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Capture mode is enabled.</td> </tr> <tr> <td>1</td> <td>PWM mode is enabled.</td> </tr> </tbody> </table> <p><b>Note:</b> To enable PWM mode, you must also clear the <b>TACMR</b> bit and set the <b>TAMR</b> field to 0x2.</p>	Value	Description	0	Capture mode is enabled.	1	PWM mode is enabled.
Value	Description									
0	Capture mode is enabled.									
1	PWM mode is enabled.									
2	TACMR	R/W	0	GPTM TimerA Capture Mode  The <b>TACMR</b> values are defined as follows:  <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Edge-Count mode</td> </tr> <tr> <td>1</td> <td>Edge-Time mode</td> </tr> </tbody> </table>	Value	Description	0	Edge-Count mode	1	Edge-Time mode
Value	Description									
0	Edge-Count mode									
1	Edge-Time mode									

Bit/Field	Name	Type	Reset	Description										
1:0	TAMR	R/W	0x0	<p>GPTM TimerA Mode</p> <p>The <code>TAMR</code> values are defined as follows:</p> <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0x0</td><td>Reserved</td></tr><tr><td>0x1</td><td>One-Shot Timer mode</td></tr><tr><td>0x2</td><td>Periodic Timer mode</td></tr><tr><td>0x3</td><td>Capture mode</td></tr></tbody></table> <p>The Timer mode is based on the timer configuration defined by bits 2:0 in the <b>GPTMCFG</b> register (16-or 32-bit).</p> <p>In 16-bit timer configuration, <code>TAMR</code> controls the 16-bit timer modes for TimerA.</p> <p>In 32-bit timer configuration, this register controls the mode and the contents of <b>GPTMTBMR</b> are ignored.</p>	Value	Description	0x0	Reserved	0x1	One-Shot Timer mode	0x2	Periodic Timer mode	0x3	Capture mode
Value	Description													
0x0	Reserved													
0x1	One-Shot Timer mode													
0x2	Periodic Timer mode													
0x3	Capture mode													

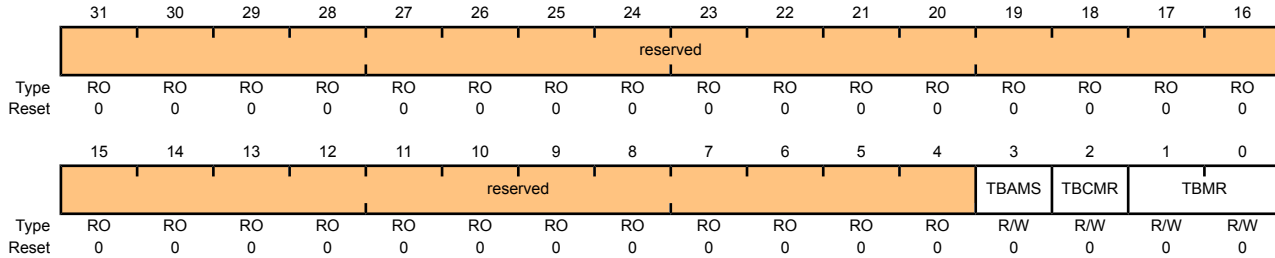


### Register 3: GPTM TimerB Mode (GPTMTBMR), offset 0x008

This register configures the GPTM based on the configuration selected in the **GPTMCFG** register. When in 16-bit PWM mode, set the **TBAMS** bit to 0x1, the **TBCMR** bit to 0x0, and the **TBMR** field to 0x2.

#### GPTM TimerB Mode (GPTMTBMR)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x008  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
3	TBAMS	R/W	0	GPTM TimerB Alternate Mode Select  The <b>TBAMS</b> values are defined as follows:  <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Capture mode is enabled.</td> </tr> <tr> <td>1</td> <td>PWM mode is enabled.</td> </tr> </tbody> </table> <p><b>Note:</b> To enable PWM mode, you must also clear the <b>TBCMR</b> bit and set the <b>TBMR</b> field to 0x2.</p>	Value	Description	0	Capture mode is enabled.	1	PWM mode is enabled.
Value	Description									
0	Capture mode is enabled.									
1	PWM mode is enabled.									
2	TBCMR	R/W	0	GPTM TimerB Capture Mode  The <b>TBCMR</b> values are defined as follows:  <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Edge-Count mode</td> </tr> <tr> <td>1</td> <td>Edge-Time mode</td> </tr> </tbody> </table>	Value	Description	0	Edge-Count mode	1	Edge-Time mode
Value	Description									
0	Edge-Count mode									
1	Edge-Time mode									

Bit/Field	Name	Type	Reset	Description										
1:0	TBMR	R/W	0x0	<p>GPTM TimerB Mode</p> <p>The TBMR values are defined as follows:</p> <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0x0</td><td>Reserved</td></tr><tr><td>0x1</td><td>One-Shot Timer mode</td></tr><tr><td>0x2</td><td>Periodic Timer mode</td></tr><tr><td>0x3</td><td>Capture mode</td></tr></tbody></table> <p>The timer mode is based on the timer configuration defined by bits 2:0 in the <b>GPTMCFG</b> register.</p> <p>In 16-bit timer configuration, these bits control the 16-bit timer modes for TimerB.</p> <p>In 32-bit timer configuration, this register's contents are ignored and <b>GPTMTAMR</b> is used.</p>	Value	Description	0x0	Reserved	0x1	One-Shot Timer mode	0x2	Periodic Timer mode	0x3	Capture mode
Value	Description													
0x0	Reserved													
0x1	One-Shot Timer mode													
0x2	Periodic Timer mode													
0x3	Capture mode													

## Register 4: GPTM Control (GPTMCTL), offset 0x00C

This register is used alongside the **GPTMCFG** and **GMTMTnMR** registers to fine-tune the timer configuration, and to enable other features such as timer stall and the output trigger. The output trigger can be used to initiate transfers on the ADC module.

### GPTM Control (GPTMCTL)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x00C  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	TBPWML	TBOTE	reserved	TBEVENT	TBSTALL	TBEN	reserved	TAPWML	TAOTE	RTCEN	TAEVENT	TASTALL	TAEN		
Type	RO	R/W	R/W	RO	R/W	R/W	R/W	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:15	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	TBPWML	R/W	0	GPTM TimerB PWM Output Level  The <b>TBPWML</b> values are defined as follows:  Value Description 0 Output is unaffected. 1 Output is inverted.
13	TBOTE	R/W	0	GPTM TimerB Output Trigger Enable  The <b>TBOTE</b> values are defined as follows:  Value Description 0 The output TimerB trigger is disabled. 1 The output TimerB trigger is enabled.
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description										
11:10	TBEVENT	R/W	0x0	<p>GPTM TimerB Event Mode</p> <p>The TBEVENT values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Positive edge</td> </tr> <tr> <td>0x1</td> <td>Negative edge</td> </tr> <tr> <td>0x2</td> <td>Reserved</td> </tr> <tr> <td>0x3</td> <td>Both edges</td> </tr> </tbody> </table>	Value	Description	0x0	Positive edge	0x1	Negative edge	0x2	Reserved	0x3	Both edges
Value	Description													
0x0	Positive edge													
0x1	Negative edge													
0x2	Reserved													
0x3	Both edges													
9	TBSTALL	R/W	0	<p>GPTM TimerB Stall Enable</p> <p>The TBSTALL values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>TimerB stalling is disabled.</td> </tr> <tr> <td>1</td> <td>TimerB stalling is enabled.</td> </tr> </tbody> </table>	Value	Description	0	TimerB stalling is disabled.	1	TimerB stalling is enabled.				
Value	Description													
0	TimerB stalling is disabled.													
1	TimerB stalling is enabled.													
8	TBEN	R/W	0	<p>GPTM TimerB Enable</p> <p>The TBEN values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>TimerB is disabled.</td> </tr> <tr> <td>1</td> <td>TimerB is enabled and begins counting or the capture logic is enabled based on the <b>GPTMCFG</b> register.</td> </tr> </tbody> </table>	Value	Description	0	TimerB is disabled.	1	TimerB is enabled and begins counting or the capture logic is enabled based on the <b>GPTMCFG</b> register.				
Value	Description													
0	TimerB is disabled.													
1	TimerB is enabled and begins counting or the capture logic is enabled based on the <b>GPTMCFG</b> register.													
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
6	TAPWML	R/W	0	<p>GPTM TimerA PWM Output Level</p> <p>The TAPWML values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Output is unaffected.</td> </tr> <tr> <td>1</td> <td>Output is inverted.</td> </tr> </tbody> </table>	Value	Description	0	Output is unaffected.	1	Output is inverted.				
Value	Description													
0	Output is unaffected.													
1	Output is inverted.													
5	TAOTE	R/W	0	<p>GPTM TimerA Output Trigger Enable</p> <p>The TAOTE values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The output TimerA trigger is disabled.</td> </tr> <tr> <td>1</td> <td>The output TimerA trigger is enabled.</td> </tr> </tbody> </table>	Value	Description	0	The output TimerA trigger is disabled.	1	The output TimerA trigger is enabled.				
Value	Description													
0	The output TimerA trigger is disabled.													
1	The output TimerA trigger is enabled.													

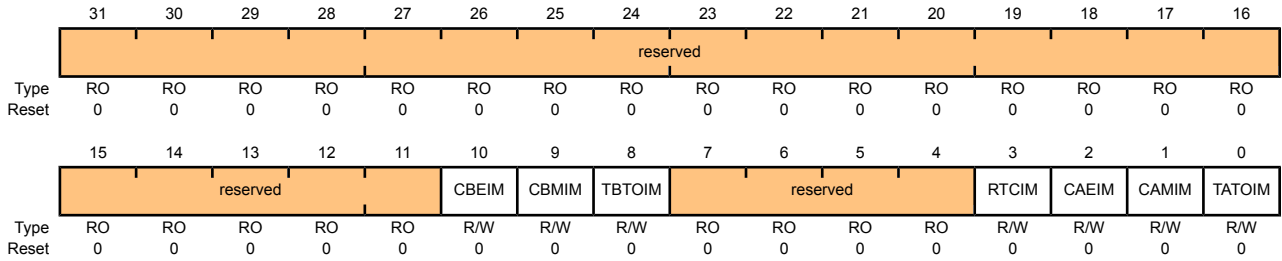
Bit/Field	Name	Type	Reset	Description										
4	RTCEN	R/W	0	<p>GPTM RTC Enable</p> <p>The <code>RTCEN</code> values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>RTC counting is disabled.</td> </tr> <tr> <td>1</td> <td>RTC counting is enabled.</td> </tr> </tbody> </table>	Value	Description	0	RTC counting is disabled.	1	RTC counting is enabled.				
Value	Description													
0	RTC counting is disabled.													
1	RTC counting is enabled.													
3:2	TAEVENT	R/W	0x0	<p>GPTM TimerA Event Mode</p> <p>The <code>TAEVENT</code> values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Positive edge</td> </tr> <tr> <td>0x1</td> <td>Negative edge</td> </tr> <tr> <td>0x2</td> <td>Reserved</td> </tr> <tr> <td>0x3</td> <td>Both edges</td> </tr> </tbody> </table>	Value	Description	0x0	Positive edge	0x1	Negative edge	0x2	Reserved	0x3	Both edges
Value	Description													
0x0	Positive edge													
0x1	Negative edge													
0x2	Reserved													
0x3	Both edges													
1	TASTALL	R/W	0	<p>GPTM TimerA Stall Enable</p> <p>The <code>TASTALL</code> values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>TimerA stalling is disabled.</td> </tr> <tr> <td>1</td> <td>TimerA stalling is enabled.</td> </tr> </tbody> </table>	Value	Description	0	TimerA stalling is disabled.	1	TimerA stalling is enabled.				
Value	Description													
0	TimerA stalling is disabled.													
1	TimerA stalling is enabled.													
0	TAEN	R/W	0	<p>GPTM TimerA Enable</p> <p>The <code>TAEN</code> values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>TimerA is disabled.</td> </tr> <tr> <td>1</td> <td>TimerA is enabled and begins counting or the capture logic is enabled based on the <code>GPTMCFG</code> register.</td> </tr> </tbody> </table>	Value	Description	0	TimerA is disabled.	1	TimerA is enabled and begins counting or the capture logic is enabled based on the <code>GPTMCFG</code> register.				
Value	Description													
0	TimerA is disabled.													
1	TimerA is enabled and begins counting or the capture logic is enabled based on the <code>GPTMCFG</code> register.													

### Register 5: GPTM Interrupt Mask (GPTMIMR), offset 0x018

This register allows software to enable/disable GPTM controller-level interrupts. Writing a 1 enables the interrupt, while writing a 0 disables it.

#### GPTM Interrupt Mask (GPTMIMR)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x018  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	CBEIM	R/W	0	GPTM CaptureB Event Interrupt Mask The CBEIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
9	CBMIM	R/W	0	GPTM CaptureB Match Interrupt Mask The CBMIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
8	TBTOIM	R/W	0	GPTM TimerB Time-Out Interrupt Mask The TBTOIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
7:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

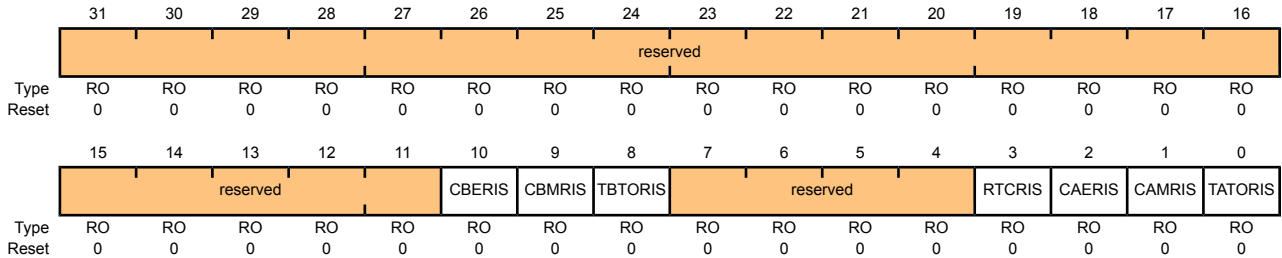
Bit/Field	Name	Type	Reset	Description						
3	RTCIM	R/W	0	<p>GPTM RTC Interrupt Mask</p> <p>The RTCIM values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Interrupt is disabled.</td> </tr> <tr> <td>1</td> <td>Interrupt is enabled.</td> </tr> </tbody> </table>	Value	Description	0	Interrupt is disabled.	1	Interrupt is enabled.
Value	Description									
0	Interrupt is disabled.									
1	Interrupt is enabled.									
2	CAEIM	R/W	0	<p>GPTM CaptureA Event Interrupt Mask</p> <p>The CAEIM values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Interrupt is disabled.</td> </tr> <tr> <td>1</td> <td>Interrupt is enabled.</td> </tr> </tbody> </table>	Value	Description	0	Interrupt is disabled.	1	Interrupt is enabled.
Value	Description									
0	Interrupt is disabled.									
1	Interrupt is enabled.									
1	CAMIM	R/W	0	<p>GPTM CaptureA Match Interrupt Mask</p> <p>The CAMIM values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Interrupt is disabled.</td> </tr> <tr> <td>1</td> <td>Interrupt is enabled.</td> </tr> </tbody> </table>	Value	Description	0	Interrupt is disabled.	1	Interrupt is enabled.
Value	Description									
0	Interrupt is disabled.									
1	Interrupt is enabled.									
0	TATOIM	R/W	0	<p>GPTM TimerA Time-Out Interrupt Mask</p> <p>The TATOIM values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Interrupt is disabled.</td> </tr> <tr> <td>1</td> <td>Interrupt is enabled.</td> </tr> </tbody> </table>	Value	Description	0	Interrupt is disabled.	1	Interrupt is enabled.
Value	Description									
0	Interrupt is disabled.									
1	Interrupt is enabled.									

### Register 6: GPTM Raw Interrupt Status (GPTMRIS), offset 0x01C

This register shows the state of the GPTM's internal interrupt signal. These bits are set whether or not the interrupt is masked in the **GPTMIMR** register. Each bit can be cleared by writing a 1 to its corresponding bit in **GPTMICR**.

#### GPTM Raw Interrupt Status (GPTMRIS)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x01C  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	CBERIS	RO	0	GPTM CaptureB Event Raw Interrupt This is the CaptureB Event interrupt status prior to masking.
9	CBMRIS	RO	0	GPTM CaptureB Match Raw Interrupt This is the CaptureB Match interrupt status prior to masking.
8	TBTORIS	RO	0	GPTM TimerB Time-Out Raw Interrupt This is the TimerB time-out interrupt status prior to masking.
7:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	RTCRIS	RO	0	GPTM RTC Raw Interrupt This is the RTC Event interrupt status prior to masking.
2	CAERIS	RO	0	GPTM CaptureA Event Raw Interrupt This is the CaptureA Event interrupt status prior to masking.
1	CAMRIS	RO	0	GPTM CaptureA Match Raw Interrupt This is the CaptureA Match interrupt status prior to masking.
0	TATORIS	RO	0	GPTM TimerA Time-Out Raw Interrupt This the TimerA time-out interrupt status prior to masking.



## Register 7: GPTM Masked Interrupt Status (GPTMMIS), offset 0x020

This register show the state of the GPTM's controller-level interrupt. If an interrupt is unmasked in **GPTMIMR**, and there is an event that causes the interrupt to be asserted, the corresponding bit is set in this register. All bits are cleared by writing a 1 to the corresponding bit in **GPTMICR**.

### GPTM Masked Interrupt Status (GPTMMIS)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x020  
 Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved				CBEMIS	CBMMIS	TBTOMIS	reserved						RTCMIS	CAEMIS	CAMMIS	TATOMIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

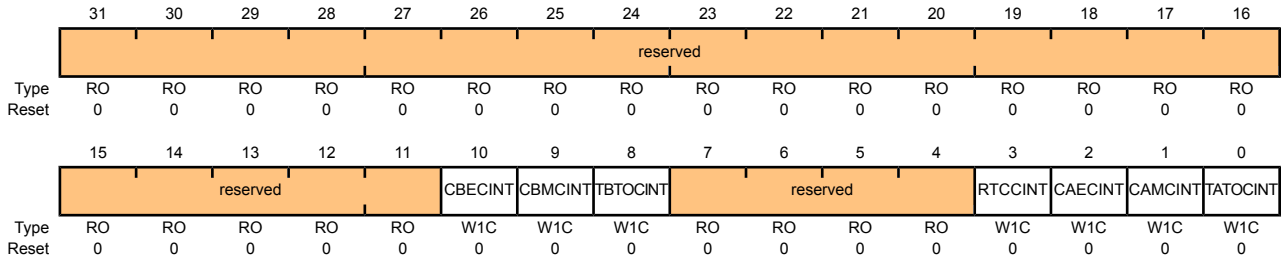
Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	CBEMIS	RO	0	GPTM CaptureB Event Masked Interrupt This is the CaptureB event interrupt status after masking.
9	CBMMIS	RO	0	GPTM CaptureB Match Masked Interrupt This is the CaptureB match interrupt status after masking.
8	TBTOMIS	RO	0	GPTM TimerB Time-Out Masked Interrupt This is the TimerB time-out interrupt status after masking.
7:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	RTCMIS	RO	0	GPTM RTC Masked Interrupt This is the RTC event interrupt status after masking.
2	CAEMIS	RO	0	GPTM CaptureA Event Masked Interrupt This is the CaptureA event interrupt status after masking.
1	CAMMIS	RO	0	GPTM CaptureA Match Masked Interrupt This is the CaptureA match interrupt status after masking.
0	TATOMIS	RO	0	GPTM TimerA Time-Out Masked Interrupt This is the TimerA time-out interrupt status after masking.

### Register 8: GPTM Interrupt Clear (GPTMICR), offset 0x024

This register is used to clear the status bits in the **GPTMRIS** and **GPTMMIS** registers. Writing a 1 to a bit clears the corresponding bit in the **GPTMRIS** and **GPTMMIS** registers.

#### GPTM Interrupt Clear (GPTMICR)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x024  
 Type W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	CBECINT	W1C	0	GPTM CaptureB Event Interrupt Clear  The <b>CBECINT</b> values are defined as follows:  Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
9	CBMCINT	W1C	0	GPTM CaptureB Match Interrupt Clear  The <b>CBMCINT</b> values are defined as follows:  Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
8	TBTOCINT	W1C	0	GPTM TimerB Time-Out Interrupt Clear  The <b>TBTOCINT</b> values are defined as follows:  Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
7:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description						
3	RTCCINT	W1C	0	<p>GPTM RTC Interrupt Clear</p> <p>The <code>RTCCINT</code> values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The interrupt is unaffected.</td> </tr> <tr> <td>1</td> <td>The interrupt is cleared.</td> </tr> </tbody> </table>	Value	Description	0	The interrupt is unaffected.	1	The interrupt is cleared.
Value	Description									
0	The interrupt is unaffected.									
1	The interrupt is cleared.									
2	CAECINT	W1C	0	<p>GPTM CaptureA Event Interrupt Clear</p> <p>The <code>CAECINT</code> values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The interrupt is unaffected.</td> </tr> <tr> <td>1</td> <td>The interrupt is cleared.</td> </tr> </tbody> </table>	Value	Description	0	The interrupt is unaffected.	1	The interrupt is cleared.
Value	Description									
0	The interrupt is unaffected.									
1	The interrupt is cleared.									
1	CAMCINT	W1C	0	<p>GPTM CaptureA Match Raw Interrupt</p> <p>This is the CaptureA match interrupt status after masking.</p>						
0	TATOCINT	W1C	0	<p>GPTM TimerA Time-Out Raw Interrupt</p> <p>The <code>TATOCINT</code> values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The interrupt is unaffected.</td> </tr> <tr> <td>1</td> <td>The interrupt is cleared.</td> </tr> </tbody> </table>	Value	Description	0	The interrupt is unaffected.	1	The interrupt is cleared.
Value	Description									
0	The interrupt is unaffected.									
1	The interrupt is cleared.									

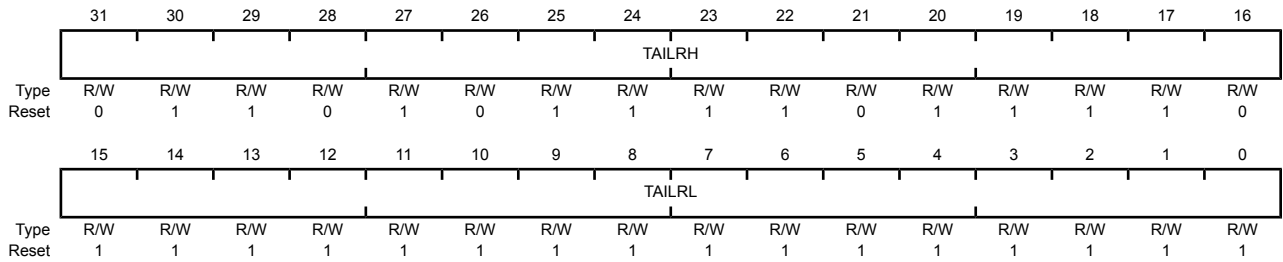
### Register 9: GPTM TimerA Interval Load (GPTMTAILR), offset 0x028

This register is used to load the starting count value into the timer. When GPTM is configured to one of the 32-bit modes, **GPTMTAILR** appears as a 32-bit register (the upper 16-bits correspond to the contents of the **GPTM TimerB Interval Load (GPTMTBILR)** register). In 16-bit mode, the upper 16 bits of this register read as 0s and have no effect on the state of **GPTMTBILR**.

#### GPTM TimerA Interval Load (GPTMTAILR)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x028

Type R/W, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)



Bit/Field	Name	Type	Reset	Description
31:16	TAILRH	R/W	0xFFFF (32-bit mode) 0x0000 (16-bit mode)	GPTM TimerA Interval Load Register High  When configured for 32-bit mode via the <b>GPTMCFG</b> register, the <b>GPTM TimerB Interval Load (GPTMTBILR)</b> register loads this value on a write. A read returns the current value of <b>GPTMTBILR</b> .  In 16-bit mode, this field reads as 0 and does not have an effect on the state of <b>GPTMTBILR</b> .
15:0	TAILRL	R/W	0xFFFF	GPTM TimerA Interval Load Register Low  For both 16- and 32-bit modes, writing this field loads the counter for TimerA. A read returns the current value of <b>GPTMTAILR</b> .

## Register 10: GPTM TimerB Interval Load (GPTMTBILR), offset 0x02C

This register is used to load the starting count value into TimerB. When the GPTM is configured to a 32-bit mode, **GPTMTBILR** returns the current value of TimerB and ignores writes.

### GPTM TimerB Interval Load (GPTMTBILR)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x02C  
 Type R/W, reset 0x0000.FFFF

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TBILRL	R/W	0xFFFF	GPTM TimerB Interval Load Register

When the GPTM is not configured as a 32-bit timer, a write to this field updates **GPTMTBILR**. In 32-bit mode, writes are ignored, and reads return the current value of **GPTMTBILR**.

### Register 11: GPTM TimerA Match (GPTMTAMATCHR), offset 0x030

This register is used in 32-bit Real-Time Clock mode and 16-bit PWM and Input Edge Count modes.

#### GPTM TimerA Match (GPTMTAMATCHR)

Timer0 base: 0x4003.0000

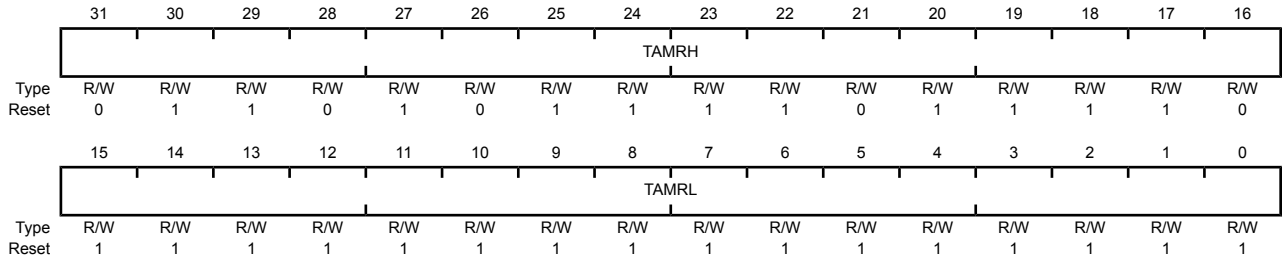
Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Timer3 base: 0x4003.3000

Offset 0x030

Type R/W, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)



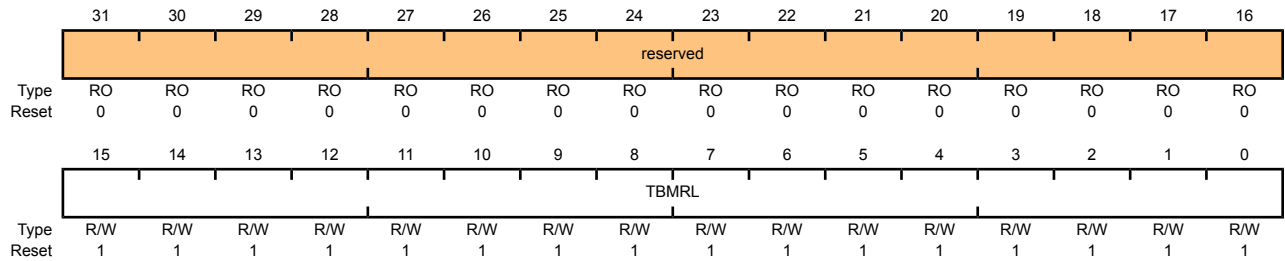
Bit/Field	Name	Type	Reset	Description
31:16	TAMRH	R/W	0xFFFF (32-bit mode) 0x0000 (16-bit mode)	GPTM TimerA Match Register High  When configured for 32-bit Real-Time Clock (RTC) mode via the <b>GPTMCFG</b> register, this value is compared to the upper half of <b>GPTMTAR</b> , to determine match events.  In 16-bit mode, this field reads as 0 and does not have an effect on the state of <b>GPTMTBMATCHR</b> .
15:0	TAMRL	R/W	0xFFFF	GPTM TimerA Match Register Low  When configured for 32-bit Real-Time Clock (RTC) mode via the <b>GPTMCFG</b> register, this value is compared to the lower half of <b>GPTMTAR</b> , to determine match events.  When configured for PWM mode, this value along with <b>GPTMTAILR</b> , determines the duty cycle of the output PWM signal.  When configured for Edge Count mode, this value along with <b>GPTMTAILR</b> , determines how many edge events are counted. The total number of edge events counted is equal to the value in <b>GPTMTAILR</b> minus this value.

## Register 12: GPTM TimerB Match (GPTMTBMATCHR), offset 0x034

This register is used in 16-bit PWM and Input Edge Count modes.

### GPTM TimerB Match (GPTMTBMATCHR)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x034  
 Type R/W, reset 0x0000.FFFF



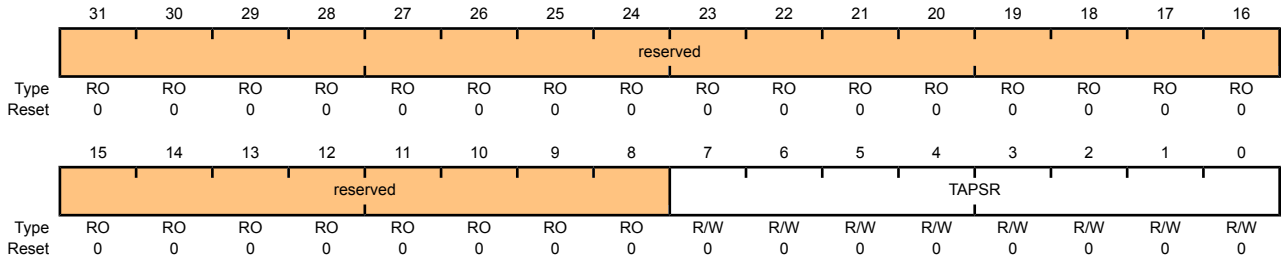
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TBMRL	R/W	0xFFFF	GPTM TimerB Match Register Low  When configured for PWM mode, this value along with <b>GPTMTBILR</b> , determines the duty cycle of the output PWM signal.  When configured for Edge Count mode, this value along with <b>GPTMTBILR</b> , determines how many edge events are counted. The total number of edge events counted is equal to the value in <b>GPTMTBILR</b> minus this value.

### Register 13: GPTM TimerA Prescale (GPTMTAPR), offset 0x038

This register allows software to extend the range of the 16-bit timers when operating in one-shot or periodic mode.

#### GPTM TimerA Prescale (GPTMTAPR)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x038  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TAPSR	R/W	0x00	GPTM TimerA Prescale  The register loads this value on a write. A read returns the current value of the register.  Refer to Table 11-2 on page 286 for more details and an example.



**Register 14: GPTM TimerB Prescale (GPTMTBPR), offset 0x03C**

This register allows software to extend the range of the 16-bit timers when operating in one-shot or periodic mode.

**GPTM TimerB Prescale (GPTMTBPR)**

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x03C  
 Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TBPSR	R/W	0x00	GPTM TimerB Prescale  The register loads this value on a write. A read returns the current value of this register.  Refer to Table 11-2 on page 286 for more details and an example.

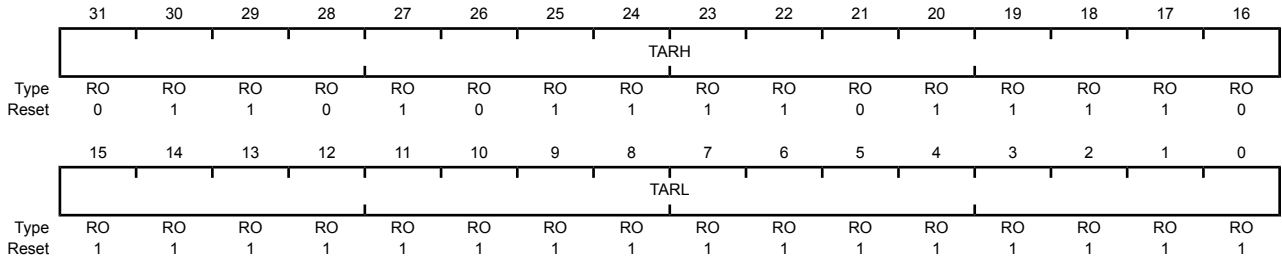
### Register 15: GPTM TimerA (GPTMTAR), offset 0x048

This register shows the current value of the TimerA counter in all cases except for Input Edge Count mode. When in this mode, this register contains the time at which the last edge event took place.

#### GPTM TimerA (GPTMTAR)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x048

Type RO, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)



Bit/Field	Name	Type	Reset	Description
31:16	TARH	RO	0xFFFF (32-bit mode) 0x0000 (16-bit mode)	GPTM TimerA Register High If the <b>GPTMCFG</b> is in a 32-bit mode, TimerB value is read. If the <b>GPTMCFG</b> is in a 16-bit mode, this is read as zero.
15:0	TARL	RO	0xFFFF	GPTM TimerA Register Low A read returns the current value of the <b>GPTM TimerA Count Register</b> , except in Input Edge Count mode, when it returns the timestamp from the last edge event.

## Register 16: GPTM TimerB (GPTMTBR), offset 0x04C

This register shows the current value of the TimerB counter in all cases except for Input Edge Count mode. When in this mode, this register contains the time at which the last edge event took place.

### GPTM TimerB (GPTMTBR)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x04C  
 Type RO, reset 0x0000.FFFF

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TBRL	RO	0xFFFF	GPTM TimerB

A read returns the current value of the **GPTM TimerB Count Register**, except in Input Edge Count mode, when it returns the timestamp from the last edge event.

## 12 Watchdog Timer

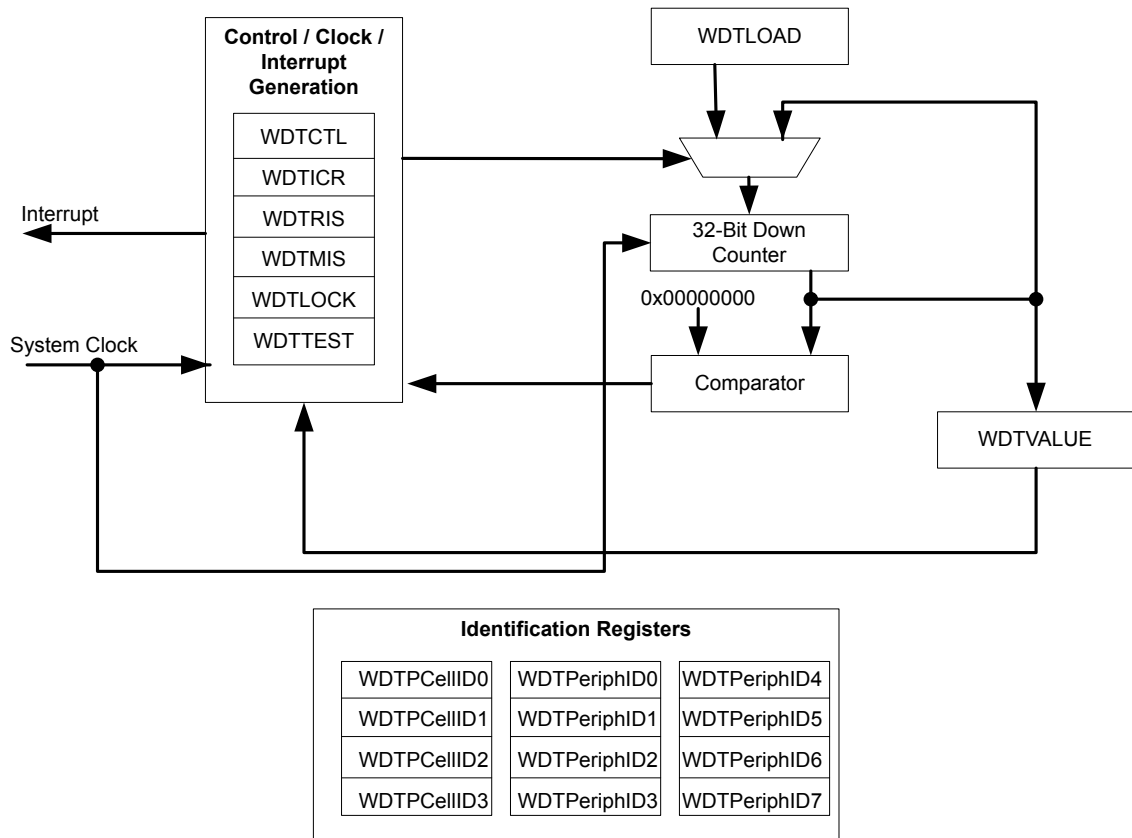
A watchdog timer can generate nonmaskable interrupts (NMIs) or a reset when a time-out value is reached. The watchdog timer is used to regain control when a system has failed due to a software error or due to the failure of an external device to respond in the expected way.

The Stellaris® Watchdog Timer module consists of a 32-bit down counter, a programmable load register, interrupt generation logic, a locking register, and user-enabled stalling.

The Watchdog Timer can be configured to generate an interrupt to the controller on its first time-out, and to generate a reset signal on its second time-out. Once the Watchdog Timer has been configured, the lock register can be written to prevent the timer configuration from being inadvertently altered.

### 12.1 Block Diagram

Figure 12-1. WDT Module Block Diagram



### 12.2 Functional Description

The Watchdog Timer module generates the first time-out signal when the 32-bit counter reaches the zero state after being enabled; enabling the counter also enables the watchdog timer interrupt. After the first time-out event, the 32-bit counter is re-loaded with the value of the **Watchdog Timer Load (WDTLOAD)** register, and the timer resumes counting down from that value. Once the

Watchdog Timer has been configured, the **Watchdog Timer Lock (WDTLOCK)** register is written, which prevents the timer configuration from being inadvertently altered by software.

If the timer counts down to its zero state again before the first time-out interrupt is cleared, and the reset signal has been enabled (via the `WatchdogResetEnable` function), the Watchdog timer asserts its reset signal to the system. If the interrupt is cleared before the 32-bit counter reaches its second time-out, the 32-bit counter is loaded with the value in the **WDTLOAD** register, and counting resumes from that value.

If **WDTLOAD** is written with a new value while the Watchdog Timer counter is counting, then the counter is loaded with the new value and continues counting.

Writing to **WDTLOAD** does not clear an active interrupt. An interrupt must be specifically cleared by writing to the **Watchdog Interrupt Clear (WDTICR)** register.

The Watchdog module interrupt and reset generation can be enabled or disabled as required. When the interrupt is re-enabled, the 32-bit counter is preloaded with the load register value and not its last state.

## 12.3 Initialization and Configuration

To use the WDT, its peripheral clock must be enabled by setting the `WDT` bit in the **RCGC0** register. The Watchdog Timer is configured using the following sequence:

1. Load the **WDTLOAD** register with the desired timer load value.
2. If the Watchdog is configured to trigger system resets, set the `RESEN` bit in the **WDTCTL** register.
3. Set the `INTEN` bit in the **WDTCTL** register to enable the Watchdog and lock the control register.

If software requires that all of the watchdog registers are locked, the Watchdog Timer module can be fully locked by writing any value to the **WDTLOCK** register. To unlock the Watchdog Timer, write a value of `0x1ACC.E551`.

## 12.4 Register Map

Table 12-1 on page 317 lists the Watchdog registers. The offset listed is a hexadecimal increment to the register's address, relative to the Watchdog Timer base address of `0x4000.0000`.

**Table 12-1. Watchdog Timer Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	WDTLOAD	R/W	0xFFFF.FFFF	Watchdog Load	319
0x004	WDTVALUE	RO	0xFFFF.FFFF	Watchdog Value	320
0x008	WDTCTL	R/W	0x0000.0000	Watchdog Control	321
0x00C	WDTICR	WO	-	Watchdog Interrupt Clear	322
0x010	WDTRIS	RO	0x0000.0000	Watchdog Raw Interrupt Status	323
0x014	WDTMIS	RO	0x0000.0000	Watchdog Masked Interrupt Status	324
0x418	WDTTEST	R/W	0x0000.0000	Watchdog Test	325
0xC00	WDTLOCK	R/W	0x0000.0000	Watchdog Lock	326

Offset	Name	Type	Reset	Description	See page
0xFD0	WDTPeriphID4	RO	0x0000.0000	Watchdog Peripheral Identification 4	327
0xFD4	WDTPeriphID5	RO	0x0000.0000	Watchdog Peripheral Identification 5	328
0xFD8	WDTPeriphID6	RO	0x0000.0000	Watchdog Peripheral Identification 6	329
0xFDC	WDTPeriphID7	RO	0x0000.0000	Watchdog Peripheral Identification 7	330
0xFE0	WDTPeriphID0	RO	0x0000.0005	Watchdog Peripheral Identification 0	331
0xFE4	WDTPeriphID1	RO	0x0000.0018	Watchdog Peripheral Identification 1	332
0xFE8	WDTPeriphID2	RO	0x0000.0018	Watchdog Peripheral Identification 2	333
0xFEC	WDTPeriphID3	RO	0x0000.0001	Watchdog Peripheral Identification 3	334
0xFF0	WDTPCellID0	RO	0x0000.000D	Watchdog PrimeCell Identification 0	335
0xFF4	WDTPCellID1	RO	0x0000.00F0	Watchdog PrimeCell Identification 1	336
0xFF8	WDTPCellID2	RO	0x0000.0005	Watchdog PrimeCell Identification 2	337
0xFFC	WDTPCellID3	RO	0x0000.00B1	Watchdog PrimeCell Identification 3	338

## 12.5 Register Descriptions

The remainder of this section lists and describes the WDT registers, in numerical order by address offset.

## Register 1: Watchdog Load (WDTLOAD), offset 0x000

This register is the 32-bit interval value used by the 32-bit counter. When this register is written, the value is immediately loaded and the counter restarts counting down from the new value. If the **WDTLOAD** register is loaded with 0x0000.0000, an interrupt is immediately generated.

### Watchdog Load (WDTLOAD)

Base 0x4000.0000

Offset 0x000

Type R/W, reset 0xFFFF.FFFF

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

Bit/Field	Name	Type	Reset	Description
31:0	WDTLoad	R/W	0xFFFF.FFFF	Watchdog Load Value

## Register 2: Watchdog Value (WDTVALUE), offset 0x004

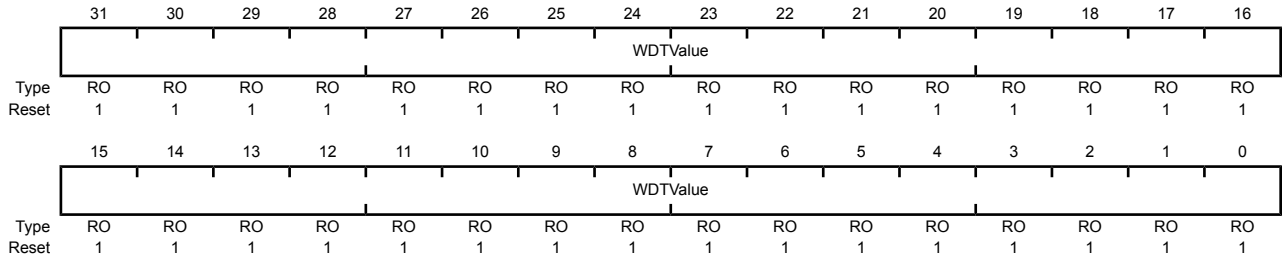
This register contains the current count value of the timer.

### Watchdog Value (WDTVALUE)

Base 0x4000.0000

Offset 0x004

Type RO, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	WDTValue	RO	0xFFFF.FFFF	Watchdog Value Current value of the 32-bit down counter.



### Register 3: Watchdog Control (WDTCTL), offset 0x008

This register is the watchdog control register. The watchdog timer can be configured to generate a reset signal (on second time-out) or an interrupt on time-out.

When the watchdog interrupt has been enabled, all subsequent writes to the control register are ignored. The only mechanism that can re-enable writes is a hardware reset.

#### Watchdog Control (WDTCTL)

Base 0x4000.0000  
Offset 0x008  
Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description						
31:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
1	RESEN	R/W	0	Watchdog Reset Enable  The RESEN values are defined as follows:  <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disabled.</td> </tr> <tr> <td>1</td> <td>Enable the Watchdog module reset output.</td> </tr> </tbody> </table>	Value	Description	0	Disabled.	1	Enable the Watchdog module reset output.
Value	Description									
0	Disabled.									
1	Enable the Watchdog module reset output.									
0	INTEN	R/W	0	Watchdog Interrupt Enable  The INTEN values are defined as follows:  <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Interrupt event disabled (once this bit is set, it can only be cleared by a hardware reset).</td> </tr> <tr> <td>1</td> <td>Interrupt event enabled. Once enabled, all writes are ignored.</td> </tr> </tbody> </table>	Value	Description	0	Interrupt event disabled (once this bit is set, it can only be cleared by a hardware reset).	1	Interrupt event enabled. Once enabled, all writes are ignored.
Value	Description									
0	Interrupt event disabled (once this bit is set, it can only be cleared by a hardware reset).									
1	Interrupt event enabled. Once enabled, all writes are ignored.									

### Register 4: Watchdog Interrupt Clear (WDTICR), offset 0x00C

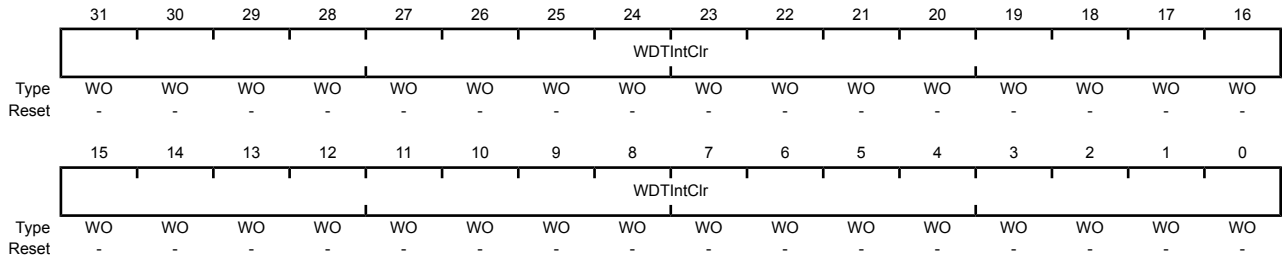
This register is the interrupt clear register. A write of any value to this register clears the Watchdog interrupt and reloads the 32-bit counter from the **WDTLOAD** register. Value for a read or reset is indeterminate.

#### Watchdog Interrupt Clear (WDTICR)

Base 0x4000.0000

Offset 0x00C

Type WO, reset -



Bit/Field	Name	Type	Reset	Description
31:0	WDTIntClr	WO	-	Watchdog Interrupt Clear

## Register 5: Watchdog Raw Interrupt Status (WDTRIS), offset 0x010

This register is the raw interrupt status register. Watchdog interrupt events can be monitored via this register if the controller interrupt is masked.

### Watchdog Raw Interrupt Status (WDTRIS)

Base 0x4000.0000

Offset 0x010

Type RO, reset 0x0000.0000

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

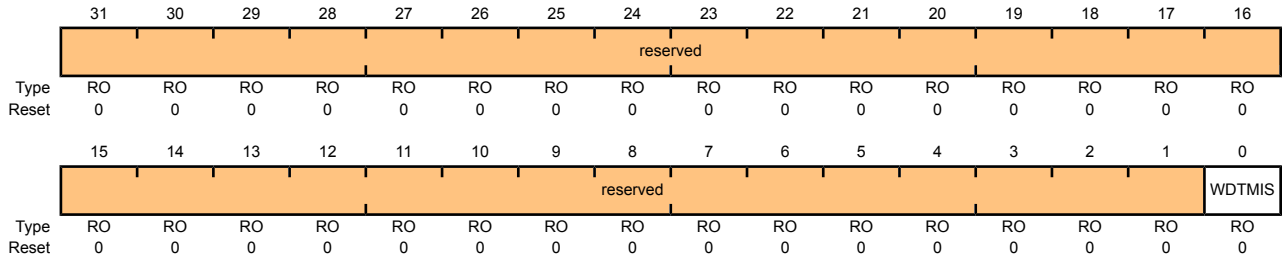
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	WDTRIS	RO	0	Watchdog Raw Interrupt Status Gives the raw interrupt state (prior to masking) of <b>WDTINTR</b> .

### Register 6: Watchdog Masked Interrupt Status (WDTMIS), offset 0x014

This register is the masked interrupt status register. The value of this register is the logical AND of the raw interrupt bit and the Watchdog interrupt enable bit.

#### Watchdog Masked Interrupt Status (WDTMIS)

Base 0x4000.0000  
 Offset 0x014  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	WDTMIS	RO	0	Watchdog Masked Interrupt Status  Gives the masked interrupt state (after masking) of the <b>WDTINTR</b> interrupt.

**Register 7: Watchdog Test (WDTTEST), offset 0x418**

This register provides user-enabled stalling when the microcontroller asserts the CPU halt flag during debug.

**Watchdog Test (WDTTEST)**

Base 0x4000.0000  
Offset 0x418  
Type R/W, reset 0x0000.0000

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

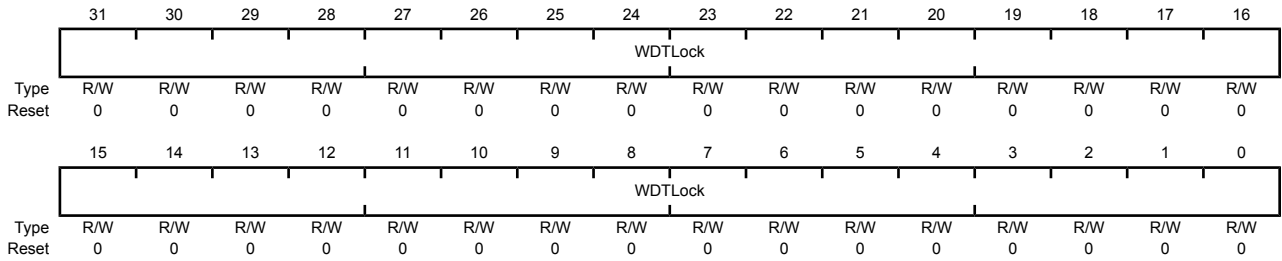
Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	STALL	R/W	0	<p>Watchdog Stall Enable</p> <p>When set to 1, if the Stellaris<sup>®</sup> microcontroller is stopped with a debugger, the watchdog timer stops counting. Once the microcontroller is restarted, the watchdog timer resumes counting.</p>
7:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

### Register 8: Watchdog Lock (WDTLOCK), offset 0xC00

Writing 0x1ACC.E551 to the **WDTLOCK** register enables write access to all other registers. Writing any other value to the **WDTLOCK** register re-enables the locked state for register writes to all the other registers. Reading the **WDTLOCK** register returns the lock status rather than the 32-bit value written. Therefore, when write accesses are disabled, reading the **WDTLOCK** register returns 0x0000.0001 (when locked; otherwise, the returned value is 0x0000.0000 (unlocked)).

#### Watchdog Lock (WDTLOCK)

Base 0x4000.0000  
 Offset 0xC00  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:0	WDTLock	R/W	0x0000	Watchdog Lock  A write of the value 0x1ACC.E551 unlocks the watchdog registers for write access. A write of any other value reapplies the lock, preventing any register updates.  A read of this register returns the following values: <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0000.0001</td> <td>Locked</td> </tr> <tr> <td>0x0000.0000</td> <td>Unlocked</td> </tr> </tbody> </table>	Value	Description	0x0000.0001	Locked	0x0000.0000	Unlocked
Value	Description									
0x0000.0001	Locked									
0x0000.0000	Unlocked									

## Register 9: Watchdog Peripheral Identification 4 (WDTPeriphID4), offset 0xFD0

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### Watchdog Peripheral Identification 4 (WDTPeriphID4)

Base 0x4000.0000

Offset 0xFD0

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

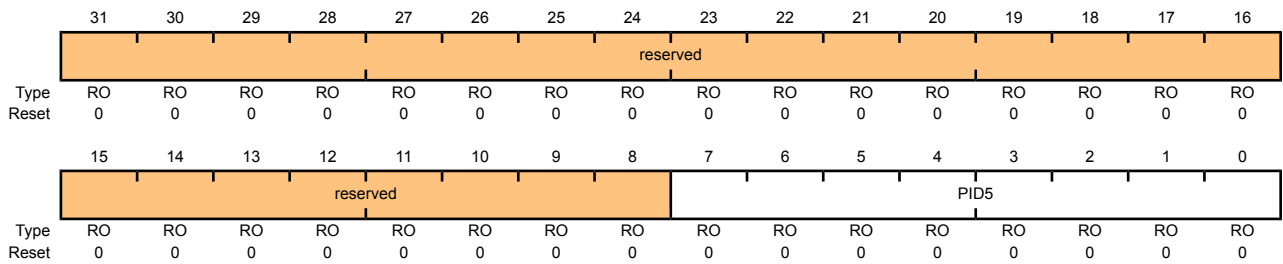
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	WDT Peripheral ID Register[7:0]

## Register 10: Watchdog Peripheral Identification 5 (WDTPeriphID5), offset 0xFD4

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### Watchdog Peripheral Identification 5 (WDTPeriphID5)

Base 0x4000.0000  
 Offset 0xFD4  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	WDT Peripheral ID Register[15:8]



## Register 11: Watchdog Peripheral Identification 6 (WDTPeriphID6), offset 0xFD8

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### Watchdog Peripheral Identification 6 (WDTPeriphID6)

Base 0x4000.0000

Offset 0xFD8

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID6							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

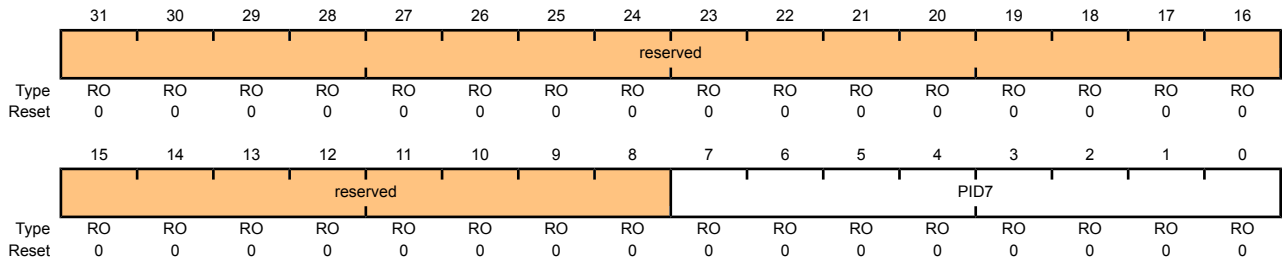
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	WDT Peripheral ID Register[23:16]

## Register 12: Watchdog Peripheral Identification 7 (WDTPeriphID7), offset 0xFDC

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### Watchdog Peripheral Identification 7 (WDTPeriphID7)

Base 0x4000.0000  
 Offset 0xFDC  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	WDT Peripheral ID Register[31:24]

## Register 13: Watchdog Peripheral Identification 0 (WDTPeriphID0), offset 0xFE0

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### Watchdog Peripheral Identification 0 (WDTPeriphID0)

Base 0x4000.0000

Offset 0xFE0

Type RO, reset 0x0000.0005

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

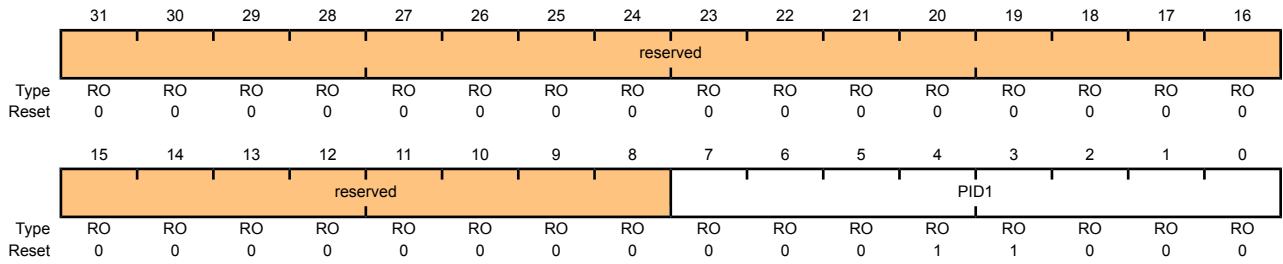
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x05	Watchdog Peripheral ID Register[7:0]

## Register 14: Watchdog Peripheral Identification 1 (WDTPeriphID1), offset 0xFE4

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### Watchdog Peripheral Identification 1 (WDTPeriphID1)

Base 0x4000.0000  
 Offset 0xFE4  
 Type RO, reset 0x0000.0018



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x18	Watchdog Peripheral ID Register[15:8]

## Register 15: Watchdog Peripheral Identification 2 (WDTPeriphID2), offset 0xFE8

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### Watchdog Peripheral Identification 2 (WDTPeriphID2)

Base 0x4000.0000

Offset 0xFE8

Type RO, reset 0x0000.0018

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	Watchdog Peripheral ID Register[23:16]

## Register 16: Watchdog Peripheral Identification 3 (WDTPeriphID3), offset 0xFEC

The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### Watchdog Peripheral Identification 3 (WDTPeriphID3)

Base 0x4000.0000

Offset 0xFEC

Type RO, reset 0x0000.0001

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID3							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	Watchdog Peripheral ID Register[31:24]

**Register 17: Watchdog PrimeCell Identification 0 (WDTPCellID0), offset 0xFF0**

The **WDTPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

## Watchdog PrimeCell Identification 0 (WDTPCellID0)

Base 0x4000.0000

Offset 0xFF0

Type RO, reset 0x0000.000D

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

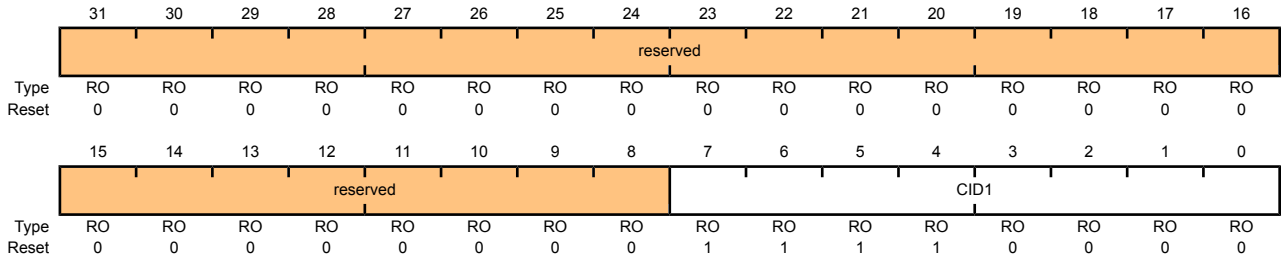
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	Watchdog PrimeCell ID Register[7:0]

### Register 18: Watchdog PrimeCell Identification 1 (WDTPCellID1), offset 0xFF4

The **WDTPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

#### Watchdog PrimeCell Identification 1 (WDTPCellID1)

Base 0x4000.0000  
 Offset 0xFF4  
 Type RO, reset 0x0000.00F0



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	Watchdog PrimeCell ID Register[15:8]



**Register 19: Watchdog PrimeCell Identification 2 (WDTPCellID2), offset 0xFF8**

The **WDTPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

## Watchdog PrimeCell Identification 2 (WDTPCellID2)

Base 0x4000.0000

Offset 0xFF8

Type RO, reset 0x0000.0005

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

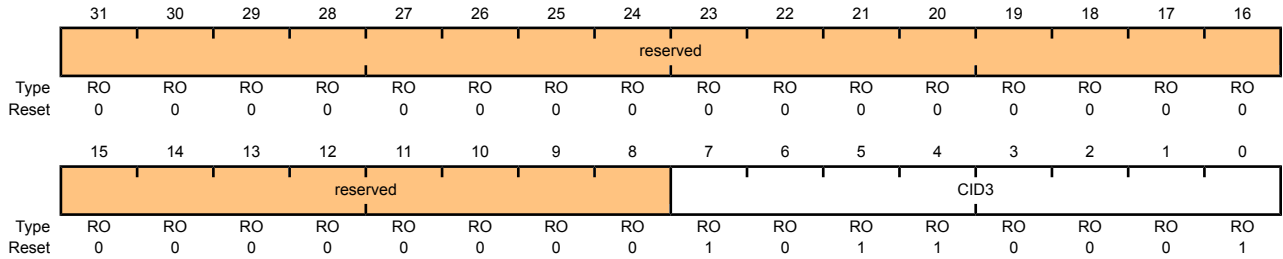
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	Watchdog PrimeCell ID Register[23:16]

### Register 20: Watchdog PrimeCell Identification 3 (WDTPCellID3), offset 0xFFC

The **WDTPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

#### Watchdog PrimeCell Identification 3 (WDTPCellID3)

Base 0x4000.0000  
 Offset 0xFFC  
 Type RO, reset 0x0000.00B1



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	Watchdog PrimeCell ID Register[31:24]

## 13 Analog-to-Digital Converter (ADC)

An analog-to-digital converter (ADC) is a peripheral that converts a continuous analog voltage to a discrete digital number.

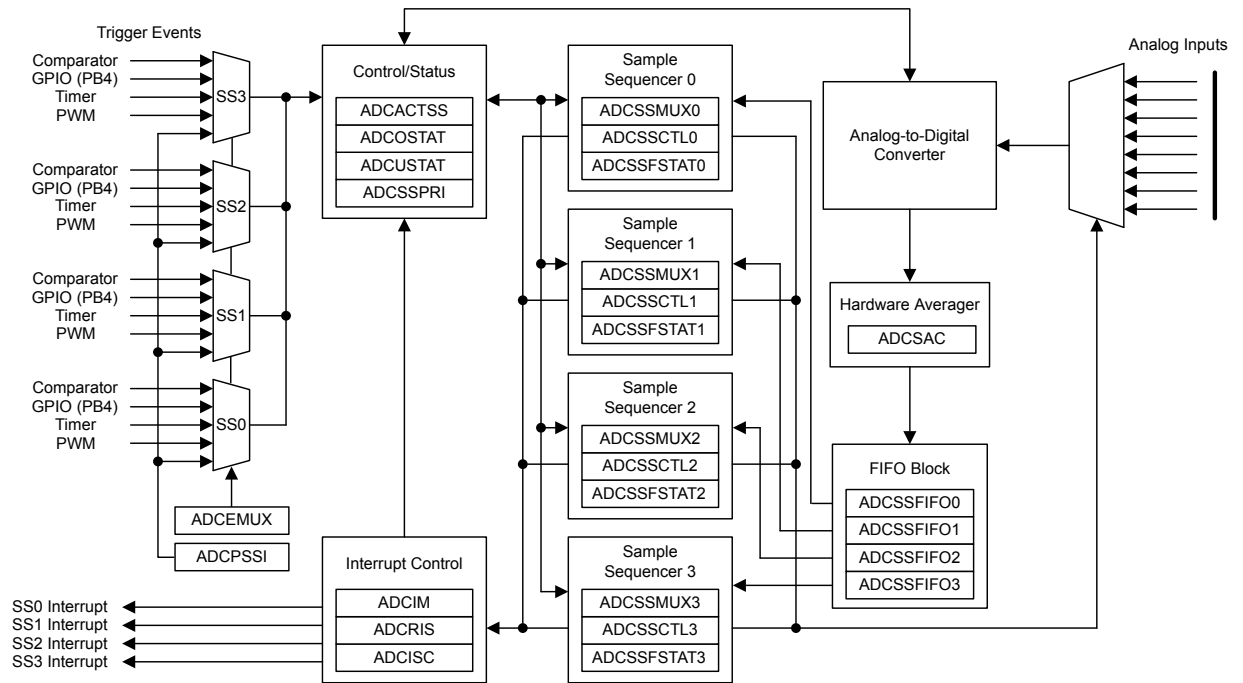
The Stellaris<sup>®</sup> ADC module features 10-bit conversion resolution and supports eight input channels, plus an internal temperature sensor. The ADC module contains a programmable sequencer which allows for the sampling of multiple analog input sources without controller intervention. Each sample sequence provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequence priority.

The Stellaris<sup>®</sup> ADC provides the following features:

- Eight analog input channels
- Single-ended and differential-input configurations
- Internal temperature sensor
- Sample rate of 500 thousand samples/second
- Four programmable sample conversion sequences from one to eight entries long, with corresponding conversion result FIFOs
- Flexible trigger control
  - Controller (software)
  - Timers
  - GPIO
- Hardware averaging of up to 64 samples for improved accuracy
- An internal 3-V reference is used by the converter.
- Power and ground for the analog circuitry is separate from the digital power and ground.

## 13.1 Block Diagram

Figure 13-1. ADC Module Block Diagram



## 13.2 Functional Description

The Stellaris<sup>®</sup> ADC collects sample data by using a programmable sequence-based approach instead of the traditional single or double-sampling approach found on many ADC modules. Each *sample sequence* is a fully programmed series of consecutive (back-to-back) samples, allowing the ADC to collect data from multiple input sources without having to be re-configured or serviced by the controller. The programming of each sample in the sample sequence includes parameters such as the input source and mode (differential versus single-ended input), interrupt generation on sample completion, and the indicator for the last sample in the sequence.

### 13.2.1 Sample Sequencers

The sampling control and data capture is handled by the Sample Sequencers. All of the sequencers are identical in implementation except for the number of samples that can be captured and the depth of the FIFO. Table 13-1 on page 340 shows the maximum number of samples that each Sequencer can capture and its corresponding FIFO depth. In this implementation, each FIFO entry is a 32-bit word, with the lower 10 bits containing the conversion result.

Table 13-1. Samples and FIFO Depth of Sequencers

Sequencer	Number of Samples	Depth of FIFO
SS3	1	1
SS2	4	4
SS1	4	4
SS0	8	8

For a given sample sequence, each sample is defined by two 4-bit nibbles in the **ADC Sample Sequence Input Multiplexer Select (ADCSSMUXn)** and **ADC Sample Sequence Control (ADCSSCTLn)** registers, where "n" corresponds to the sequence number. The **ADCSSMUXn** nibbles select the input pin, while the **ADCSSCTLn** nibbles contain the sample control bits corresponding to parameters such as temperature sensor selection, interrupt enable, end of sequence, and differential input mode. Sample Sequencers are enabled by setting the respective **ASENn** bit in the **ADC Active Sample Sequencer (ADCACTSS)** register, but can be configured before being enabled.

When configuring a sample sequence, multiple uses of the same input pin within the same sequence is allowed. In the **ADCSSCTLn** register, the **Interrupt Enable (IE)** bits can be set for any combination of samples, allowing interrupts to be generated after every sample in the sequence if necessary. Also, the **END** bit can be set at any point within a sample sequence. For example, if Sequencer 0 is used, the **END** bit can be set in the nibble associated with the fifth sample, allowing Sequencer 0 to complete execution of the sample sequence after the fifth sample.

After a sample sequence completes execution, the result data can be retrieved from the **ADC Sample Sequence Result FIFO (ADCSSFIFO)** registers. The FIFOs are simple circular buffers that read a single address to "pop" result data. For software debug purposes, the positions of the FIFO head and tail pointers are visible in the **ADC Sample Sequence FIFO Status (ADCSSFSTATn)** registers along with **FULL** and **EMPTY** status flags. Overflow and underflow conditions are monitored using the **ADCSTAT** and **ADCUSTAT** registers.

## 13.2.2 Module Control

Outside of the Sample Sequencers, the remainder of the control logic is responsible for tasks such as interrupt generation, sequence prioritization, and trigger configuration.

Most of the ADC control logic runs at the ADC clock rate of 14-18 MHz. The internal ADC divider is configured automatically by hardware when the system **XTAL** is selected. The automatic clock divider configuration targets 16.667 MHz operation for all Stellaris<sup>®</sup> devices.

### 13.2.2.1 Interrupts

The Sample Sequencers dictate the events that cause interrupts, but they don't have control over whether the interrupt is actually sent to the interrupt controller. The ADC module's interrupt signal is controlled by the state of the **MASK** bits in the **ADC Interrupt Mask (ADCIM)** register. Interrupt status can be viewed at two locations: the **ADC Raw Interrupt Status (ADCRIS)** register, which shows the raw status of a Sample Sequencer's interrupt signal, and the **ADC Interrupt Status and Clear (ADCISC)** register, which shows the logical AND of the **ADCRIS** register's **INR** bit and the **ADCIM** register's **MASK** bits. Interrupts are cleared by writing a 1 to the corresponding **IN** bit in **ADCISC**.

### 13.2.2.2 Prioritization

When sampling events (triggers) happen concurrently, they are prioritized for processing by the values in the **ADC Sample Sequencer Priority (ADCSSPRI)** register. Valid priority values are in the range of 0-3, with 0 being the highest priority and 3 being the lowest. Multiple active Sample Sequencer units with the same priority do not provide consistent results, so software must ensure that all active Sample Sequencer units have a unique priority value.

### 13.2.2.3 Sampling Events

Sample triggering for each Sample Sequencer is defined in the **ADC Event Multiplexer Select (ADCEMUX)** register. The external peripheral triggering sources vary by Stellaris<sup>®</sup> family member,

but all devices share the "Controller" and "Always" triggers. Software can initiate sampling by setting the `CH` bits in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register.

When using the "Always" trigger, care must be taken. If a sequence's priority is too high, it is possible to starve other lower priority sequences.

### 13.2.3 Hardware Sample Averaging Circuit

Higher precision results can be generated using the hardware averaging circuit, however, the improved results are at the cost of throughput. Up to 64 samples can be accumulated and averaged to form a single data entry in the sequencer FIFO. Throughput is decreased proportionally to the number of samples in the averaging calculation. For example, if the averaging circuit is configured to average 16 samples, the throughput is decreased by a factor of 16.

By default the averaging circuit is off and all data from the converter passes through to the sequencer FIFO. The averaging hardware is controlled by the **ADC Sample Averaging Control (ADCSAC)** register (see page 358). There is a single averaging circuit and all input channels receive the same amount of averaging whether they are single-ended or differential.

### 13.2.4 Analog-to-Digital Converter

The converter itself generates a 10-bit output value for selected analog input. Special analog pads are used to minimize the distortion on the input. An internal 3 V reference is used by the converter resulting in sample values ranging from 0x000 at 0 V input to 0x3FF at 3 V input when in single-ended input mode.

### 13.2.5 Differential Sampling

In addition to traditional single-ended sampling, the ADC module supports differential sampling of two analog input channels. To enable differential sampling, software must set the `D` bit (in the **ADCSSCTL0** register) in a step's configuration nibble.

When a sequence step is configured for differential sampling, its corresponding value in the **ADCSSMUX** register must be set to one of the four differential pairs, numbered 0-3. Differential pair 0 samples analog inputs 0 and 1; differential pair 1 samples analog inputs 2 and 3; and so on (see Table 13-2 on page 342). The ADC does not support other differential pairings such as analog input 0 with analog input 3. The number of differential pairs supported is dependent on the number of analog inputs (see Table 13-2 on page 342).

**Table 13-2. Differential Sampling Pairs**

Differential Pair	Analog Inputs
0	0 and 1
1	2 and 3
2	4 and 5
3	6 and 7

The voltage sampled in differential mode is the difference between the odd and even channels:

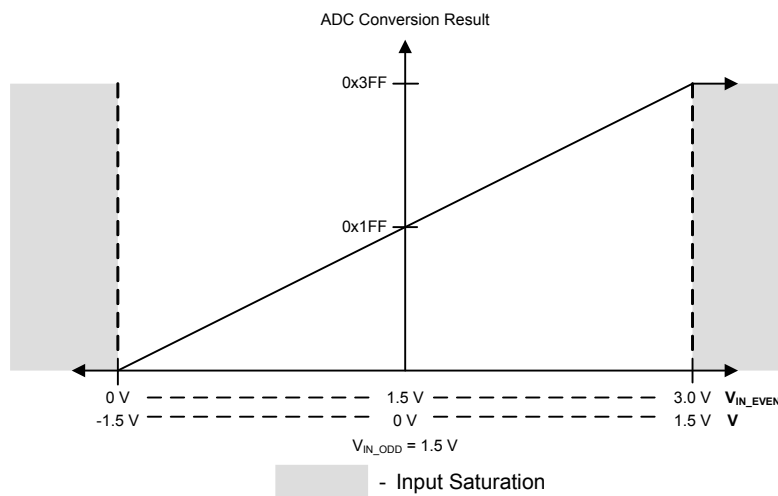
$\Delta V$  (differential voltage) =  $V_{IN\_EVEN}$  (even channels) –  $V_{IN\_ODD}$  (odd channels), therefore:

- If  $\Delta V = 0$ , then the conversion result = 0x1FF
- If  $\Delta V > 0$ , then the conversion result > 0x1FF (range is 0x1FF–0x3FF)
- If  $\Delta V < 0$ , then the conversion result < 0x1FF (range is 0–0x1FF)

The differential pairs assign polarities to the analog inputs: the even-numbered input is always positive, and the odd-numbered input is always negative. In order for a valid conversion result to appear, the negative input must be in the range of  $\pm 1.5$  V of the positive input. If an analog input is greater than 3 V or less than 0 V (the valid range for analog inputs), the input voltage is clipped, meaning it appears as either 3 V or 0 V, respectively, to the ADC.

Figure 13-2 on page 343 shows an example of the negative input centered at 1.5 V. In this configuration, the differential range spans from -1.5 V to 1.5 V. Figure 13-3 on page 343 shows an example where the negative input is centered at -0.75 V, meaning inputs on the positive input saturate past a differential voltage of -0.75 V since the input voltage is less than 0 V. Figure 13-4 on page 344 shows an example of the negative input centered at 2.25 V, where inputs on the positive channel saturate past a differential voltage of 0.75 V since the input voltage would be greater than 3 V.

**Figure 13-2. Differential Sampling Range,  $V_{IN\_ODD} = 1.5$  V**



**Figure 13-3. Differential Sampling Range,  $V_{IN\_ODD} = 0.75$  V**

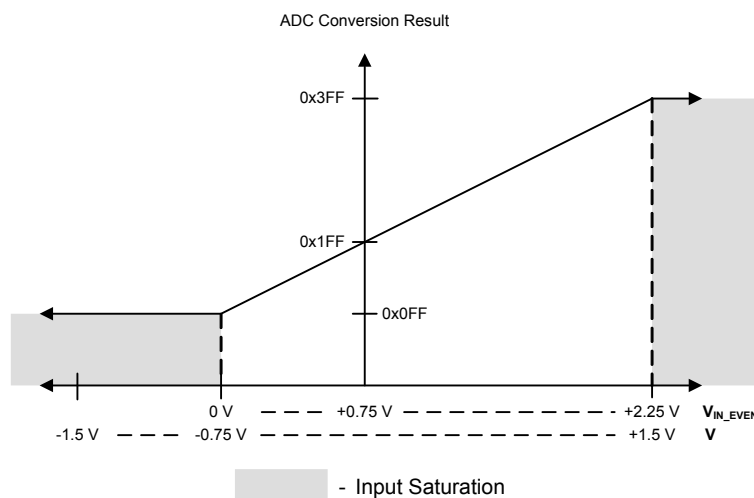
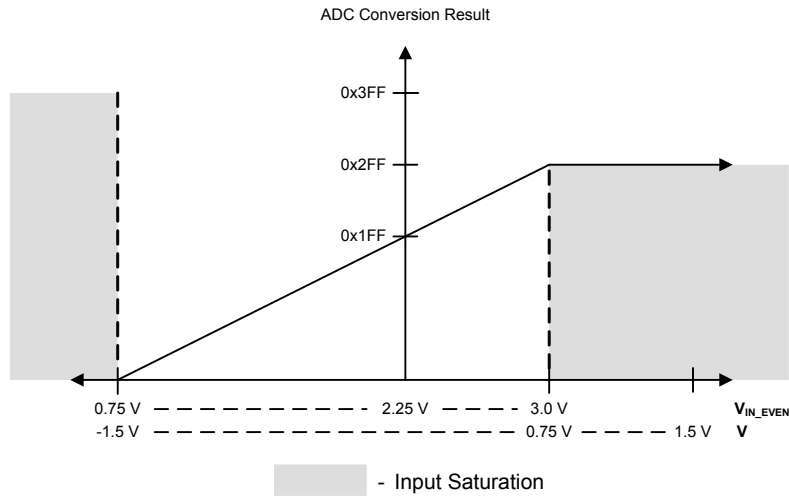


Figure 13-4. Differential Sampling Range,  $V_{IN\_ODD} = 2.25\text{ V}$



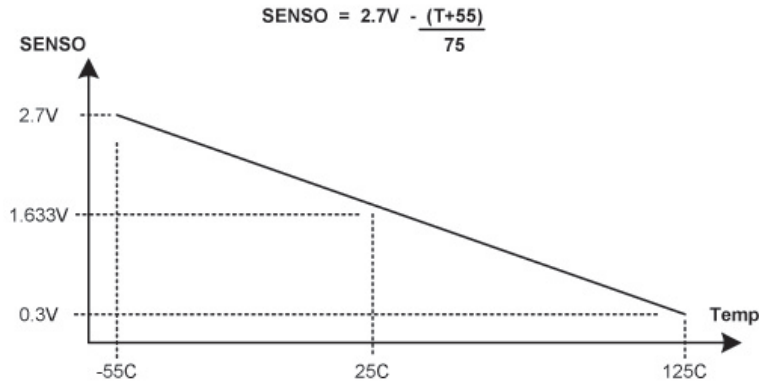
### 13.2.6 Internal Temperature Sensor

The internal temperature sensor provides an analog temperature reading as well as a reference voltage. The voltage at the output terminal SENS0 is given by the following equation:

$$SENS0 = 2.7 - ((T + 55) / 75)$$

This relation is shown in Figure 13-5 on page 344.

Figure 13-5. Internal Temperature Sensor Characteristic



### 13.3 Initialization and Configuration

In order for the ADC module to be used, the PLL must be enabled and using a supported crystal frequency (see the **RCC** register). Using unsupported frequencies can cause faulty operation in the ADC module.



### 13.3.1 Module Initialization

Initialization of the ADC module is a simple process with very few steps. The main steps include enabling the clock to the ADC, disabling the analog isolation circuit associated with all inputs that are to be used, and reconfiguring the Sample Sequencer priorities (if needed).

The initialization sequence for the ADC is as follows:

1. Enable the ADC clock by writing a value of 0x0001.0000 to the **RCGC1** register (see page 108).
2. Disable the analog isolation circuit for all ADC input pins that are to be used by writing a 1 to the appropriate bits of the **GPIOAMSEL** register (see page 269) in the associated GPIO block.
3. If required by the application, reconfigure the Sample Sequencer priorities in the **ADCSSPRI** register. The default configuration has Sample Sequencer 0 with the highest priority, and Sample Sequencer 3 as the lowest priority.

### 13.3.2 Sample Sequencer Configuration

Configuration of the Sample Sequencers is slightly more complex than the module initialization since each sample sequence is completely programmable.

The configuration for each Sample Sequencer should be as follows:

1. Ensure that the Sample Sequencer is disabled by writing a 0 to the corresponding **ASEN** bit in the **ADCACTSS** register. Programming of the Sample Sequencers is allowed without having them enabled. Disabling the Sequencer during programming prevents erroneous execution if a trigger event were to occur during the configuration process.
2. Configure the trigger event for the Sample Sequencer in the **ADCEMUX** register.
3. For each sample in the sample sequence, configure the corresponding input source in the **ADCSSMUXn** register.
4. For each sample in the sample sequence, configure the sample control bits in the corresponding nibble in the **ADCSSCTLn** register. When programming the last nibble, ensure that the **END** bit is set. Failure to set the **END** bit causes unpredictable behavior.
5. If interrupts are to be used, write a 1 to the corresponding **MASK** bit in the **ADCIM** register.
6. Enable the Sample Sequencer logic by writing a 1 to the corresponding **ASEN** bit in the **ADCACTSS** register.

## 13.4 Register Map

Table 13-3 on page 345 lists the ADC registers. The offset listed is a hexadecimal increment to the register's address, relative to the ADC base address of 0x4003.8000.

**Table 13-3. ADC Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	ADCACTSS	R/W	0x0000.0000	ADC Active Sample Sequencer	347
0x004	ADCRIS	RO	0x0000.0000	ADC Raw Interrupt Status	348

Offset	Name	Type	Reset	Description	See page
0x008	ADCIM	R/W	0x0000.0000	ADC Interrupt Mask	349
0x00C	ADCISC	R/W1C	0x0000.0000	ADC Interrupt Status and Clear	350
0x010	ADCOSTAT	R/W1C	0x0000.0000	ADC Overflow Status	351
0x014	ADCEMUX	R/W	0x0000.0000	ADC Event Multiplexer Select	352
0x018	ADCUSTAT	R/W1C	0x0000.0000	ADC Underflow Status	355
0x020	ADCSSPRI	R/W	0x0000.3210	ADC Sample Sequencer Priority	356
0x028	ADCPSSI	WO	-	ADC Processor Sample Sequence Initiate	357
0x030	ADCSAC	R/W	0x0000.0000	ADC Sample Averaging Control	358
0x040	ADCSSMUX0	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 0	359
0x044	ADCSSCTL0	R/W	0x0000.0000	ADC Sample Sequence Control 0	361
0x048	ADCSSFIFO0	RO	0x0000.0000	ADC Sample Sequence Result FIFO 0	364
0x04C	ADCSSFSTAT0	RO	0x0000.0100	ADC Sample Sequence FIFO 0 Status	365
0x060	ADCSSMUX1	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 1	366
0x064	ADCSSCTL1	R/W	0x0000.0000	ADC Sample Sequence Control 1	367
0x068	ADCSSFIFO1	RO	0x0000.0000	ADC Sample Sequence Result FIFO 1	364
0x06C	ADCSSFSTAT1	RO	0x0000.0100	ADC Sample Sequence FIFO 1 Status	365
0x080	ADCSSMUX2	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 2	366
0x084	ADCSSCTL2	R/W	0x0000.0000	ADC Sample Sequence Control 2	367
0x088	ADCSSFIFO2	RO	0x0000.0000	ADC Sample Sequence Result FIFO 2	364
0x08C	ADCSSFSTAT2	RO	0x0000.0100	ADC Sample Sequence FIFO 2 Status	365
0x0A0	ADCSSMUX3	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 3	369
0x0A4	ADCSSCTL3	R/W	0x0000.0002	ADC Sample Sequence Control 3	370
0x0A8	ADCSSFIFO3	RO	0x0000.0000	ADC Sample Sequence Result FIFO 3	364
0x0AC	ADCSSFSTAT3	RO	0x0000.0100	ADC Sample Sequence FIFO 3 Status	365

## 13.5 Register Descriptions

The remainder of this section lists and describes the ADC registers, in numerical order by address offset.

## Register 1: ADC Active Sample Sequencer (ADCACTSS), offset 0x000

This register controls the activation of the Sample Sequencers. Each Sample Sequencer can be enabled/disabled independently.

### ADC Active Sample Sequencer (ADCACTSS)

Base 0x4003.8000  
Offset 0x000  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												ASEN3	ASEN2	ASEN1	ASEN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

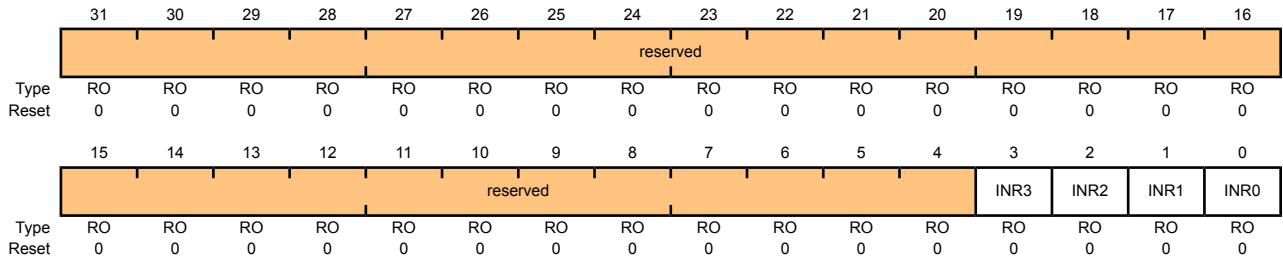
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	ASEN3	R/W	0	<p>ADC SS3 Enable</p> <p>Specifies whether Sample Sequencer 3 is enabled. If set, the sample sequence logic for Sequencer 3 is active. Otherwise, the Sequencer is inactive.</p>
2	ASEN2	R/W	0	<p>ADC SS2 Enable</p> <p>Specifies whether Sample Sequencer 2 is enabled. If set, the sample sequence logic for Sequencer 2 is active. Otherwise, the Sequencer is inactive.</p>
1	ASEN1	R/W	0	<p>ADC SS1 Enable</p> <p>Specifies whether Sample Sequencer 1 is enabled. If set, the sample sequence logic for Sequencer 1 is active. Otherwise, the Sequencer is inactive.</p>
0	ASEN0	R/W	0	<p>ADC SS0 Enable</p> <p>Specifies whether Sample Sequencer 0 is enabled. If set, the sample sequence logic for Sequencer 0 is active. Otherwise, the Sequencer is inactive.</p>

## Register 2: ADC Raw Interrupt Status (ADCRIS), offset 0x004

This register shows the status of the raw interrupt signal of each Sample Sequencer. These bits may be polled by software to look for interrupt conditions without having to generate controller interrupts.

### ADC Raw Interrupt Status (ADCRIS)

Base 0x4003.8000  
 Offset 0x004  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	INR3	RO	0	SS3 Raw Interrupt Status  Set by hardware when a sample with its respective <b>ADCSSCTL3</b> <small>IE</small> bit has completed conversion. This bit is cleared by writing a 1 to the <b>ADCISC</b> <small>IN3</small> bit.
2	INR2	RO	0	SS2 Raw Interrupt Status  Set by hardware when a sample with its respective <b>ADCSSCTL2</b> <small>IE</small> bit has completed conversion. This bit is cleared by writing a 1 to the <b>ADCISC</b> <small>IN2</small> bit.
1	INR1	RO	0	SS1 Raw Interrupt Status  Set by hardware when a sample with its respective <b>ADCSSCTL1</b> <small>IE</small> bit has completed conversion. This bit is cleared by writing a 1 to the <b>ADCISC</b> <small>IN1</small> bit.
0	INR0	RO	0	SS0 Raw Interrupt Status  Set by hardware when a sample with its respective <b>ADCSSCTL0</b> <small>IE</small> bit has completed conversion. This bit is cleared by writing a 1 to the <b>ADCISC</b> <small>IN0</small> bit.

### Register 3: ADC Interrupt Mask (ADCIM), offset 0x008

This register controls whether the Sample Sequencer raw interrupt signals are promoted to controller interrupts. The raw interrupt signal for each Sample Sequencer can be masked independently.

#### ADC Interrupt Mask (ADCIM)

Base 0x4003.8000  
Offset 0x008  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												MASK3	MASK2	MASK1	MASK0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	MASK3	R/W	0	SS3 Interrupt Mask  Specifies whether the raw interrupt signal from Sample Sequencer 3 ( <b>ADCRIS</b> register <b>INR3</b> bit) is promoted to a controller interrupt. If set, the raw interrupt signal is promoted to a controller interrupt. Otherwise, it is not.
2	MASK2	R/W	0	SS2 Interrupt Mask  Specifies whether the raw interrupt signal from Sample Sequencer 2 ( <b>ADCRIS</b> register <b>INR2</b> bit) is promoted to a controller interrupt. If set, the raw interrupt signal is promoted to a controller interrupt. Otherwise, it is not.
1	MASK1	R/W	0	SS1 Interrupt Mask  Specifies whether the raw interrupt signal from Sample Sequencer 1 ( <b>ADCRIS</b> register <b>INR1</b> bit) is promoted to a controller interrupt. If set, the raw interrupt signal is promoted to a controller interrupt. Otherwise, it is not.
0	MASK0	R/W	0	SS0 Interrupt Mask  Specifies whether the raw interrupt signal from Sample Sequencer 0 ( <b>ADCRIS</b> register <b>INR0</b> bit) is promoted to a controller interrupt. If set, the raw interrupt signal is promoted to a controller interrupt. Otherwise, it is not.

### Register 4: ADC Interrupt Status and Clear (ADCISC), offset 0x00C

This register provides the mechanism for clearing interrupt conditions, and shows the status of controller interrupts generated by the Sample Sequencers. When read, each bit field is the logical AND of the respective *INR* and *MASK* bits. Interrupts are cleared by writing a 1 to the corresponding bit position. If software is polling the **ADCRIS** instead of generating interrupts, the *INR* bits are still cleared via the **ADCISC** register, even if the *IN* bit is not set.

#### ADC Interrupt Status and Clear (ADCISC)

Base 0x4003.8000  
 Offset 0x00C  
 Type R/W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved													IN3	IN2	IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	IN3	R/W1C	0	SS3 Interrupt Status and Clear  This bit is set by hardware when the <i>MASK3</i> and <i>INR3</i> bits are both 1, providing a level-based interrupt to the controller. It is cleared by writing a 1, and also clears the <i>INR3</i> bit.
2	IN2	R/W1C	0	SS2 Interrupt Status and Clear  This bit is set by hardware when the <i>MASK2</i> and <i>INR2</i> bits are both 1, providing a level based interrupt to the controller. It is cleared by writing a 1, and also clears the <i>INR2</i> bit.
1	IN1	R/W1C	0	SS1 Interrupt Status and Clear  This bit is set by hardware when the <i>MASK1</i> and <i>INR1</i> bits are both 1, providing a level based interrupt to the controller. It is cleared by writing a 1, and also clears the <i>INR1</i> bit.
0	IN0	R/W1C	0	SS0 Interrupt Status and Clear  This bit is set by hardware when the <i>MASK0</i> and <i>INR0</i> bits are both 1, providing a level based interrupt to the controller. It is cleared by writing a 1, and also clears the <i>INR0</i> bit.

## Register 5: ADC Overflow Status (ADCOSTAT), offset 0x010

This register indicates overflow conditions in the Sample Sequencer FIFOs. Once the overflow condition has been handled by software, the condition can be cleared by writing a 1 to the corresponding bit position.

### ADC Overflow Status (ADCOSTAT)

Base 0x4003.8000

Offset 0x010

Type R/W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved												OV3	OV2	OV1	OV0	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

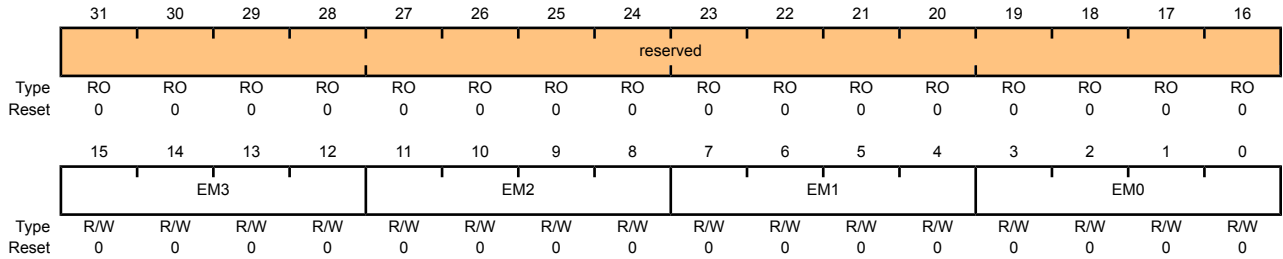
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	OV3	R/W1C	0	<p>SS3 FIFO Overflow</p> <p>This bit specifies that the FIFO for Sample Sequencer 3 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped and this bit is set by hardware to indicate the occurrence of dropped data. This bit is cleared by writing a 1.</p>
2	OV2	R/W1C	0	<p>SS2 FIFO Overflow</p> <p>This bit specifies that the FIFO for Sample Sequencer 2 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped and this bit is set by hardware to indicate the occurrence of dropped data. This bit is cleared by writing a 1.</p>
1	OV1	R/W1C	0	<p>SS1 FIFO Overflow</p> <p>This bit specifies that the FIFO for Sample Sequencer 1 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped and this bit is set by hardware to indicate the occurrence of dropped data. This bit is cleared by writing a 1.</p>
0	OV0	R/W1C	0	<p>SS0 FIFO Overflow</p> <p>This bit specifies that the FIFO for Sample Sequencer 0 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped and this bit is set by hardware to indicate the occurrence of dropped data. This bit is cleared by writing a 1.</p>

### Register 6: ADC Event Multiplexer Select (ADCEMUX), offset 0x014

The **ADCEMUX** selects the event (trigger) that initiates sampling for each Sample Sequencer. Each Sample Sequencer can be configured with a unique trigger source.

#### ADC Event Multiplexer Select (ADCEMUX)

Base 0x4003.8000  
 Offset 0x014  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description																								
31:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.																								
15:12	EM3	R/W	0x00	SS3 Trigger Select  This field selects the trigger source for Sample Sequencer 3.  The valid configurations for this field are:  <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Value</th> <th>Event</th> </tr> </thead> <tbody> <tr><td>0x0</td><td>Controller (default)</td></tr> <tr><td>0x1</td><td>Reserved</td></tr> <tr><td>0x2</td><td>Reserved</td></tr> <tr><td>0x3</td><td>Reserved</td></tr> <tr><td>0x4</td><td>External (GPIO PB4)</td></tr> <tr><td>0x5</td><td>Timer</td></tr> <tr><td>0x6</td><td>Reserved</td></tr> <tr><td>0x7</td><td>Reserved</td></tr> <tr><td>0x8</td><td>Reserved</td></tr> <tr><td>0x9-0xE</td><td>reserved</td></tr> <tr><td>0xF</td><td>Always (continuously sample)</td></tr> </tbody> </table>	Value	Event	0x0	Controller (default)	0x1	Reserved	0x2	Reserved	0x3	Reserved	0x4	External (GPIO PB4)	0x5	Timer	0x6	Reserved	0x7	Reserved	0x8	Reserved	0x9-0xE	reserved	0xF	Always (continuously sample)
Value	Event																											
0x0	Controller (default)																											
0x1	Reserved																											
0x2	Reserved																											
0x3	Reserved																											
0x4	External (GPIO PB4)																											
0x5	Timer																											
0x6	Reserved																											
0x7	Reserved																											
0x8	Reserved																											
0x9-0xE	reserved																											
0xF	Always (continuously sample)																											



Bit/Field	Name	Type	Reset	Description																								
11:8	EM2	R/W	0x00	<p>SS2 Trigger Select</p> <p>This field selects the trigger source for Sample Sequencer 2.</p> <p>The valid configurations for this field are:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Event</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Controller (default)</td> </tr> <tr> <td>0x1</td> <td>Reserved</td> </tr> <tr> <td>0x2</td> <td>Reserved</td> </tr> <tr> <td>0x3</td> <td>Reserved</td> </tr> <tr> <td>0x4</td> <td>External (GPIO PB4)</td> </tr> <tr> <td>0x5</td> <td>Timer</td> </tr> <tr> <td>0x6</td> <td>Reserved</td> </tr> <tr> <td>0x7</td> <td>Reserved</td> </tr> <tr> <td>0x8</td> <td>Reserved</td> </tr> <tr> <td>0x9-0xE</td> <td>reserved</td> </tr> <tr> <td>0xF</td> <td>Always (continuously sample)</td> </tr> </tbody> </table>	Value	Event	0x0	Controller (default)	0x1	Reserved	0x2	Reserved	0x3	Reserved	0x4	External (GPIO PB4)	0x5	Timer	0x6	Reserved	0x7	Reserved	0x8	Reserved	0x9-0xE	reserved	0xF	Always (continuously sample)
Value	Event																											
0x0	Controller (default)																											
0x1	Reserved																											
0x2	Reserved																											
0x3	Reserved																											
0x4	External (GPIO PB4)																											
0x5	Timer																											
0x6	Reserved																											
0x7	Reserved																											
0x8	Reserved																											
0x9-0xE	reserved																											
0xF	Always (continuously sample)																											
7:4	EM1	R/W	0x00	<p>SS1 Trigger Select</p> <p>This field selects the trigger source for Sample Sequencer 1.</p> <p>The valid configurations for this field are:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Event</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Controller (default)</td> </tr> <tr> <td>0x1</td> <td>Reserved</td> </tr> <tr> <td>0x2</td> <td>Reserved</td> </tr> <tr> <td>0x3</td> <td>Reserved</td> </tr> <tr> <td>0x4</td> <td>External (GPIO PB4)</td> </tr> <tr> <td>0x5</td> <td>Timer</td> </tr> <tr> <td>0x6</td> <td>Reserved</td> </tr> <tr> <td>0x7</td> <td>Reserved</td> </tr> <tr> <td>0x8</td> <td>Reserved</td> </tr> <tr> <td>0x9-0xE</td> <td>reserved</td> </tr> <tr> <td>0xF</td> <td>Always (continuously sample)</td> </tr> </tbody> </table>	Value	Event	0x0	Controller (default)	0x1	Reserved	0x2	Reserved	0x3	Reserved	0x4	External (GPIO PB4)	0x5	Timer	0x6	Reserved	0x7	Reserved	0x8	Reserved	0x9-0xE	reserved	0xF	Always (continuously sample)
Value	Event																											
0x0	Controller (default)																											
0x1	Reserved																											
0x2	Reserved																											
0x3	Reserved																											
0x4	External (GPIO PB4)																											
0x5	Timer																											
0x6	Reserved																											
0x7	Reserved																											
0x8	Reserved																											
0x9-0xE	reserved																											
0xF	Always (continuously sample)																											

Bit/Field	Name	Type	Reset	Description																								
3:0	EM0	R/W	0x00	SS0 Trigger Select This field selects the trigger source for Sample Sequencer 0. The valid configurations for this field are:  <table><thead><tr><th>Value</th><th>Event</th></tr></thead><tbody><tr><td>0x0</td><td>Controller (default)</td></tr><tr><td>0x1</td><td>Reserved</td></tr><tr><td>0x2</td><td>Reserved</td></tr><tr><td>0x3</td><td>Reserved</td></tr><tr><td>0x4</td><td>External (GPIO PB4)</td></tr><tr><td>0x5</td><td>Timer</td></tr><tr><td>0x6</td><td>Reserved</td></tr><tr><td>0x7</td><td>Reserved</td></tr><tr><td>0x8</td><td>Reserved</td></tr><tr><td>0x9-0xE</td><td>reserved</td></tr><tr><td>0xF</td><td>Always (continuously sample)</td></tr></tbody></table>	Value	Event	0x0	Controller (default)	0x1	Reserved	0x2	Reserved	0x3	Reserved	0x4	External (GPIO PB4)	0x5	Timer	0x6	Reserved	0x7	Reserved	0x8	Reserved	0x9-0xE	reserved	0xF	Always (continuously sample)
Value	Event																											
0x0	Controller (default)																											
0x1	Reserved																											
0x2	Reserved																											
0x3	Reserved																											
0x4	External (GPIO PB4)																											
0x5	Timer																											
0x6	Reserved																											
0x7	Reserved																											
0x8	Reserved																											
0x9-0xE	reserved																											
0xF	Always (continuously sample)																											

## Register 7: ADC Underflow Status (ADCUSTAT), offset 0x018

This register indicates underflow conditions in the Sample Sequencer FIFOs. The corresponding underflow condition can be cleared by writing a 1 to the relevant bit position.

### ADC Underflow Status (ADCUSTAT)

Base 0x4003.8000

Offset 0x018

Type R/W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												UV3	UV2	UV1	UV0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

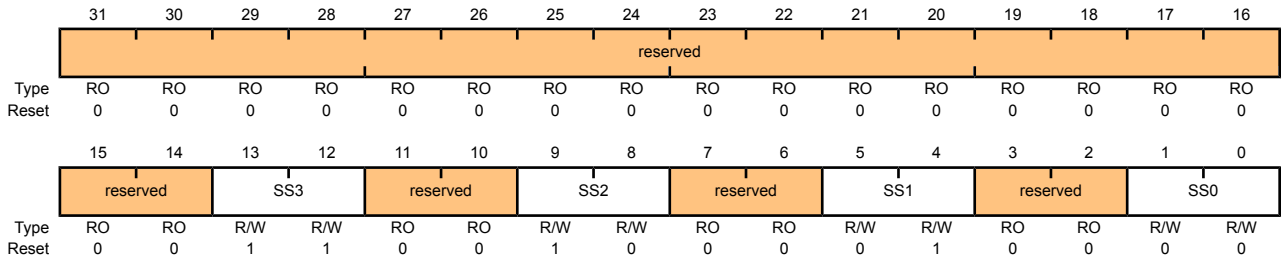
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	UV3	R/W1C	0	<p>SS3 FIFO Underflow</p> <p>This bit specifies that the FIFO for Sample Sequencer 3 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. This bit is cleared by writing a 1.</p>
2	UV2	R/W1C	0	<p>SS2 FIFO Underflow</p> <p>This bit specifies that the FIFO for Sample Sequencer 2 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. This bit is cleared by writing a 1.</p>
1	UV1	R/W1C	0	<p>SS1 FIFO Underflow</p> <p>This bit specifies that the FIFO for Sample Sequencer 1 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. This bit is cleared by writing a 1.</p>
0	UV0	R/W1C	0	<p>SS0 FIFO Underflow</p> <p>This bit specifies that the FIFO for Sample Sequencer 0 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. This bit is cleared by writing a 1.</p>

### Register 8: ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020

This register sets the priority for each of the Sample Sequencers. Out of reset, Sequencer 0 has the highest priority, and sample sequence 3 has the lowest priority. When reconfiguring sequence priorities, each sequence must have a unique priority or the ADC behavior is inconsistent.

#### ADC Sample Sequencer Priority (ADCSSPRI)

Base 0x4003.8000  
 Offset 0x020  
 Type R/W, reset 0x0000.3210



Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13:12	SS3	R/W	0x3	SS3 Priority  The SS3 field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 3. A priority encoding of 0 is highest and 3 is lowest. The priorities assigned to the Sequencers must be uniquely mapped. ADC behavior is not consistent if two or more fields are equal.
11:10	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:8	SS2	R/W	0x2	SS2 Priority  The SS2 field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 2.
7:6	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:4	SS1	R/W	0x1	SS1 Priority  The SS1 field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 1.
3:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1:0	SS0	R/W	0x0	SS0 Priority  The SS0 field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 0.

## Register 9: ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028

This register provides a mechanism for application software to initiate sampling in the Sample Sequencers. Sample sequences can be initiated individually or in any combination. When multiple sequences are triggered simultaneously, the priority encodings in **ADCSSPRI** dictate execution order.

### ADC Processor Sample Sequence Initiate (ADCPSSI)

Base 0x4003.8000

Offset 0x028

Type WO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved													SS3	SS2	SS1	SS0
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	

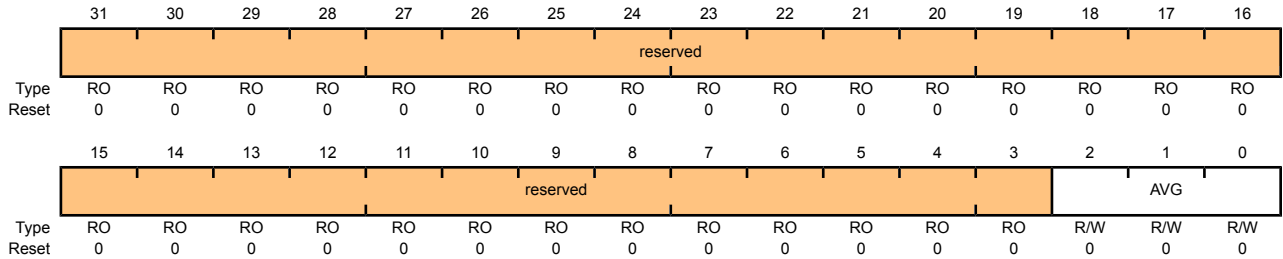
Bit/Field	Name	Type	Reset	Description
31:4	reserved	WO	-	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	SS3	WO	-	SS3 Initiate  Only a write by software is valid; a read of the register returns no meaningful data. When set by software, sampling is triggered on Sample Sequencer 3, assuming the Sequencer is enabled in the <b>ADCACTSS</b> register.
2	SS2	WO	-	SS2 Initiate  Only a write by software is valid; a read of the register returns no meaningful data. When set by software, sampling is triggered on Sample Sequencer 2, assuming the Sequencer is enabled in the <b>ADCACTSS</b> register.
1	SS1	WO	-	SS1 Initiate  Only a write by software is valid; a read of the register returns no meaningful data. When set by software, sampling is triggered on Sample Sequencer 1, assuming the Sequencer is enabled in the <b>ADCACTSS</b> register.
0	SS0	WO	-	SS0 Initiate  Only a write by software is valid; a read of the register returns no meaningful data. When set by software, sampling is triggered on Sample Sequencer 0, assuming the Sequencer is enabled in the <b>ADCACTSS</b> register.

### Register 10: ADC Sample Averaging Control (ADCSAC), offset 0x030

This register controls the amount of hardware averaging applied to conversion results. The final conversion result stored in the FIFO is averaged from  $2^{AVG}$  consecutive ADC samples at the specified ADC speed. If AVG is 0, the sample is passed directly through without any averaging. If AVG=6, then 64 consecutive ADC samples are averaged to generate one result in the sequencer FIFO. An AVG = 7 provides unpredictable results.

#### ADC Sample Averaging Control (ADCSAC)

Base 0x4003.8000  
 Offset 0x030  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	AVG	R/W	0x0	Hardware Averaging Control  Specifies the amount of hardware averaging that will be applied to ADC samples. The AVG field can be any value between 0 and 6. Entering a value of 7 creates unpredictable results.

Value	Description
0x0	No hardware oversampling
0x1	2x hardware oversampling
0x2	4x hardware oversampling
0x3	8x hardware oversampling
0x4	16x hardware oversampling
0x5	32x hardware oversampling
0x6	64x hardware oversampling
0x7	Reserved

## Register 11: ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040

This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 0.

This register is 32-bits wide and contains information for eight possible samples.

### ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0)

Base 0x4003.8000  
Offset 0x040  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved	MUX7			reserved	MUX6			reserved	MUX5			reserved	MUX4		
Type	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	MUX3			reserved	MUX2			reserved	MUX1			reserved	MUX0		
Type	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30:28	MUX7	R/W	0	8th Sample Input Select  The MUX7 field is used during the eighth sample of a sequence executed with the Sample Sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion. The value set here indicates the corresponding pin, for example, a value of 1 indicates the input is ADC1.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26:24	MUX6	R/W	0	7th Sample Input Select  The MUX6 field is used during the seventh sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
23	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22:20	MUX5	R/W	0	6th Sample Input Select  The MUX5 field is used during the sixth sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
19	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
18:16	MUX4	R/W	0	<b>5th Sample Input Select</b>  The MUX4 field is used during the fifth sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14:12	MUX3	R/W	0	<b>4th Sample Input Select</b>  The MUX3 field is used during the fourth sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10:8	MUX2	R/W	0	<b>3rd Sample Input Select</b>  The MUX2 field is used during the third sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:4	MUX1	R/W	0	<b>2nd Sample Input Select</b>  The MUX1 field is used during the second sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	MUX0	R/W	0	<b>1st Sample Input Select</b>  The MUX0 field is used during the first sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.



## Register 12: ADC Sample Sequence Control 0 (ADCSCTL0), offset 0x044

This register contains the configuration information for each sample for a sequence executed with Sample Sequencer 0. When configuring a sample sequence, the `END` bit must be set at some point, whether it be after the first sample, last sample, or any sample in between.

This register is 32-bits wide and contains information for eight possible samples.

### ADC Sample Sequence Control 0 (ADCSCTL0)

Base 0x4003.8000  
Offset 0x044  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TS7	IE7	END7	D7	TS6	IE6	END6	D6	TS5	IE5	END5	D5	TS4	IE4	END4	D4
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	TS7	R/W	0	8th Sample Temp Sensor Select  The <code>TS7</code> bit is used during the eighth sample of the sample sequence and specifies the input source of the sample. If set, the temperature sensor is read. Otherwise, the input pin specified by the <code>ADCSSMUX</code> register is read.
30	IE7	R/W	0	8th Sample Interrupt Enable  The <code>IE7</code> bit is used during the eighth sample of the sample sequence and specifies whether the raw interrupt signal ( <code>INR0</code> bit) is asserted at the end of the sample's conversion. If the <code>MASK0</code> bit in the <code>ADCIM</code> register is set, the interrupt is promoted to a controller-level interrupt. When this bit is set, the raw interrupt is asserted, otherwise it is not. It is legal to have multiple samples within a sequence generate interrupts.
29	END7	R/W	0	8th Sample is End of Sequence  The <code>END7</code> bit indicates that this is the last sample of the sequence. It is possible to end the sequence on any sample position. Samples defined after the sample containing a set <code>END</code> are not requested for conversion even though the fields may be non-zero. It is required that software write the <code>END</code> bit somewhere within the sequence. (Sample Sequencer 3, which only has a single sample in the sequence, is hardwired to have the <code>END0</code> bit set.)  Setting this bit indicates that this sample is the last in the sequence.
28	D7	R/W	0	8th Sample Diff Input Select  The <code>D7</code> bit indicates that the analog input is to be differentially sampled. The corresponding <code>ADCSSMUXx</code> nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". The temperature sensor does not have a differential option. When set, the analog inputs are differentially sampled.
27	TS6	R/W	0	7th Sample Temp Sensor Select  Same definition as <code>TS7</code> but used during the seventh sample.

Bit/Field	Name	Type	Reset	Description
26	IE6	R/W	0	7th Sample Interrupt Enable Same definition as IE7 but used during the seventh sample.
25	END6	R/W	0	7th Sample is End of Sequence Same definition as END7 but used during the seventh sample.
24	D6	R/W	0	7th Sample Diff Input Select Same definition as D7 but used during the seventh sample.
23	TS5	R/W	0	6th Sample Temp Sensor Select Same definition as TS7 but used during the sixth sample.
22	IE5	R/W	0	6th Sample Interrupt Enable Same definition as IE7 but used during the sixth sample.
21	END5	R/W	0	6th Sample is End of Sequence Same definition as END7 but used during the sixth sample.
20	D5	R/W	0	6th Sample Diff Input Select Same definition as D7 but used during the sixth sample.
19	TS4	R/W	0	5th Sample Temp Sensor Select Same definition as TS7 but used during the fifth sample.
18	IE4	R/W	0	5th Sample Interrupt Enable Same definition as IE7 but used during the fifth sample.
17	END4	R/W	0	5th Sample is End of Sequence Same definition as END7 but used during the fifth sample.
16	D4	R/W	0	5th Sample Diff Input Select Same definition as D7 but used during the fifth sample.
15	TS3	R/W	0	4th Sample Temp Sensor Select Same definition as TS7 but used during the fourth sample.
14	IE3	R/W	0	4th Sample Interrupt Enable Same definition as IE7 but used during the fourth sample.
13	END3	R/W	0	4th Sample is End of Sequence Same definition as END7 but used during the fourth sample.
12	D3	R/W	0	4th Sample Diff Input Select Same definition as D7 but used during the fourth sample.
11	TS2	R/W	0	3rd Sample Temp Sensor Select Same definition as TS7 but used during the third sample.

Bit/Field	Name	Type	Reset	Description
10	IE2	R/W	0	3rd Sample Interrupt Enable Same definition as IE7 but used during the third sample.
9	END2	R/W	0	3rd Sample is End of Sequence Same definition as END7 but used during the third sample.
8	D2	R/W	0	3rd Sample Diff Input Select Same definition as D7 but used during the third sample.
7	TS1	R/W	0	2nd Sample Temp Sensor Select Same definition as TS7 but used during the second sample.
6	IE1	R/W	0	2nd Sample Interrupt Enable Same definition as IE7 but used during the second sample.
5	END1	R/W	0	2nd Sample is End of Sequence Same definition as END7 but used during the second sample.
4	D1	R/W	0	2nd Sample Diff Input Select Same definition as D7 but used during the second sample.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as TS7 but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as IE7 but used during the first sample.
1	END0	R/W	0	1st Sample is End of Sequence Same definition as END7 but used during the first sample. Since this sequencer has only one entry, this bit must be set.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as D7 but used during the first sample.

**Register 13: ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048**

**Register 14: ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068**

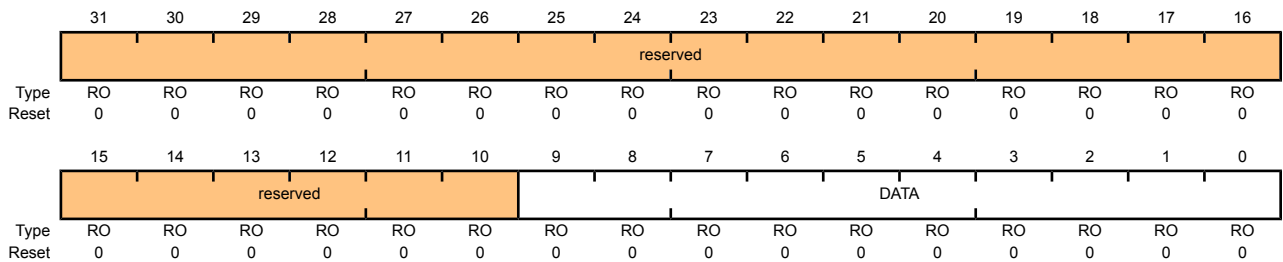
**Register 15: ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088**

**Register 16: ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8**

This register contains the conversion results for samples collected with the Sample Sequencer (the **ADCSSFIFO0** register is used for Sample Sequencer 0, **ADCSSFIFO1** for Sequencer 1, **ADCSSFIFO2** for Sequencer 2, and **ADCSSFIFO3** for Sequencer 3). Reads of this register return conversion result data in the order sample 0, sample 1, and so on, until the FIFO is empty. If the FIFO is not properly handled by software, overflow and underflow conditions are registered in the **ADCOSTAT** and **ADCUSTAT** registers.

ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0)

Base 0x4003.8000  
 Offset 0x048  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:0	DATA	RO	0x00	Conversion Result Data

**Register 17: ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C**

**Register 18: ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C**

**Register 19: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C**

**Register 20: ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC**

This register provides a window into the Sample Sequencer, providing full/empty status information as well as the positions of the head and tail pointers. The reset value of 0x100 indicates an empty FIFO. The **ADCSSFSTAT0** register provides status on FIFO0, **ADCSSFSTAT1** on FIFO1, **ADCSSFSTAT2** on FIFO2, and **ADCSSFSTAT3** on FIFO3.

#### ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0)

Base 0x4003.8000  
Offset 0x04C  
Type RO, reset 0x0000.0100

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			FULL	reserved			EMPTY	HPTR				TPTR			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:13	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	FULL	RO	0	FIFO Full When set, indicates that the FIFO is currently full.
11:9	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	EMPTY	RO	1	FIFO Empty When set, indicates that the FIFO is currently empty.
7:4	HPTR	RO	0x00	FIFO Head Pointer This field contains the current "head" pointer index for the FIFO, that is, the next entry to be written.
3:0	TPTR	RO	0x00	FIFO Tail Pointer This field contains the current "tail" pointer index for the FIFO, that is, the next entry to be read.

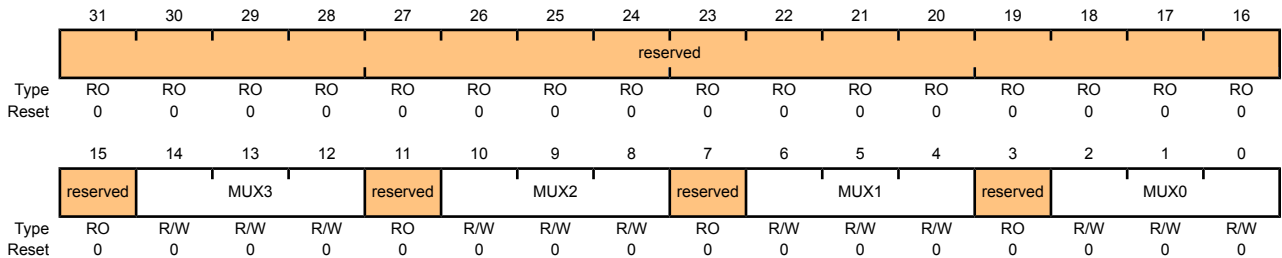
**Register 21: ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060**

**Register 22: ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080**

This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 1 or 2. These registers are 16-bits wide and contain information for four possible samples. See the **ADCSSMUX0** register on page 359 for detailed bit descriptions.

ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1)

Base 0x4003.8000  
 Offset 0x060  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:15	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14:12	MUX3	R/W	0	4th Sample Input Select
11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10:8	MUX2	R/W	0	3rd Sample Input Select
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:4	MUX1	R/W	0	2nd Sample Input Select
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	MUX0	R/W	0	1st Sample Input Select

**Register 23: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064****Register 24: ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084**

These registers contain the configuration information for each sample for a sequence executed with Sample Sequencer 1 or 2. When configuring a sample sequence, the **END** bit must be set at some point, whether it be after the first sample, last sample, or any sample in between. This register is 16-bits wide and contains information for four possible samples. See the **ADCSSCTL0** register on page 361 for detailed bit descriptions.

**ADC Sample Sequence Control 1 (ADCSSCTL1)**

Base 0x4003.8000  
Offset 0x064  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	TS3	R/W	0	4th Sample Temp Sensor Select Same definition as <b>TS7</b> but used during the fourth sample.
14	IE3	R/W	0	4th Sample Interrupt Enable Same definition as <b>IE7</b> but used during the fourth sample.
13	END3	R/W	0	4th Sample is End of Sequence Same definition as <b>END7</b> but used during the fourth sample.
12	D3	R/W	0	4th Sample Diff Input Select Same definition as <b>D7</b> but used during the fourth sample.
11	TS2	R/W	0	3rd Sample Temp Sensor Select Same definition as <b>TS7</b> but used during the third sample.
10	IE2	R/W	0	3rd Sample Interrupt Enable Same definition as <b>IE7</b> but used during the third sample.
9	END2	R/W	0	3rd Sample is End of Sequence Same definition as <b>END7</b> but used during the third sample.
8	D2	R/W	0	3rd Sample Diff Input Select Same definition as <b>D7</b> but used during the third sample.

Bit/Field	Name	Type	Reset	Description
7	TS1	R/W	0	2nd Sample Temp Sensor Select Same definition as TS7 but used during the second sample.
6	IE1	R/W	0	2nd Sample Interrupt Enable Same definition as IE7 but used during the second sample.
5	END1	R/W	0	2nd Sample is End of Sequence Same definition as END7 but used during the second sample.
4	D1	R/W	0	2nd Sample Diff Input Select Same definition as D7 but used during the second sample.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as TS7 but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as IE7 but used during the first sample.
1	END0	R/W	0	1st Sample is End of Sequence Same definition as END7 but used during the first sample. Since this sequencer has only one entry, this bit must be set.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as D7 but used during the first sample.



## Register 25: ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0

This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 3. This register is 4-bits wide and contains information for one possible sample. See the **ADCSSMUX0** register on page 359 for detailed bit descriptions.

### ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3)

Base 0x4003.8000  
Offset 0x0A0  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													MUX0		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	MUX0	R/W	0	1st Sample Input Select

### Register 26: ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4

This register contains the configuration information for each sample for a sequence executed with Sample Sequencer 3. The `END` bit is always set since there is only one sample in this sequencer. This register is 4-bits wide and contains information for one possible sample. See the **ADCSSCTL0** register on page 361 for detailed bit descriptions.

#### ADC Sample Sequence Control 3 (ADCSSCTL3)

Base 0x4003.8000  
 Offset 0x0A4  
 Type R/W, reset 0x0000.0002

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved												TS0	IE0	END0	D0	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as <code>TS7</code> but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as <code>IE7</code> but used during the first sample.
1	END0	R/W	1	1st Sample is End of Sequence Same definition as <code>END7</code> but used during the first sample. Since this sequencer has only one entry, this bit must be set.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as <code>D7</code> but used during the first sample.

## 14 Universal Asynchronous Receivers/Transmitters (UARTs)

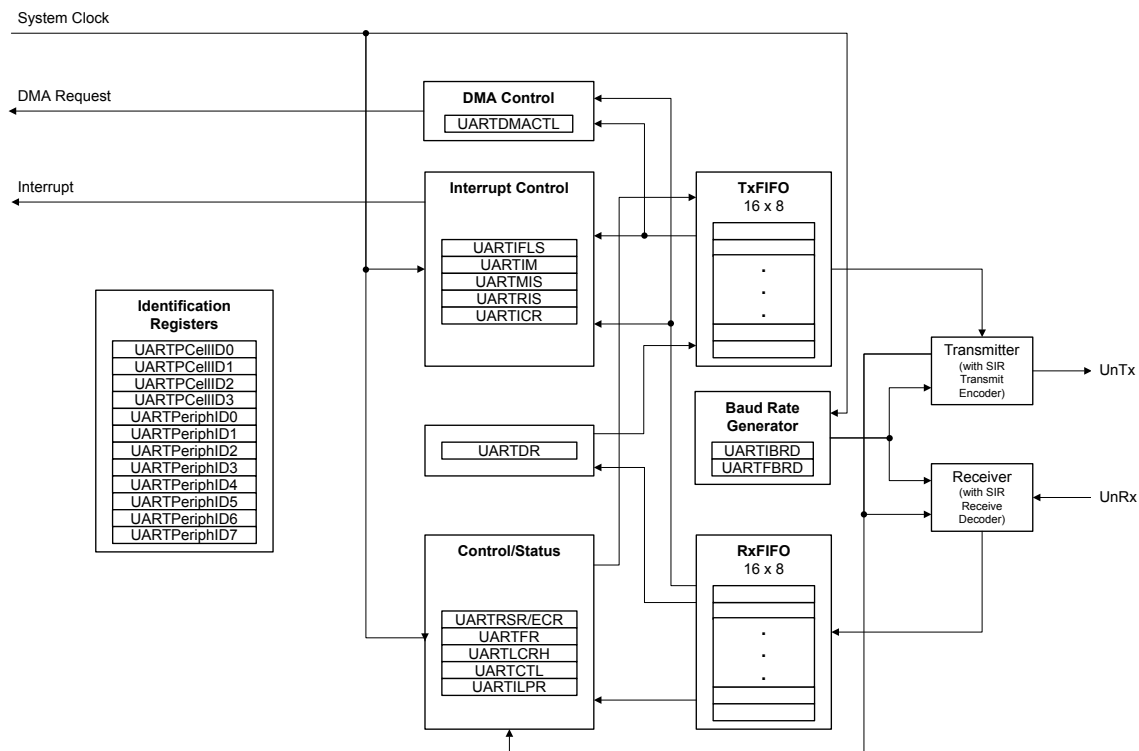
The Stellaris<sup>®</sup> Universal Asynchronous Receiver/Transmitter (UART) provides fully programmable, 16C550-type serial interface characteristics. The LM3S1607 controller is equipped with three UART modules.

Each UART has the following features:

- Separate transmit and receive FIFOs
- Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
- FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
- Programmable baud-rate generator allowing rates up to 3.125 Mbps
- Standard asynchronous communication bits for start, stop, and parity
- False start bit detection
- Line-break generation and detection
- Fully programmable serial interface characteristics:
  - 5, 6, 7, or 8 data bits
  - Even, odd, stick, or no-parity bit generation/detection
  - 1 or 2 stop bit generation
- IrDA serial-IR (SIR) encoder/decoder providing:
  - Programmable use of IrDA Serial Infrared (SIR) or UART input/output
  - Support of IrDA SIR encoder/decoder functions for data rates up to 115.2 Kbps half-duplex
  - Support of normal 3/16 and low-power (1.41-2.23  $\mu$ s) bit durations
  - Programmable internal clock generator enabling division of reference clock by 1 to 256 for low-power mode bit duration
- Dedicated DMA transmit and receive channels

## 14.1 Block Diagram

Figure 14-1. UART Module Block Diagram



## 14.2 Functional Description

Each Stellaris<sup>®</sup> UART performs the functions of parallel-to-serial and serial-to-parallel conversions. It is similar in functionality to a 16C550 UART, but is not register compatible.

The UART is configured for transmit and/or receive via the **TXE** and **RXE** bits of the **UART Control (UARTCTL)** register (see page 391). Transmit and receive are both enabled out of reset. Before any control registers are programmed, the UART must be disabled by clearing the **UARTEN** bit in **UARTCTL**. If the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

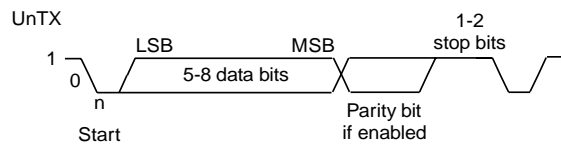
The UART peripheral also includes a serial IR (SIR) encoder/decoder block that can be connected to an infrared transceiver to implement an IrDA SIR physical layer. The SIR function is programmed using the **UARTCTL** register.

### 14.2.1 Transmit/Receive Logic

The transmit logic performs parallel-to-serial conversion on the data read from the transmit FIFO. The control logic outputs the serial bit stream beginning with a start bit, and followed by the data bits (LSB first), parity bit, and the stop bits according to the programmed configuration in the control registers. See Figure 14-2 on page 373 for details.

The receive logic performs serial-to-parallel conversion on the received bit stream after a valid start pulse has been detected. Overrun, parity, frame error checking, and line-break detection are also performed, and their status accompanies the data that is written to the receive FIFO.

Figure 14-2. UART Character Frame



## 14.2.2 Baud-Rate Generation

The baud-rate divisor is a 22-bit number consisting of a 16-bit integer and a 6-bit fractional part. The number formed by these two values is used by the baud-rate generator to determine the bit period. Having a fractional baud-rate divider allows the UART to generate all the standard baud rates.

The 16-bit integer is loaded through the **UART Integer Baud-Rate Divisor (UARTIBRD)** register (see page 387) and the 6-bit fractional part is loaded with the **UART Fractional Baud-Rate Divisor (UARTFBRD)** register (see page 388). The baud-rate divisor (BRD) has the following relationship to the system clock (where *BRDI* is the integer part of the BRD and *BRDF* is the fractional part, separated by a decimal place.)

$$BRD = BRDI + BRDF = \text{UARTSysClk} / (16 * \text{Baud Rate})$$

where *UARTSysClk* is the system clock connected to the UART.

The 6-bit fractional number (that is to be loaded into the *DIVFRAC* bit field in the **UARTFBRD** register) can be calculated by taking the fractional part of the baud-rate divisor, multiplying it by 64, and adding 0.5 to account for rounding errors:

$$\text{UARTFBRD}[\text{DIVFRAC}] = \text{integer}(\text{BRDF} * 64 + 0.5)$$

The UART generates an internal baud-rate reference clock at 16x the baud-rate (referred to as *Baud16*). This reference clock is divided by 16 to generate the transmit clock, and is used for error detection during receive operations.

Along with the **UART Line Control, High Byte (UARTLCRH)** register (see page 389), the **UARTIBRD** and **UARTFBRD** registers form an internal 30-bit register. This internal register is only updated when a write operation to **UARTLCRH** is performed, so any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register for the changes to take effect.

To update the baud-rate registers, there are four possible sequences:

- **UARTIBRD** write, **UARTFBRD** write, and **UARTLCRH** write
- **UARTFBRD** write, **UARTIBRD** write, and **UARTLCRH** write
- **UARTIBRD** write and **UARTLCRH** write
- **UARTFBRD** write and **UARTLCRH** write

## 14.2.3 Data Transmission

Data received or transmitted is stored in two 16-byte FIFOs, though the receive FIFO has an extra four bits per character for status information. For transmission, data is written into the transmit FIFO. If the UART is enabled, it causes a data frame to start transmitting with the parameters indicated in the **UARTLCRH** register. Data continues to be transmitted until there is no data left in the transmit

FIFO. The `BUSY` bit in the **UART Flag (UARTFR)** register (see page 384) is asserted as soon as data is written to the transmit FIFO (that is, if the FIFO is non-empty) and remains asserted while data is being transmitted. The `BUSY` bit is negated only when the transmit FIFO is empty, and the last character has been transmitted from the shift register, including the stop bits. The UART can indicate that it is busy even though the UART may no longer be enabled.

When the receiver is idle (the `UnRx` is continuously 1) and the data input goes Low (a start bit has been received), the receive counter begins running and data is sampled on the eighth cycle of `Baud16` (described in “Transmit/Receive Logic” on page 372).

The start bit is valid if `UnRx` is still low on the eighth cycle of `Baud16`, otherwise a false start bit is detected and it is ignored. Start bit errors can be viewed in the **UART Receive Status (UARTSR)** register (see page 382). If the start bit was valid, successive data bits are sampled on every 16th cycle of `Baud16` (that is, one bit period later) according to the programmed length of the data characters. The parity bit is then checked if parity mode was enabled. Data length and parity are defined in the **UARTLCRH** register.

Lastly, a valid stop bit is confirmed if `UnRx` is High, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO, with any error bits associated with that word.

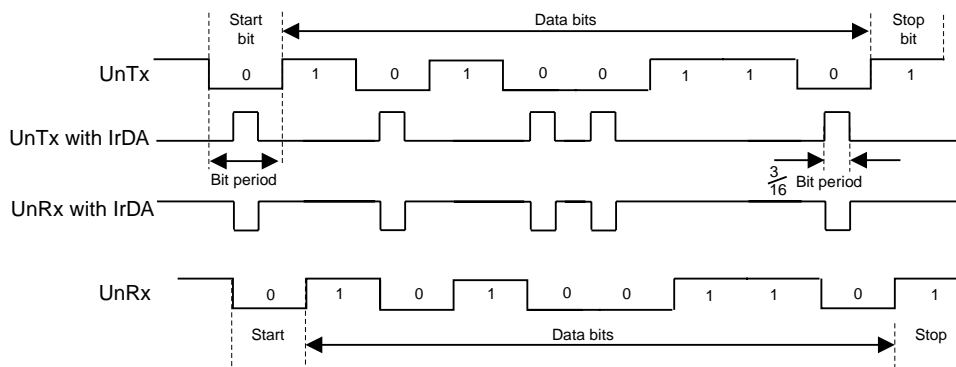
#### 14.2.4 Serial IR (SIR)

The UART peripheral includes an IrDA serial-IR (SIR) encoder/decoder block. The IrDA SIR block provides functionality that converts between an asynchronous UART data stream, and half-duplex serial SIR interface. No analog processing is performed on-chip. The role of the SIR block is to provide a digital encoded output, and decoded input to the UART. The UART signal pins can be connected to an infrared transceiver to implement an IrDA SIR physical layer link. The SIR block has two modes of operation:

- In normal IrDA mode, a zero logic level is transmitted as high pulse of 3/16th duration of the selected baud rate bit period on the output pin, while logic one levels are transmitted as a static LOW signal. These levels control the driver of an infrared transmitter, sending a pulse of light for each zero. On the reception side, the incoming light pulses energize the photo transistor base of the receiver, pulling its output LOW. This drives the UART input pin LOW.
- In low-power IrDA mode, the width of the transmitted infrared pulse is set to three times the period of the internally generated `IrLPBaud16` signal (1.63  $\mu$ s, assuming a nominal 1.8432 MHz frequency) by changing the appropriate bit in the **UARTCR** register. See page 386 for more information on IrDA low-power pulse-duration configuration.

Figure 14-3 on page 375 shows the UART transmit and receive signals, with and without IrDA modulation.

Figure 14-3. IrDA Data Modulation



In both normal and low-power IrDA modes:

- During transmission, the UART data bit is used as the base for encoding
- During reception, the decoded bits are transferred to the UART receive logic

The IrDA SIR physical layer specifies a half-duplex communication link, with a minimum 10 ms delay between transmission and reception. This delay must be generated by software because it is not automatically supported by the UART. The delay is required because the infrared receiver electronics might become biased, or even saturated from the optical power coupled from the adjacent transmitter LED. This delay is known as latency, or receiver setup time.

### 14.2.5 FIFO Operation

The UART has two 16-entry FIFOs; one for transmit and one for receive. Both FIFOs are accessed via the **UART Data (UARTDR)** register (see page 380). Read operations of the **UARTDR** register return a 12-bit value consisting of 8 data bits and 4 error flags while write operations place 8-bit data in the transmit FIFO.

Out of reset, both FIFOs are disabled and act as 1-byte-deep holding registers. The FIFOs are enabled by setting the `FEN` bit in **UARTLCRH** (page 389).

FIFO status can be monitored via the **UART Flag (UARTFR)** register (see page 384) and the **UART Receive Status (UARTRSR)** register. Hardware monitors empty, full and overrun conditions. The **UARTFR** register contains empty and full flags (`TXFE`, `TXFF`, `RXFE`, and `RXFF` bits) and the **UARTRSR** register shows overrun status via the `OE` bit.

The trigger points at which the FIFOs generate interrupts is controlled via the **UART Interrupt FIFO Level Select (UARTIFLS)** register (see page 393). Both FIFOs can be individually configured to trigger interrupts at different levels. Available configurations include  $\frac{1}{8}$ ,  $\frac{1}{4}$ ,  $\frac{1}{2}$ ,  $\frac{3}{4}$ , and  $\frac{7}{8}$ . For example, if the  $\frac{1}{4}$  option is selected for the receive FIFO, the UART generates a receive interrupt after 4 data bytes are received. Out of reset, both FIFOs are configured to trigger an interrupt at the  $\frac{1}{2}$  mark.

### 14.2.6 Interrupts

The UART can generate interrupts when the following conditions are observed:

- Overrun Error
- Break Error

- Parity Error
- Framing Error
- Receive Timeout
- Transmit (when condition defined in the `TXIFLSEL` bit in the **UARTIFLS** register is met)
- Receive (when condition defined in the `RXIFLSEL` bit in the **UARTIFLS** register is met)

All of the interrupt events are ORed together before being sent to the interrupt controller, so the UART can only generate a single interrupt request to the controller at any given time. Software can service multiple interrupt events in a single interrupt service routine by reading the **UART Masked Interrupt Status (UARTMIS)** register (see page 398).

The interrupt events that can trigger a controller-level interrupt are defined in the **UART Interrupt Mask (UARTIM)** register (see page 395) by setting the corresponding `IM` bit to 1. If interrupts are not used, the raw interrupt status is always visible via the **UART Raw Interrupt Status (UARTRIS)** register (see page 397).

Interrupts are always cleared (for both the **UARTMIS** and **UARTRIS** registers) by setting the corresponding bit in the **UART Interrupt Clear (UARTICR)** register (see page 399).

The receive timeout interrupt is asserted when the receive FIFO is not empty, and no further data is received over a 32-bit period. The receive timeout interrupt is cleared either when the FIFO becomes empty through reading all the data (or by reading the holding register), or when a 1 is written to the corresponding bit in the **UARTICR** register.

### 14.2.7 Loopback Operation

The UART can be placed into an internal loopback mode for diagnostic or debug work. This is accomplished by setting the `LBE` bit in the **UARTCTL** register (see page 391). In loopback mode, data transmitted on `UnTx` is received on the `UnRx` input.

### 14.2.8 DMA Operation

The UART provides an interface connected to the  $\mu$ DMA controller. The DMA operation of the UART is enabled through the **UART DMA Control (UARTDMACTL)** register. When DMA operation is enabled, the UART will assert a DMA request on the receive or transmit channel when the associated FIFO can transfer data. For the receive channel, a single transfer request is asserted whenever there is any data in the receive FIFO. A burst transfer request is asserted whenever the amount of data in the receive FIFO is at or above the FIFO trigger level. For the transmit channel, a single transfer request is asserted whenever there is at least one empty location in the transmit FIFO. The burst request is asserted whenever the transmit FIFO contains fewer characters than the FIFO trigger level. The single and burst DMA transfer requests are handled automatically by the  $\mu$ DMA controller depending on how the DMA channel is configured.

To enable DMA operation for the receive channel, the `RXDMAE` bit of the **DMA Control (UARTDMACTL)** register should be set. To enable DMA operation for the transmit channel, the `TXDMAE` bit of **UARTDMACTL** should be set. The UART can also be configured to stop using DMA for the receive channel if a receive error occurs. If the `DMAERR` bit of **UARTDMACR** is set, then when a receive error occurs, the DMA receive requests will be automatically disabled. This error condition can be cleared by clearing the UART error interrupt.

If DMA is enabled, then the  $\mu$ DMA controller will trigger an interrupt when a transfer is complete. The interrupt will occur on the UART interrupt vector. Therefore, if interrupts are used for UART



operation and DMA is enabled, the UART interrupt handler must be designed to handle the  $\mu$ DMA completion interrupt.

See “Micro Direct Memory Access ( $\mu$ DMA)” on page 176 for more details about programming the  $\mu$ DMA controller.

### 14.2.9 IrDA SIR block

The IrDA SIR block contains an IrDA serial IR (SIR) protocol encoder/decoder. When enabled, the SIR block uses the `UnTx` and `UnRx` pins for the SIR protocol, which should be connected to an IR transceiver.

The SIR block can receive and transmit, but it is only half-duplex so it cannot do both at the same time. Transmission must be stopped before data can be received. The IrDA SIR physical layer specifies a minimum 10-ms delay between transmission and reception.

## 14.3 Initialization and Configuration

To use the UARTs, the peripheral clock must be enabled by setting the `UART0`, `UART1`, or `UART2` bits in the **RCGC1** register.

This section discusses the steps that are required to use a UART module. For this example, the UART clock is assumed to be 20 MHz and the desired UART configuration is:

- 115200 baud rate
- Data length of 8 bits
- One stop bit
- No parity
- FIFOs disabled
- No interrupts

The first thing to consider when programming the UART is the baud-rate divisor (BRD), since the **UARTIBRD** and **UARTFBRD** registers must be written before the **UARTLCRH** register. Using the equation described in “Baud-Rate Generation” on page 373, the BRD can be calculated:

$$\text{BRD} = 20,000,000 / (16 * 115,200) = 10.8507$$

which means that the `DIVINT` field of the **UARTIBRD** register (see page 387) should be set to 10. The value to be loaded into the **UARTFBRD** register (see page 388) is calculated by the equation:

$$\text{UARTFBRD}[\text{DIVFRAC}] = \text{integer}(0.8507 * 64 + 0.5) = 54$$

With the BRD values in hand, the UART configuration is written to the module in the following order:

1. Disable the UART by clearing the `UARTEN` bit in the **UARTCTL** register.
2. Write the integer portion of the BRD to the **UARTIBRD** register.
3. Write the fractional portion of the BRD to the **UARTFBRD** register.
4. Write the desired serial parameters to the **UARTLCRH** register (in this case, a value of `0x0000.0060`).

5. Optionally, configure the uDMA channel (see “Micro Direct Memory Access (μDMA)” on page 176) and enable the DMA option(s) in the **UARTDMACTL** register.
6. Enable the UART by setting the **UARTEN** bit in the **UARTCTL** register.

## 14.4 Register Map

Table 14-1 on page 378 lists the UART registers. The offset listed is a hexadecimal increment to the register’s address, relative to that UART’s base address:

- UART0: 0x4000.C000
- UART1: 0x4000.D000
- UART2: 0x4000.E000

**Note:** The UART must be disabled (see the **UARTEN** bit in the **UARTCTL** register on page 391) before any of the control registers are reprogrammed. When the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

**Table 14-1. UART Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	UARTDR	R/W	0x0000.0000	UART Data	380
0x004	UARTSR/UARTECR	R/W	0x0000.0000	UART Receive Status/Error Clear	382
0x018	UARTFR	RO	0x0000.0090	UART Flag	384
0x020	UARTILPR	R/W	0x0000.0000	UART IrDA Low-Power Register	386
0x024	UARTIBRD	R/W	0x0000.0000	UART Integer Baud-Rate Divisor	387
0x028	UARTFBRD	R/W	0x0000.0000	UART Fractional Baud-Rate Divisor	388
0x02C	UARTLCRH	R/W	0x0000.0000	UART Line Control	389
0x030	UARTCTL	R/W	0x0000.0300	UART Control	391
0x034	UARTIFLS	R/W	0x0000.0012	UART Interrupt FIFO Level Select	393
0x038	UARTIM	R/W	0x0000.0000	UART Interrupt Mask	395
0x03C	UARTRIS	RO	0x0000.000F	UART Raw Interrupt Status	397
0x040	UARTMIS	RO	0x0000.0000	UART Masked Interrupt Status	398
0x044	UARTICR	W1C	0x0000.0000	UART Interrupt Clear	399
0x048	UARTDMACTL	R/W	0x0000.0000	UART DMA Control	401
0xFD0	UARTPeriphID4	RO	0x0000.0000	UART Peripheral Identification 4	402
0xFD4	UARTPeriphID5	RO	0x0000.0000	UART Peripheral Identification 5	403
0xFD8	UARTPeriphID6	RO	0x0000.0000	UART Peripheral Identification 6	404
0xFDC	UARTPeriphID7	RO	0x0000.0000	UART Peripheral Identification 7	405
0xFE0	UARTPeriphID0	RO	0x0000.0011	UART Peripheral Identification 0	406

Offset	Name	Type	Reset	Description	See page
0xFE4	UARTPeriphID1	RO	0x0000.0000	UART Peripheral Identification 1	407
0xFE8	UARTPeriphID2	RO	0x0000.0018	UART Peripheral Identification 2	408
0xFEC	UARTPeriphID3	RO	0x0000.0001	UART Peripheral Identification 3	409
0xFF0	UARTPCellID0	RO	0x0000.000D	UART PrimeCell Identification 0	410
0xFF4	UARTPCellID1	RO	0x0000.00F0	UART PrimeCell Identification 1	411
0xFF8	UARTPCellID2	RO	0x0000.0005	UART PrimeCell Identification 2	412
0xFFC	UARTPCellID3	RO	0x0000.00B1	UART PrimeCell Identification 3	413

## 14.5 Register Descriptions

The remainder of this section lists and describes the UART registers, in numerical order by address offset.

### Register 1: UART Data (UARTDR), offset 0x000

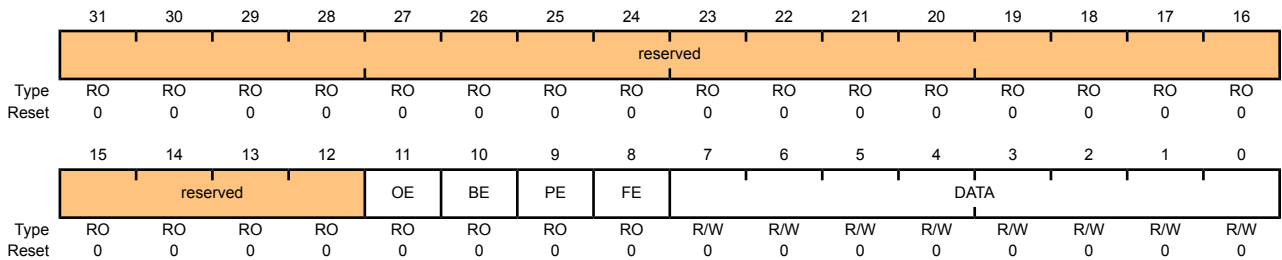
This register is the data register (the interface to the FIFOs).

When FIFOs are enabled, data written to this location is pushed onto the transmit FIFO. If FIFOs are disabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). A write to this register initiates a transmission from the UART.

For received data, if the FIFO is enabled, the data byte and the 4-bit status (break, frame, parity, and overrun) is pushed onto the 12-bit wide receive FIFO. If FIFOs are disabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO). The received data can be retrieved by reading this register.

#### UART Data (UARTDR)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x000  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	OE	RO	0	UART Overrun Error  The OE values are defined as follows:  Value Description 0 There has been no data loss due to a FIFO overrun. 1 New data was received when the FIFO was full, resulting in data loss.
10	BE	RO	0	UART Break Error  This bit is set to 1 when a break condition is detected, indicating that the receive data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits).  In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the received data input goes to a 1 (marking state) and the next valid start bit is received.

---

Bit/Field	Name	Type	Reset	Description
9	PE	RO	0	UART Parity Error  This bit is set to 1 when the parity of the received data character does not match the parity defined by bits 2 and 7 of the <b>UARTLCRH</b> register.  In FIFO mode, this error is associated with the character at the top of the FIFO.
8	FE	RO	0	UART Framing Error  This bit is set to 1 when the received character does not have a valid stop bit (a valid stop bit is 1).
7:0	DATA	R/W	0	Data Transmitted or Received  When written, the data that is to be transmitted via the UART. When read, the data that was received by the UART.

## Register 2: UART Receive Status/Error Clear (UARTRSR/UARTECR), offset 0x004

The **UARTRSR/UARTECR** register is the receive status register/error clear register.

In addition to the **UARTDR** register, receive status can also be read from the **UARTRSR** register. If the status is read from this register, then the status information corresponds to the entry read from **UARTDR** prior to reading **UARTRSR**. The status information for overrun is set immediately when an overrun condition occurs.

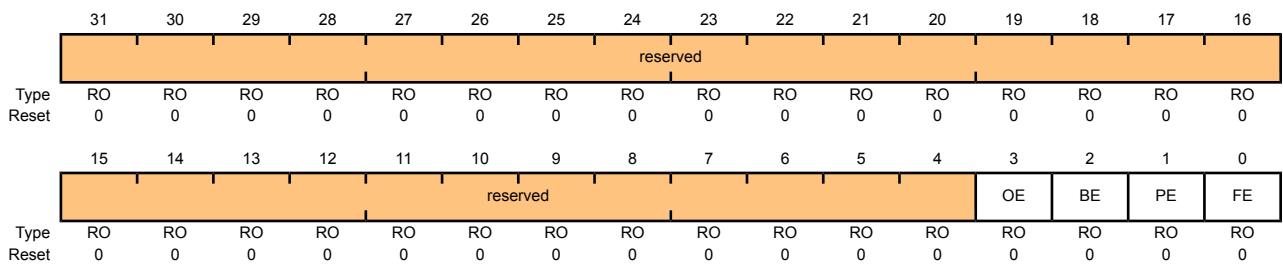
The **UARTRSR** register cannot be written.

A write of any value to the **UARTECR** register clears the framing, parity, break, and overrun errors. All the bits are cleared to 0 on reset.

### Read-Only Receive Status (UARTRSR) Register

UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x004  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	OE	RO	0	UART Overrun Error  When this bit is set to 1, data is received and the FIFO is already full. This bit is cleared to 0 by a write to <b>UARTECR</b> .  The FIFO contents remain valid since no further data is written when the FIFO is full, only the contents of the shift register are overwritten. The CPU must now read the data in order to empty the FIFO.
2	BE	RO	0	UART Break Error  This bit is set to 1 when a break condition is detected, indicating that the received data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits).  This bit is cleared to 0 by a write to <b>UARTECR</b> .  In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state) and the next valid start bit is received.

Bit/Field	Name	Type	Reset	Description
1	PE	RO	0	<p>UART Parity Error</p> <p>This bit is set to 1 when the parity of the received data character does not match the parity defined by bits 2 and 7 of the <b>UARTLCRH</b> register.</p> <p>This bit is cleared to 0 by a write to <b>UARTECR</b>.</p>
0	FE	RO	0	<p>UART Framing Error</p> <p>This bit is set to 1 when the received character does not have a valid stop bit (a valid stop bit is 1).</p> <p>This bit is cleared to 0 by a write to <b>UARTECR</b>.</p> <p>In FIFO mode, this error is associated with the character at the top of the FIFO.</p>

### Write-Only Error Clear (UARTECR) Register

#### UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x004  
 Type WO, reset 0x0000.0000



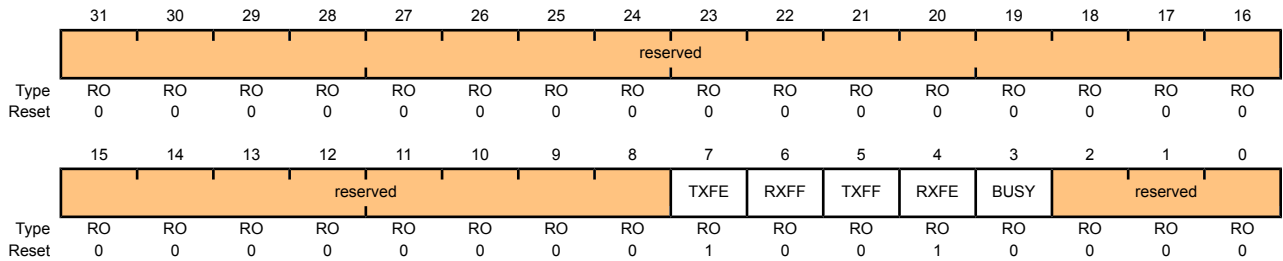
Bit/Field	Name	Type	Reset	Description
31:8	reserved	WO	0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>
7:0	DATA	WO	0	<p>Error Clear</p> <p>A write to this register of any data clears the framing, parity, break, and overrun flags.</p>

### Register 3: UART Flag (UARTFR), offset 0x018

The **UARTFR** register is the flag register. After reset, the **TXFF**, **RXFF**, and **BUSY** bits are 0, and **TXFE** and **RXFE** bits are 1.

#### UART Flag (UARTFR)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x018  
 Type RO, reset 0x0000.0090



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TXFE	RO	1	UART Transmit FIFO Empty  The meaning of this bit depends on the state of the <b>FEN</b> bit in the <b>UARTLCRH</b> register.  If the FIFO is disabled ( <b>FEN</b> is 0), this bit is set when the transmit holding register is empty.  If the FIFO is enabled ( <b>FEN</b> is 1), this bit is set when the transmit FIFO is empty.
6	RXFF	RO	0	UART Receive FIFO Full  The meaning of this bit depends on the state of the <b>FEN</b> bit in the <b>UARTLCRH</b> register.  If the FIFO is disabled, this bit is set when the receive holding register is full.  If the FIFO is enabled, this bit is set when the receive FIFO is full.
5	TXFF	RO	0	UART Transmit FIFO Full  The meaning of this bit depends on the state of the <b>FEN</b> bit in the <b>UARTLCRH</b> register.  If the FIFO is disabled, this bit is set when the transmit holding register is full.  If the FIFO is enabled, this bit is set when the transmit FIFO is full.



---

Bit/Field	Name	Type	Reset	Description
4	RXFE	RO	1	UART Receive FIFO Empty  The meaning of this bit depends on the state of the <code>FEN</code> bit in the <b>UARTLCRH</b> register.  If the FIFO is disabled, this bit is set when the receive holding register is empty.  If the FIFO is enabled, this bit is set when the receive FIFO is empty.
3	BUSY	RO	0	UART Busy  When this bit is 1, the UART is busy transmitting data. This bit remains set until the complete byte, including all stop bits, has been sent from the shift register.  This bit is set as soon as the transmit FIFO becomes non-empty (regardless of whether UART is enabled).
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

### Register 4: UART IrDA Low-Power Register (UARTILPR), offset 0x020

The **UARTILPR** register is an 8-bit read/write register that stores the low-power counter divisor value used to derive the low-power SIR pulse width clock by dividing down the system clock (SysClk). All the bits are cleared to 0 when reset.

The internal  $F_{IrLPBaud16}$  clock is generated by dividing down SysClk according to the low-power divisor value written to **UARTILPR**. The duration of SIR pulses generated when low-power mode is enabled is three times the period of the  $F_{IrLPBaud16}$  clock. The low-power divisor value is calculated as follows:

$$ILPDVSR = SysClk / F_{IrLPBaud16}$$

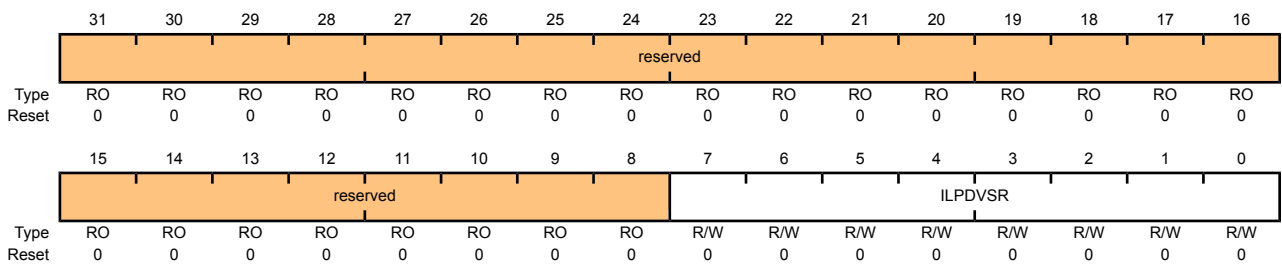
where  $F_{IrLPBaud16}$  is nominally 1.8432 MHz.

You must choose the divisor so that  $1.42 \text{ MHz} < F_{IrLPBaud16} < 2.12 \text{ MHz}$ , which results in a low-power pulse duration of 1.41–2.11  $\mu\text{s}$  (three times the period of  $F_{IrLPBaud16}$ ). The minimum frequency of  $F_{IrLPBaud16}$  ensures that pulses less than one period of  $F_{IrLPBaud16}$  are rejected, but that pulses greater than 1.4  $\mu\text{s}$  are accepted as valid pulses.

**Note:** Zero is an illegal value. Programming a zero value results in no  $F_{IrLPBaud16}$  pulses being generated.

#### UART IrDA Low-Power Register (UARTILPR)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x020  
 Type R/W, reset 0x0000.0000



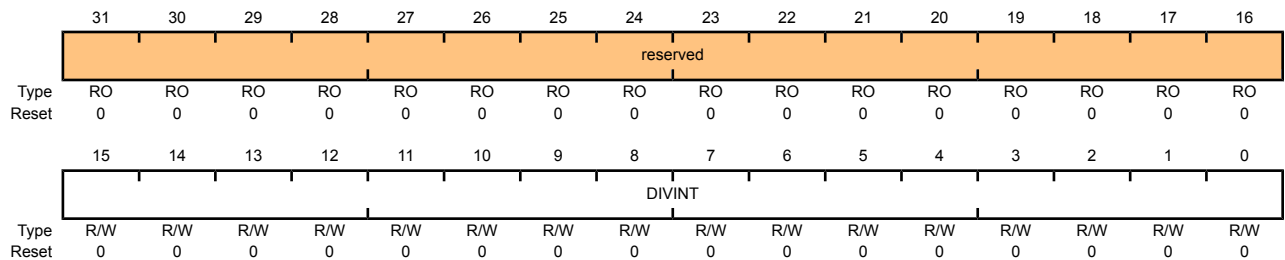
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ILPDVSR	R/W	0x00	IrDA Low-Power Divisor  This is an 8-bit low-power divisor value.

## Register 5: UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024

The **UARTIBRD** register is the integer part of the baud-rate divisor value. All the bits are cleared on reset. The minimum possible divide ratio is 1 (when **UARTIBRD**=0), in which case the **UARTFBRD** register is ignored. When changing the **UARTIBRD** register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register. See “Baud-Rate Generation” on page 373 for configuration details.

### UART Integer Baud-Rate Divisor (UARTIBRD)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x024  
 Type R/W, reset 0x0000.0000



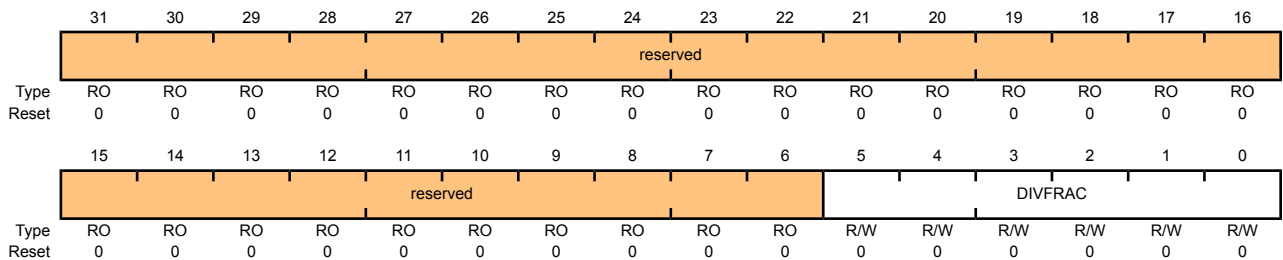
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DIVINT	R/W	0x0000	Integer Baud-Rate Divisor

### Register 6: UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028

The **UARTFBRD** register is the fractional part of the baud-rate divisor value. All the bits are cleared on reset. When changing the **UARTFBRD** register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register. See “Baud-Rate Generation” on page 373 for configuration details.

#### UART Fractional Baud-Rate Divisor (UARTFBRD)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x028  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	DIVFRAC	R/W	0x000	Fractional Baud-Rate Divisor

## Register 7: UART Line Control (UARTLCRH), offset 0x02C

The **UARTLCRH** register is the line control register. Serial parameters such as data length, parity, and stop bit selection are implemented in this register.

When updating the baud-rate divisor (**UARTIBRD** and/or **UARTIFRD**), the **UARTLCRH** register must also be written. The write strobe for the baud-rate divisor registers is tied to the **UARTLCRH** register.

### UART Line Control (UARTLCRH)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x02C  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								SPS	WLEN		FEN	STP2	EPS	PEN	BRK
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description										
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
7	SPS	R/W	0	<p>UART Stick Parity Select</p> <p>When bits 1, 2, and 7 of <b>UARTLCRH</b> are set, the parity bit is transmitted and checked as a 0. When bits 1 and 7 are set and 2 is cleared, the parity bit is transmitted and checked as a 1.</p> <p>When this bit is cleared, stick parity is disabled.</p>										
6:5	WLEN	R/W	0	<p>UART Word Length</p> <p>The bits indicate the number of data bits transmitted or received in a frame as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x3</td> <td>8 bits</td> </tr> <tr> <td>0x2</td> <td>7 bits</td> </tr> <tr> <td>0x1</td> <td>6 bits</td> </tr> <tr> <td>0x0</td> <td>5 bits (default)</td> </tr> </tbody> </table>	Value	Description	0x3	8 bits	0x2	7 bits	0x1	6 bits	0x0	5 bits (default)
Value	Description													
0x3	8 bits													
0x2	7 bits													
0x1	6 bits													
0x0	5 bits (default)													
4	FEN	R/W	0	<p>UART Enable FIFOs</p> <p>If this bit is set to 1, transmit and receive FIFO buffers are enabled (FIFO mode).</p> <p>When cleared to 0, FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers.</p>										

Bit/Field	Name	Type	Reset	Description
3	STP2	R/W	0	<b>UART Two Stop Bits Select</b>  If this bit is set to 1, two stop bits are transmitted at the end of a frame. The receive logic does not check for two stop bits being received.
2	EPS	R/W	0	<b>UART Even Parity Select</b>  If this bit is set to 1, even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits.  When cleared to 0, then odd parity is performed, which checks for an odd number of 1s.  This bit has no effect when parity is disabled by the <code>PEN</code> bit.
1	PEN	R/W	0	<b>UART Parity Enable</b>  If this bit is set to 1, parity checking and generation is enabled; otherwise, parity is disabled and no parity bit is added to the data frame.
0	BRK	R/W	0	<b>UART Send Break</b>  If this bit is set to 1, a Low level is continually output on the <code>U<sub>n</sub>TX</code> output, after completing transmission of the current character. For the proper execution of the break command, the software must set this bit for at least two frames (character periods). For normal use, this bit must be cleared to 0.

## Register 8: UART Control (UARTCTL), offset 0x030

The **UARTCTL** register is the control register. All the bits are cleared on reset except for the Transmit Enable (TXE) and Receive Enable (RXE) bits, which are set to 1.

To enable the UART module, the **UARTEN** bit must be set to 1. If software requires a configuration change in the module, the **UARTEN** bit must be cleared before the configuration changes are written. If the UART is disabled during a transmit or receive operation, the current transaction is completed prior to the UART stopping.

**Note:** The **UARTCTL** register should not be changed while the UART is enabled or else the results are unpredictable. The following sequence is recommended for making changes to the **UARTCTL** register.

1. Disable the UART.
2. Wait for the end of transmission or reception of the current character.
3. Flush the transmit FIFO by disabling bit 4 (**FEN**) in the line control register (**UARTLCRH**).
4. Reprogram the control register.
5. Enable the UART.

### UART Control (UARTCTL)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x030  
 Type R/W, reset 0x0000.0300

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved						RXE	TXE	LBE	reserved				SIRLP	SIREN	UARTEN
Type	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	RXE	R/W	1	UART Receive Enable  If this bit is set to 1, the receive section of the UART is enabled. When the UART is disabled in the middle of a receive, it completes the current character before stopping.

**Note:** To enable reception, the **UARTEN** bit must also be set.

Bit/Field	Name	Type	Reset	Description
8	TXE	R/W	1	<p>UART Transmit Enable</p> <p>If this bit is set to 1, the transmit section of the UART is enabled. When the UART is disabled in the middle of a transmission, it completes the current character before stopping.</p> <p><b>Note:</b> To enable transmission, the <code>UARTEN</code> bit must also be set.</p>
7	LBE	R/W	0	<p>UART Loop Back Enable</p> <p>If this bit is set to 1, the <code>UnTX</code> path is fed through the <code>UnRX</code> path.</p>
6:3	reserved	RO	0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>
2	SIRLP	R/W	0	<p>UART SIR Low Power Mode</p> <p>This bit selects the IrDA encoding mode. If this bit is cleared to 0, low-level bits are transmitted as an active High pulse with a width of 3/16th of the bit period. If this bit is set to 1, low-level bits are transmitted with a pulse width which is 3 times the period of the <code>IrLPBaud16</code> input signal, regardless of the selected bit rate. Setting this bit uses less power, but might reduce transmission distances. See page 386 for more information.</p>
1	SIREN	R/W	0	<p>UART SIR Enable</p> <p>If this bit is set to 1, the IrDA SIR block is enabled, and the UART will transmit and receive data using SIR protocol.</p>
0	UARTEN	R/W	0	<p>UART Enable</p> <p>If this bit is set to 1, the UART is enabled. When the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.</p>



### Register 9: UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034

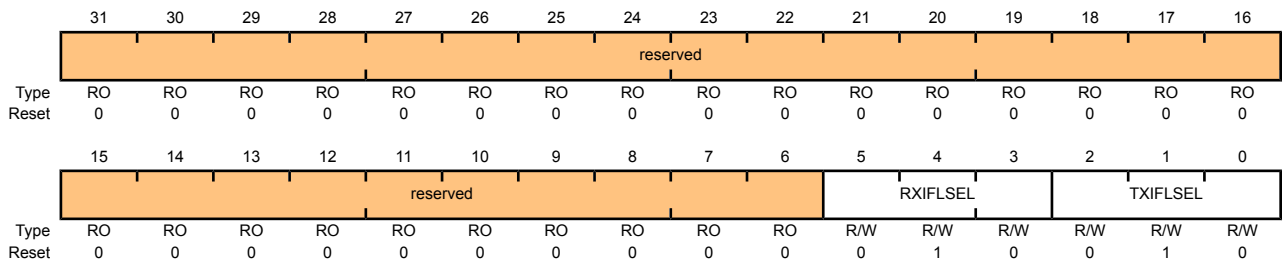
The **UARTIFLS** register is the interrupt FIFO level select register. You can use this register to define the FIFO level at which the **TXRIS** and **RXRIS** bits in the **UARTRIS** register are triggered.

The interrupts are generated based on a transition through a level rather than being based on the level. That is, the interrupts are generated when the fill level progresses through the trigger level. For example, if the receive trigger level is set to the half-way mark, the interrupt is triggered as the module is receiving the 9th character.

Out of reset, the **TXIFLSEL** and **RXIFLSEL** bits are configured so that the FIFOs trigger an interrupt at the half-way mark.

#### UART Interrupt FIFO Level Select (UARTIFLS)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x034  
 Type R/W, reset 0x0000.0012



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:3	RXIFLSEL	R/W	0x2	UART Receive Interrupt FIFO Level Select

The trigger points for the receive interrupt are as follows:

Value	Description
0x0	RX FIFO ≥ 1/8 full
0x1	RX FIFO ≥ 1/4 full
0x2	RX FIFO ≥ 1/2 full (default)
0x3	RX FIFO ≥ 3/4 full
0x4	RX FIFO ≥ 7/8 full
0x5-0x7	Reserved

Bit/Field	Name	Type	Reset	Description
2:0	TXIFLSEL	R/W	0x2	UART Transmit Interrupt FIFO Level Select The trigger points for the transmit interrupt are as follows:  Value Description 0x0 TX FIFO $\leq$ 1/8 full 0x1 TX FIFO $\leq$ 1/4 full 0x2 TX FIFO $\leq$ 1/2 full (default) 0x3 TX FIFO $\leq$ 3/4 full 0x4 TX FIFO $\leq$ 7/8 full 0x5-0x7 Reserved

## Register 10: UART Interrupt Mask (UARTIM), offset 0x038

The **UARTIM** register is the interrupt mask set/clear register.

On a read, this register gives the current value of the mask on the relevant interrupt. Writing a 1 to a bit allows the corresponding raw interrupt signal to be routed to the interrupt controller. Writing a 0 prevents the raw interrupt signal from being sent to the interrupt controller.

### UART Interrupt Mask (UARTIM)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x038  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved					OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM	reserved			
Type	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEIM	R/W	0	UART Overrun Error Interrupt Mask On a read, the current mask for the <b>OEIM</b> interrupt is returned. Setting this bit to 1 promotes the <b>OEIM</b> interrupt to the interrupt controller.
9	BEIM	R/W	0	UART Break Error Interrupt Mask On a read, the current mask for the <b>BEIM</b> interrupt is returned. Setting this bit to 1 promotes the <b>BEIM</b> interrupt to the interrupt controller.
8	PEIM	R/W	0	UART Parity Error Interrupt Mask On a read, the current mask for the <b>PEIM</b> interrupt is returned. Setting this bit to 1 promotes the <b>PEIM</b> interrupt to the interrupt controller.
7	FEIM	R/W	0	UART Framing Error Interrupt Mask On a read, the current mask for the <b>FEIM</b> interrupt is returned. Setting this bit to 1 promotes the <b>FEIM</b> interrupt to the interrupt controller.
6	RTIM	R/W	0	UART Receive Time-Out Interrupt Mask On a read, the current mask for the <b>RTIM</b> interrupt is returned. Setting this bit to 1 promotes the <b>RTIM</b> interrupt to the interrupt controller.
5	TXIM	R/W	0	UART Transmit Interrupt Mask On a read, the current mask for the <b>TXIM</b> interrupt is returned. Setting this bit to 1 promotes the <b>TXIM</b> interrupt to the interrupt controller.

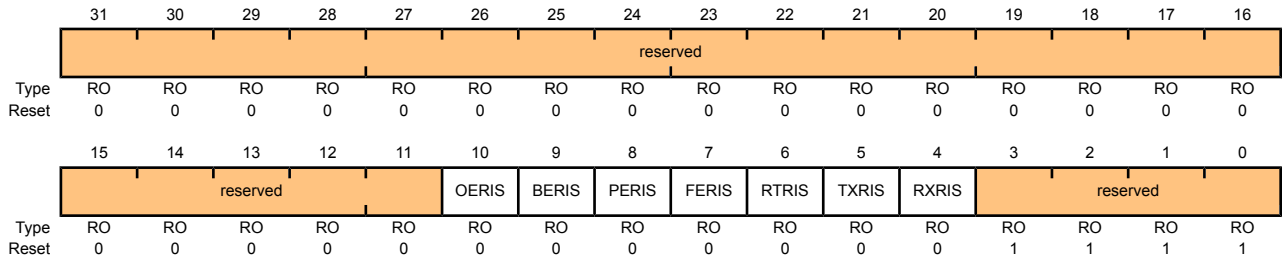
Bit/Field	Name	Type	Reset	Description
4	RXIM	R/W	0	UART Receive Interrupt Mask On a read, the current mask for the <code>RXIM</code> interrupt is returned. Setting this bit to 1 promotes the <code>RXIM</code> interrupt to the interrupt controller.
3:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

### Register 11: UART Raw Interrupt Status (UARTRIS), offset 0x03C

The **UARTRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt. A write has no effect.

#### UART Raw Interrupt Status (UARTRIS)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x03C  
 Type RO, reset 0x0000.000F



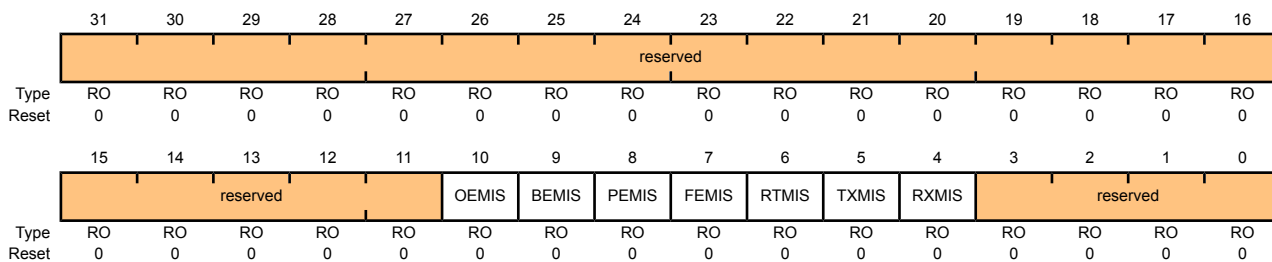
Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OERIS	RO	0	UART Overrun Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
9	BERIS	RO	0	UART Break Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
8	PERIS	RO	0	UART Parity Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
7	FERIS	RO	0	UART Framing Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
6	RTRIS	RO	0	UART Receive Time-Out Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
5	TXRIS	RO	0	UART Transmit Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
4	RXRIS	RO	0	UART Receive Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
3:0	reserved	RO	0xF	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

### Register 12: UART Masked Interrupt Status (UARTMIS), offset 0x040

The **UARTMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

#### UART Masked Interrupt Status (UARTMIS)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x040  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEMIS	RO	0	UART Overrun Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
9	BEMIS	RO	0	UART Break Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
8	PEMIS	RO	0	UART Parity Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
7	FEMIS	RO	0	UART Framing Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
6	RTMIS	RO	0	UART Receive Time-Out Masked Interrupt Status Gives the masked interrupt state of this interrupt.
5	TXMIS	RO	0	UART Transmit Masked Interrupt Status Gives the masked interrupt state of this interrupt.
4	RXMIS	RO	0	UART Receive Masked Interrupt Status Gives the masked interrupt state of this interrupt.
3:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 13: UART Interrupt Clear (UARTICR), offset 0x044

The **UARTICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt (both raw interrupt and masked interrupt, if enabled) is cleared. A write of 0 has no effect.

### UART Interrupt Clear (UARTICR)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x044  
 Type W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved					OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC	reserved			
Type	RO	RO	RO	RO	RO	W1C	W1C	W1C	W1C	W1C	W1C	W1C	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description						
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
10	OEIC	W1C	0	Overrun Error Interrupt Clear The <b>OEIC</b> values are defined as follows: <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No effect on the interrupt.</td> </tr> <tr> <td>1</td> <td>Clears interrupt.</td> </tr> </tbody> </table>	Value	Description	0	No effect on the interrupt.	1	Clears interrupt.
Value	Description									
0	No effect on the interrupt.									
1	Clears interrupt.									
9	BEIC	W1C	0	Break Error Interrupt Clear The <b>BEIC</b> values are defined as follows: <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No effect on the interrupt.</td> </tr> <tr> <td>1</td> <td>Clears interrupt.</td> </tr> </tbody> </table>	Value	Description	0	No effect on the interrupt.	1	Clears interrupt.
Value	Description									
0	No effect on the interrupt.									
1	Clears interrupt.									
8	PEIC	W1C	0	Parity Error Interrupt Clear The <b>PEIC</b> values are defined as follows: <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No effect on the interrupt.</td> </tr> <tr> <td>1</td> <td>Clears interrupt.</td> </tr> </tbody> </table>	Value	Description	0	No effect on the interrupt.	1	Clears interrupt.
Value	Description									
0	No effect on the interrupt.									
1	Clears interrupt.									

Bit/Field	Name	Type	Reset	Description
7	FEIC	W1C	0	Framing Error Interrupt Clear The FEIC values are defined as follows:  Value Description 0 No effect on the interrupt. 1 Clears interrupt.
6	RTIC	W1C	0	Receive Time-Out Interrupt Clear The RTIC values are defined as follows:  Value Description 0 No effect on the interrupt. 1 Clears interrupt.
5	TXIC	W1C	0	Transmit Interrupt Clear The TXIC values are defined as follows:  Value Description 0 No effect on the interrupt. 1 Clears interrupt.
4	RXIC	W1C	0	Receive Interrupt Clear The RXIC values are defined as follows:  Value Description 0 No effect on the interrupt. 1 Clears interrupt.
3:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.



**Register 14: UART DMA Control (UARTDMACTL), offset 0x048**

The **UARTDMACTL** register is the DMA control register.

**UART DMA Control (UARTDMACTL)**

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x048  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													DMAERR	TXDMAE	RXDMAE
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

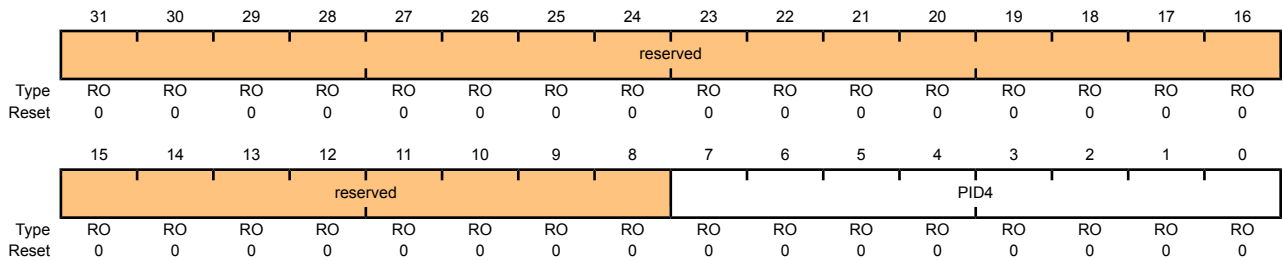
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	DMAERR	R/W	0	DMA on Error  If this bit is set to 1, DMA receive requests are automatically disabled when a receive error occurs.
1	TXDMAE	R/W	0	Transmit DMA Enable  If this bit is set to 1, DMA for the transmit FIFO is enabled.
0	RXDMAE	R/W	0	Receive DMA Enable  If this bit is set to 1, DMA for the receive FIFO is enabled.

### Register 15: UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

#### UART Peripheral Identification 4 (UARTPeriphID4)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0xFD0  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x0000	UART Peripheral ID Register[7:0]  Can be used by software to identify the presence of this peripheral.

**Register 16: UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4**

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

## UART Peripheral Identification 5 (UARTPeriphID5)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0xFD4

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID5							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

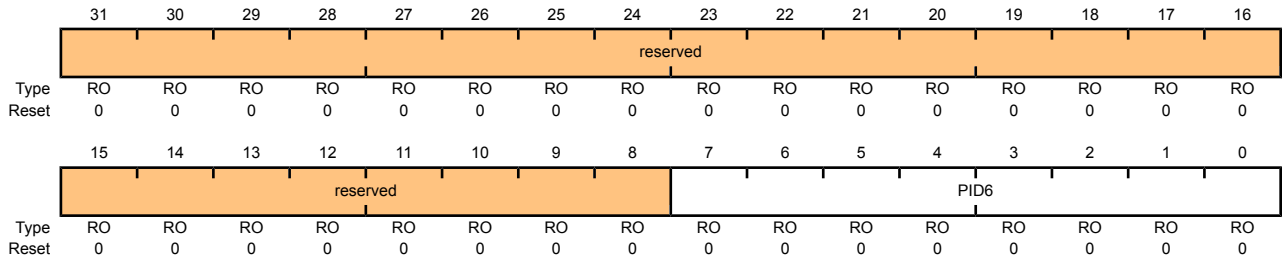
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x0000	UART Peripheral ID Register[15:8] Can be used by software to identify the presence of this peripheral.

## Register 17: UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

### UART Peripheral Identification 6 (UARTPeriphID6)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0xFD8  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x0000	UART Peripheral ID Register[23:16]  Can be used by software to identify the presence of this peripheral.

**Register 18: UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC**

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

## UART Peripheral Identification 7 (UARTPeriphID7)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0xFDC  
 Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID7							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

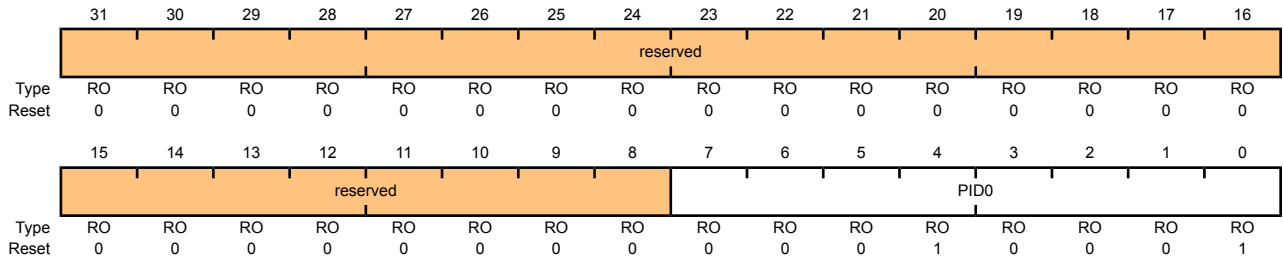
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x0000	UART Peripheral ID Register[31:24]  Can be used by software to identify the presence of this peripheral.

### Register 19: UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

#### UART Peripheral Identification 0 (UARTPeriphID0)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0xFE0  
 Type RO, reset 0x0000.0011



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x11	UART Peripheral ID Register[7:0]  Can be used by software to identify the presence of this peripheral.

**Register 20: UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4**

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

## UART Peripheral Identification 1 (UARTPeriphID1)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0xFE4  
 Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

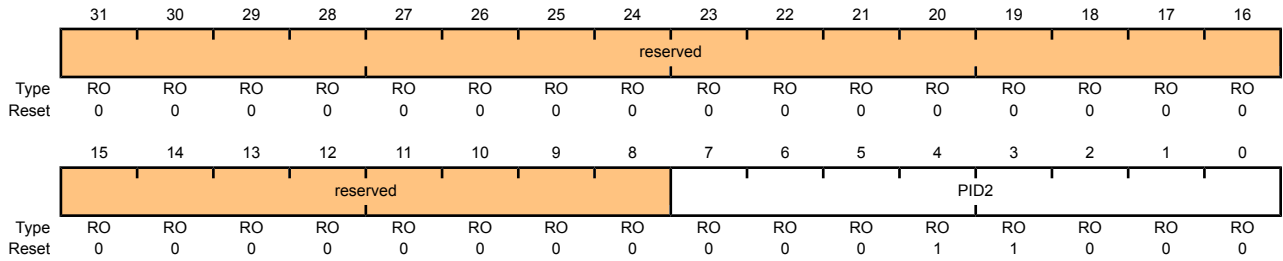
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	UART Peripheral ID Register[15:8]  Can be used by software to identify the presence of this peripheral.

## Register 21: UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

### UART Peripheral Identification 2 (UARTPeriphID2)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0xFE8  
 Type RO, reset 0x0000.0018



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	UART Peripheral ID Register[23:16]  Can be used by software to identify the presence of this peripheral.



## Register 22: UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

### UART Peripheral Identification 3 (UARTPeriphID3)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0xFEC

Type RO, reset 0x0000.0001

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID3							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

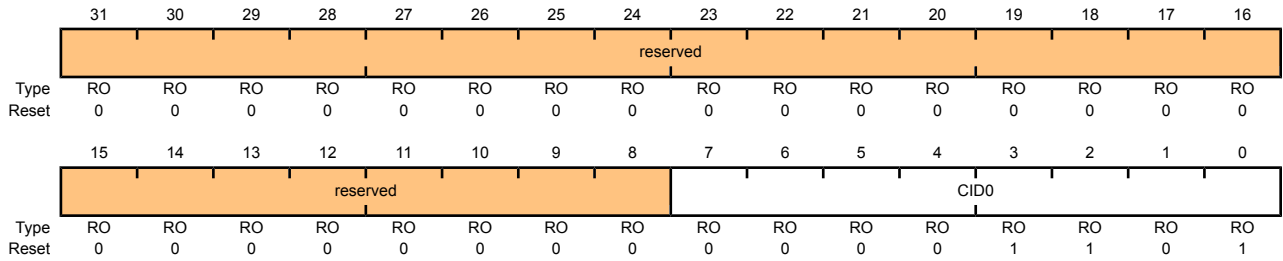
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	UART Peripheral ID Register[31:24] Can be used by software to identify the presence of this peripheral.

### Register 23: UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

#### UART PrimeCell Identification 0 (UARTPCellID0)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0xFF0  
 Type RO, reset 0x0000.000D



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	UART PrimeCell ID Register[7:0] Provides software a standard cross-peripheral identification system.

## Register 24: UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

### UART PrimeCell Identification 1 (UARTPCellID1)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0xFF4  
 Type RO, reset 0x0000.00F0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

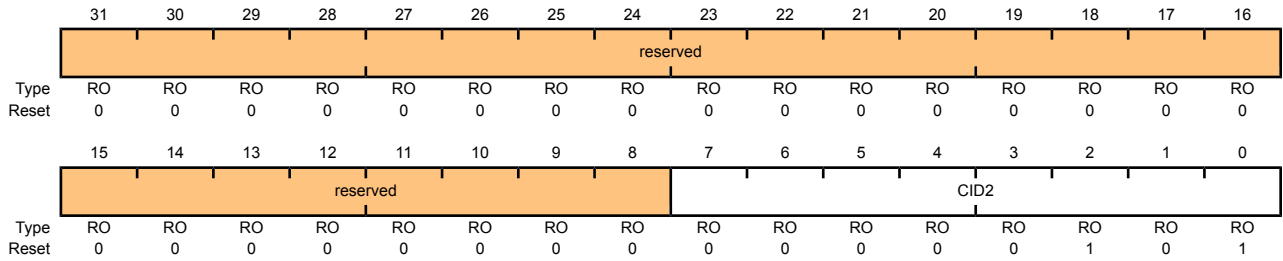
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	UART PrimeCell ID Register[15:8] Provides software a standard cross-peripheral identification system.

### Register 25: UART PrimeCell Identification 2 (UARTPCelIID2), offset 0xFF8

The **UARTPCelIIDn** registers are hard-coded and the fields within the registers determine the reset values.

#### UART PrimeCell Identification 2 (UARTPCelIID2)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0xFF8  
 Type RO, reset 0x0000.0005



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	UART PrimeCell ID Register[23:16] Provides software a standard cross-peripheral identification system.

## Register 26: UART PrimeCell Identification 3 (UARTPCelIID3), offset 0xFFC

The **UARTPCelIIDn** registers are hard-coded and the fields within the registers determine the reset values.

### UART PrimeCell Identification 3 (UARTPCelIID3)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0xFFC  
 Type RO, reset 0x0000.00B1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID3							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	UART PrimeCell ID Register[31:24] Provides software a standard cross-peripheral identification system.

## 15 Synchronous Serial Interface (SSI)

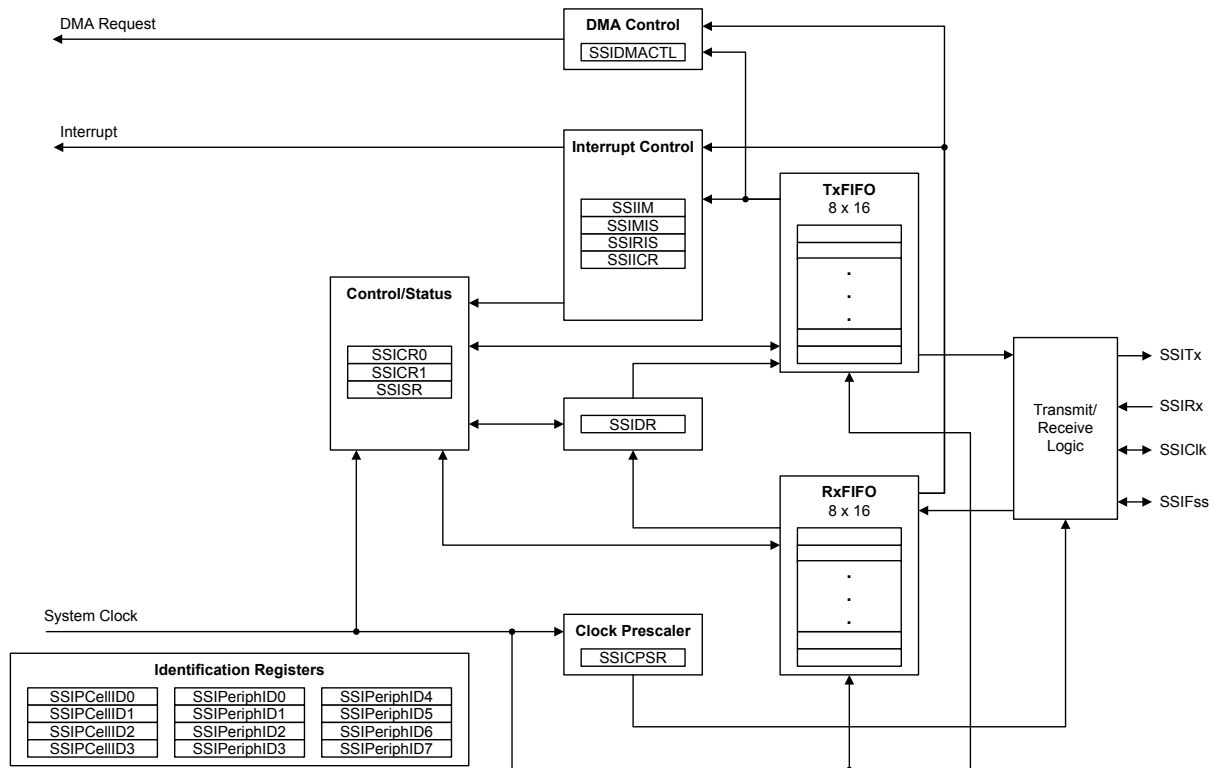
The Stellaris<sup>®</sup> Synchronous Serial Interface (SSI) is a master or slave interface for synchronous serial communication with peripheral devices that have either Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces.

The Stellaris<sup>®</sup> SSI module has the following features:

- Master or slave operation
- Support for Direct Memory Access (DMA)
- Programmable clock bit rate and prescale
- Separate transmit and receive FIFOs, 16 bits wide, 8 locations deep
- Programmable interface operation for Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces
- Programmable data frame size from 4 to 16 bits
- Internal loopback test mode for diagnostic/debug testing

### 15.1 Block Diagram

Figure 15-1. SSI Module Block Diagram



## 15.2 Functional Description

The SSI performs serial-to-parallel conversion on data received from a peripheral device. The CPU accesses data, control, and status information. The transmit and receive paths are buffered with internal FIFO memories allowing up to eight 16-bit values to be stored independently in both transmit and receive modes. The SSI also supports the DMA interface. The transmit and receive FIFOs can be programmed as destination/source addresses in the DMA module. DMA operation is enabled by setting the appropriate bit(s) in the **SSIDMACTL** register (see page 440).

### 15.2.1 Bit Rate Generation

The SSI includes a programmable bit rate clock divider and prescaler to generate the serial output clock. Bit rates are supported to MHz and higher, although maximum bit rate is determined by peripheral devices.

The serial bit rate is derived by dividing down the input clock (FSysClk). The clock is first divided by an even prescale value CPDVSr from 2 to 254, which is programmed in the **SSI Clock Prescale (SSICPSR)** register (see page 434). The clock is further divided by a value from 1 to 256, which is  $1 + SCR$ , where SCR is the value programmed in the **SSI Control0 (SSICR0)** register (see page 427).

The frequency of the output clock SSIClk is defined by:

$$SSIClk = F_{SysClk} / (CPDVSr * (1 + SCR))$$

**Note:** Although the SSIClk transmit clock can theoretically be 25 MHz, the module may not be able to operate at that speed. For master mode, the system clock must be at least two times faster than the SSIClk. For slave mode, the system clock must be at least 12 times faster than the SSIClk.

See “Synchronous Serial Interface (SSI)” on page 506 to view SSI timing parameters.

### 15.2.2 FIFO Operation

#### 15.2.2.1 Transmit FIFO

The common transmit FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. The CPU writes data to the FIFO by writing the **SSI Data (SSIDR)** register (see page 431), and data is stored in the FIFO until it is read out by the transmission logic.

When configured as a master or a slave, parallel data is written into the transmit FIFO prior to serial conversion and transmission to the attached slave or master, respectively, through the SSITx pin.

#### 15.2.2.2 Receive FIFO

The common receive FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. Received data from the serial interface is stored in the buffer until read out by the CPU, which accesses the read FIFO by reading the **SSIDR** register.

When configured as a master or slave, serial data received through the SSIRx pin is registered prior to parallel loading into the attached slave or master receive FIFO, respectively.

### 15.2.3 Interrupts

The SSI can generate interrupts when the following conditions are observed:

- Transmit FIFO service
- Receive FIFO service

- Receive FIFO time-out
- Receive FIFO overrun

All of the interrupt events are ORed together before being sent to the interrupt controller, so the SSI can only generate a single interrupt request to the controller at any given time. You can mask each of the four individual maskable interrupts by setting the appropriate bits in the **SSI Interrupt Mask (SSIM)** register (see page 435). Setting the appropriate mask bit to 1 enables the interrupt.

Provision of the individual outputs, as well as a combined interrupt output, allows use of either a global interrupt service routine, or modular device drivers to handle interrupts. The transmit and receive dynamic dataflow interrupts have been separated from the status interrupts so that data can be read or written in response to the FIFO trigger levels. The status of the individual interrupt sources can be read from the **SSI Raw Interrupt Status (SSIRIS)** and **SSI Masked Interrupt Status (SSIMIS)** registers (see page 437 and page 438, respectively).

### 15.2.4 Frame Formats

Each data frame is between 4 and 16 bits long, depending on the size of data programmed, and is transmitted starting with the MSB. There are three basic frame types that can be selected:

- Texas Instruments synchronous serial
- Freescale SPI
- MICROWIRE

For all three formats, the serial clock (**SSIClk**) is held inactive while the SSI is idle, and **SSIClk** transitions at the programmed frequency only during active transmission or reception of data. The idle state of **SSIClk** is utilized to provide a receive timeout indication that occurs when the receive FIFO still contains data after a timeout period.

For Freescale SPI and MICROWIRE frame formats, the serial frame (**SSIFSS**) pin is active Low, and is asserted (pulled down) during the entire transmission of the frame.

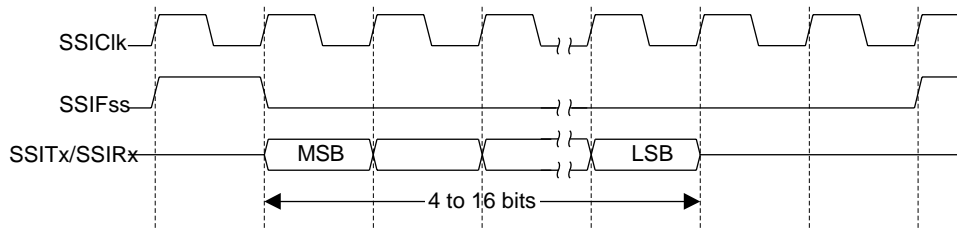
For Texas Instruments synchronous serial frame format, the **SSIFSS** pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the SSI and the off-chip slave device drive their output data on the rising edge of **SSIClk**, and latch data from the other device on the falling edge.

Unlike the full-duplex transmission of the other two frame formats, the MICROWIRE format uses a special master-slave messaging technique, which operates at half-duplex. In this mode, when a frame begins, an 8-bit control message is transmitted to the off-chip slave. During this transmit, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the requested data. The returned data can be 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

#### 15.2.4.1 Texas Instruments Synchronous Serial Frame Format

Figure 15-2 on page 417 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

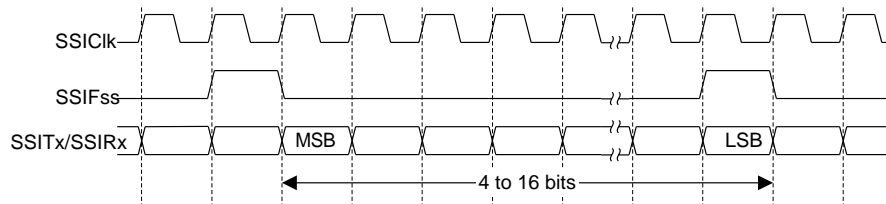


**Figure 15-2. TI Synchronous Serial Frame Format (Single Transfer)**

In this mode, SSIClk and SSIFss are forced Low, and the transmit data line SSITx is tristated whenever the SSI is idle. Once the bottom entry of the transmit FIFO contains data, SSIFss is pulsed High for one SSIClk period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of SSIClk, the MSB of the 4 to 16-bit data frame is shifted out on the SSITx pin. Likewise, the MSB of the received data is shifted onto the SSIRx pin by the off-chip serial slave device.

Both the SSI and the off-chip serial slave device then clock each data bit into their serial shifter on the falling edge of each SSIClk. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of SSIClk after the LSB has been latched.

Figure 15-3 on page 417 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

**Figure 15-3. TI Synchronous Serial Frame Format (Continuous Transfer)**

### 15.2.4.2 Freescale SPI Frame Format

The Freescale SPI interface is a four-wire interface where the SSIFss signal behaves as a slave select. The main feature of the Freescale SPI format is that the inactive state and phase of the SSIClk signal are programmable through the SPO and SPH bits within the SSISCR0 control register.

#### **SPO Clock Polarity Bit**

When the SPO clock polarity control bit is Low, it produces a steady state Low value on the SSIClk pin. If the SPO bit is High, a steady state High value is placed on the SSIClk pin when data is not being transferred.

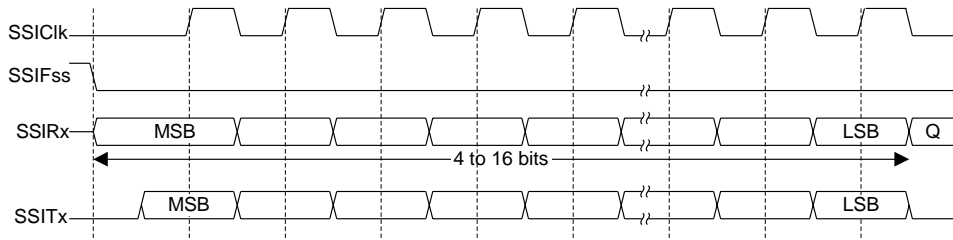
#### **SPH Phase Control Bit**

The SPH phase control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the SPH phase control bit is Low, data is captured on the first clock edge transition. If the SPH bit is High, data is captured on the second clock edge transition.

### 15.2.4.3 Freescale SPI Frame Format with SPO=0 and SPH=0

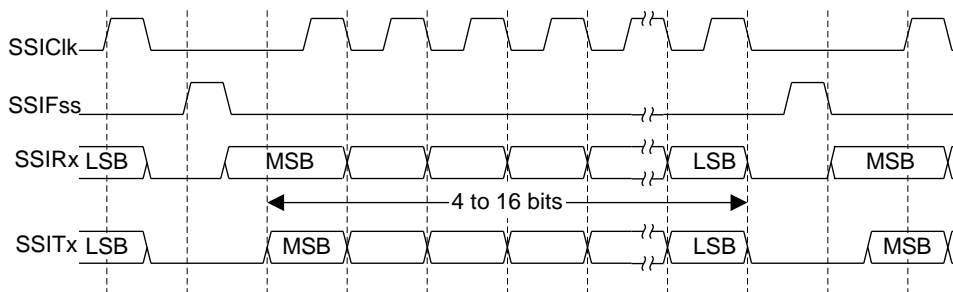
Single and continuous transmission signal sequences for Freescale SPI format with SPO=0 and SPH=0 are shown in Figure 15-4 on page 418 and Figure 15-5 on page 418.

**Figure 15-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0**



**Note:** Q is undefined.

**Figure 15-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0**



In this configuration, during idle periods:

- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low. This causes slave data to be enabled onto the SSIRx input line of the master. The master SSITx output pad is enabled.

One half SSIClk period later, valid master data is transferred to the SSITx pin. Now that both the master and slave data have been set, the SSIClk master clock pin goes High after one further half SSIClk period.

The data is now captured on the rising and propagated on the falling edges of the SSIClk signal.

In the case of a single word transmission, after all bits of the data word have been transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

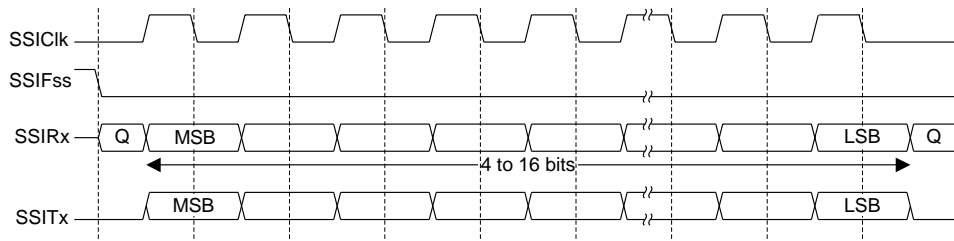
However, in the case of continuous back-to-back transmissions, the SSIFss signal must be pulsed High between each data word transfer. This is because the slave select pin freezes the data in its

serial peripheral register and does not allow it to be altered if the `SPH` bit is logic zero. Therefore, the master device must raise the `SSIFSS` pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the `SSIFSS` pin is returned to its idle state one `SSIClk` period after the last bit has been captured.

#### 15.2.4.4 Freescale SPI Frame Format with `SPO=0` and `SPH=1`

The transfer signal sequence for Freescale SPI format with `SPO=0` and `SPH=1` is shown in Figure 15-6 on page 419, which covers both single and continuous transfers.

**Figure 15-6. Freescale SPI Frame Format with `SPO=0` and `SPH=1`**



**Note:** Q is undefined.

In this configuration, during idle periods:

- `SSIClk` is forced Low
- `SSIFss` is forced High
- The transmit data line `SSITx` is arbitrarily forced Low
- When the SSI is configured as a master, it enables the `SSIClk` pad
- When the SSI is configured as a slave, it disables the `SSIClk` pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the `SSIFss` master signal being driven Low. The master `SSITx` output is enabled. After a further one half `SSIClk` period, both master and slave valid data is enabled onto their respective transmission lines. At the same time, the `SSIClk` is enabled with a rising edge transition.

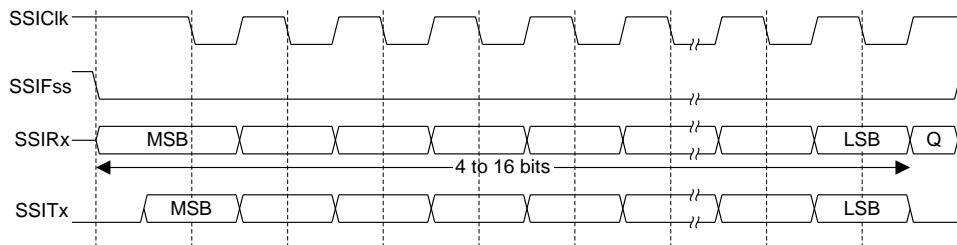
Data is then captured on the falling edges and propagated on the rising edges of the `SSIClk` signal.

In the case of a single word transfer, after all bits have been transferred, the `SSIFss` line is returned to its idle High state one `SSIClk` period after the last bit has been captured.

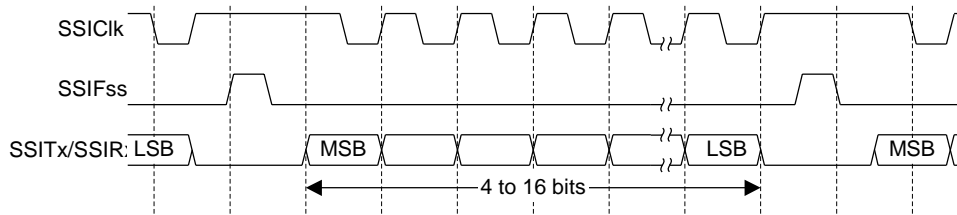
For continuous back-to-back transfers, the `SSIFss` pin is held Low between successive data words and termination is the same as that of the single word transfer.

#### 15.2.4.5 Freescale SPI Frame Format with `SPO=1` and `SPH=0`

Single and continuous transmission signal sequences for Freescale SPI format with `SPO=1` and `SPH=0` are shown in Figure 15-7 on page 420 and Figure 15-8 on page 420.

**Figure 15-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0**

**Note:** Q is undefined.

**Figure 15-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0**

In this configuration, during idle periods:

- SSIClk is forced High
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low, which causes slave data to be immediately transferred onto the SSIRx line of the master. The master SSITx output pad is enabled.

One half period later, valid master data is transferred to the SSITx line. Now that both the master and slave data have been set, the SSIClk master clock pin becomes Low after one further half SSIClk period. This means that data is captured on the falling edges and propagated on the rising edges of the SSIClk signal.

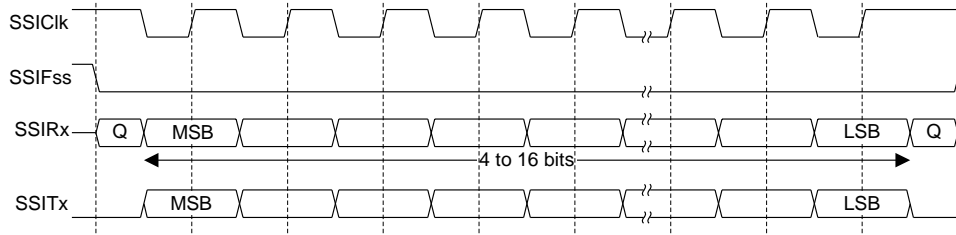
In the case of a single word transmission, after all bits of the data word are transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSIFss signal must be pulsed High between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is logic zero. Therefore, the master device must raise the SSIFss pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSIFss pin is returned to its idle state one SSIClk period after the last bit has been captured.

### 15.2.4.6 Freescale SPI Frame Format with SPO=1 and SPH=1

The transfer signal sequence for Freescale SPI format with SPO=1 and SPH=1 is shown in Figure 15-9 on page 421, which covers both single and continuous transfers.

**Figure 15-9. Freescale SPI Frame Format with SPO=1 and SPH=1**



**Note:** Q is undefined.

In this configuration, during idle periods:

- SSIClk is forced High
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low. The master SSITx output pad is enabled. After a further one-half SSIClk period, both master and slave data are enabled onto their respective transmission lines. At the same time, SSIClk is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the SSIClk signal.

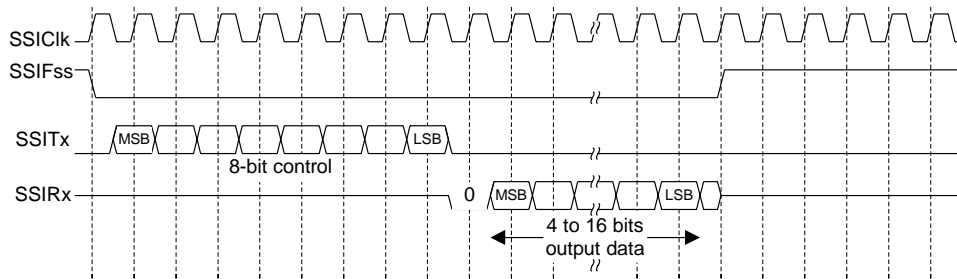
After all bits have been transferred, in the case of a single word transmission, the SSIFss line is returned to its idle high state one SSIClk period after the last bit has been captured.

For continuous back-to-back transmissions, the SSIFss pin remains in its active Low state, until the final bit of the last word has been captured, and then returns to its idle state as described above.

For continuous back-to-back transfers, the SSIFss pin is held Low between successive data words and termination is the same as that of the single word transfer.

### 15.2.4.7 MICROWIRE Frame Format

Figure 15-10 on page 422 shows the MICROWIRE frame format, again for a single frame. Figure 15-11 on page 423 shows the same format when back-to-back frames are transmitted.

**Figure 15-10. MICROWIRE Frame Format (Single Frame)**

MICROWIRE format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex, using a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SSI to the off-chip slave device. During this transmission, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the required data. The returned data is 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

In this configuration, during idle periods:

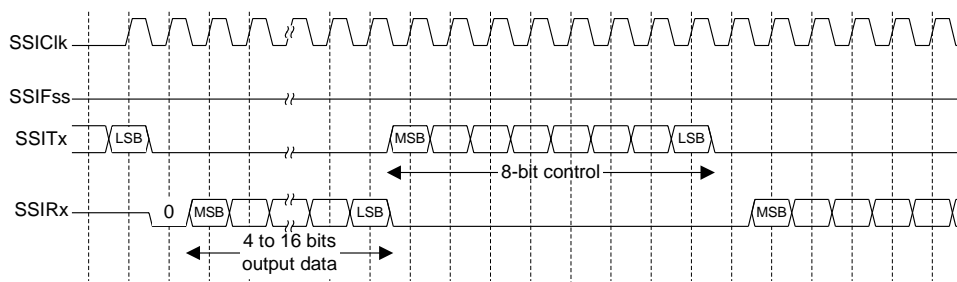
- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low

A transmission is triggered by writing a control byte to the transmit FIFO. The falling edge of SSIFss causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic, and the MSB of the 8-bit control frame to be shifted out onto the SSITx pin. SSIFss remains Low for the duration of the frame transmission. The SSIRx pin remains tristated during this transmission.

The off-chip serial slave device latches each control bit into its serial shifter on the rising edge of each SSIClk. After the last bit is latched by the slave device, the control byte is decoded during a one clock wait-state, and the slave responds by transmitting data back to the SSI. Each bit is driven onto the SSIRx line on the falling edge of SSIClk. The SSI in turn latches each bit on the rising edge of SSIClk. At the end of the frame, for single transfers, the SSIFss signal is pulled High one clock period after the last bit has been latched in the receive serial shifter, which causes the data to be transferred to the receive FIFO.

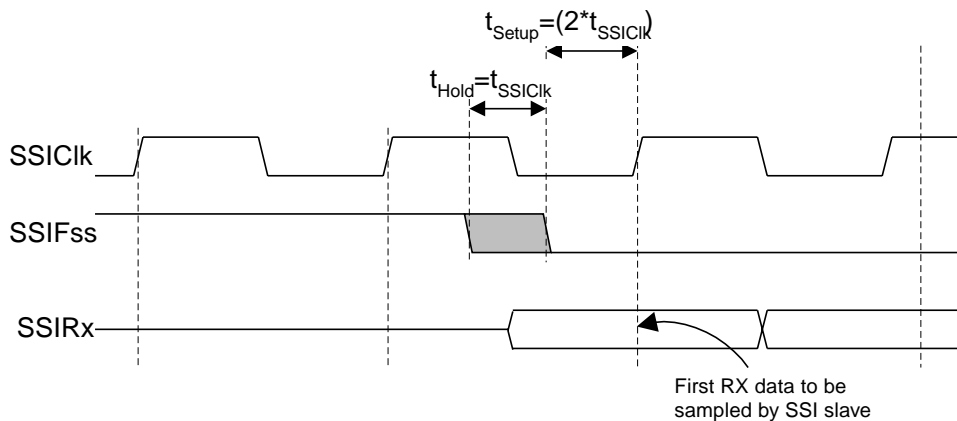
**Note:** The off-chip slave device can tristate the receive line either on the falling edge of SSIClk after the LSB has been latched by the receive shifter, or when the SSIFss pin goes High.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the SSIFss line is continuously asserted (held Low) and transmission of data occurs back-to-back. The control byte of the next frame follows directly after the LSB of the received data from the current frame. Each of the received values is transferred from the receive shifter on the falling edge of SSIClk, after the LSB of the frame has been latched into the SSI.

**Figure 15-11. MICROWIRE Frame Format (Continuous Transfer)**

In the MICROWIRE mode, the SSI slave samples the first bit of receive data on the rising edge of SSIClk after SSIFss has gone Low. Masters that drive a free-running SSIClk must ensure that the SSIFss signal has sufficient setup and hold margins with respect to the rising edge of SSIClk.

Figure 15-12 on page 423 illustrates these setup and hold time requirements. With respect to the SSIClk rising edge on which the first bit of receive data is to be sampled by the SSI slave, SSIFss must have a setup of at least two times the period of SSIClk on which the SSI operates. With respect to the SSIClk rising edge previous to this edge, SSIFss must have a hold of at least one SSIClk period.

**Figure 15-12. MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements**

### 15.2.5 DMA Operation

The SSI peripheral provides an interface connected to the  $\mu$ DMA controller. The DMA operation of the SSI is enabled through the **SSI DMA Control (SSIDMACTL)** register. When DMA operation is enabled, the SSI will assert a DMA request on the receive or transmit channel when the associated FIFO can transfer data. For the receive channel, a single transfer request is asserted whenever there is any data in the receive FIFO. A burst transfer request is asserted whenever the amount of data in the receive FIFO is 4 or more items. For the transmit channel, a single transfer request is asserted whenever there is at least one empty location in the transmit FIFO. The burst request is asserted whenever the transmit FIFO has 4 or more empty slots. The single and burst DMA transfer requests are handled automatically by the  $\mu$ DMA controller depending how the DMA channel is configured. To enable DMA operation for the receive channel, the **RXDMAE** bit of the **DMA Control (SSIDMACTL)** register should be set. To enable DMA operation for the transmit channel, the **TXDMAE** bit of **SSIDMACTL** should be set. If DMA is enabled, then the  $\mu$ DMA controller will trigger an interrupt when a transfer is complete. The interrupt will occur on the SSI interrupt vector. Therefore, if interrupts

are used for SSI operation and DMA is enabled, the SSI interrupt handler must be designed to handle the  $\mu$ DMA completion interrupt.

See “Micro Direct Memory Access ( $\mu$ DMA)” on page 176 for more details about programming the  $\mu$ DMA controller.

## 15.3 Initialization and Configuration

To use the SSI, its peripheral clock must be enabled by setting the *SSI* bit in the **RCGC1** register.

For each of the frame formats, the SSI is configured using the following steps:

1. Ensure that the *SSE* bit in the **SSICR1** register is disabled before making any configuration changes.
2. Select whether the SSI is a master or slave:
  - a. For master operations, set the **SSICR1** register to 0x0000.0000.
  - b. For slave mode (output enabled), set the **SSICR1** register to 0x0000.0004.
  - c. For slave mode (output disabled), set the **SSICR1** register to 0x0000.000C.
3. Configure the clock prescale divisor by writing the **SSICPSR** register.
4. Write the **SSICR0** register with the following configuration:
  - Serial clock rate (*SCR*)
  - Desired clock phase/polarity, if using Freescale SPI mode (*SPH* and *SPO*)
  - The protocol mode: Freescale SPI, TI SSF, MICROWIRE (*FRF*)
  - The data size (*DSS*)
5. Optionally, configure the  $\mu$ DMA channel (see “Micro Direct Memory Access ( $\mu$ DMA)” on page 176) and enable the DMA option(s) in the **SSIDMACTL** register.
6. Enable the SSI by setting the *SSE* bit in the **SSICR1** register.

As an example, assume the SSI must be configured to operate with the following parameters:

- Master operation
- Freescale SPI mode (*SPO*=1, *SPH*=1)
- 1 Mbps bit rate
- 8 data bits

Assuming the system clock is 20 MHz, the bit rate calculation would be:

$$F_{SSIClk} = F_{SysClk} / (CPSDVSR * (1 + SCR))$$

$$1 \times 10^6 = 20 \times 10^6 / (CPSDVSR * (1 + SCR))$$

In this case, if *CPSDVSR*=2, *SCR* must be 9.



The configuration sequence would be as follows:

1. Ensure that the `SSE` bit in the **SSICR1** register is disabled.
2. Write the **SSICR1** register with a value of `0x0000.0000`.
3. Write the **SSICPSR** register with a value of `0x0000.0002`.
4. Write the **SSICR0** register with a value of `0x0000.09C7`.
5. The SSI is then enabled by setting the `SSE` bit in the **SSICR1** register to 1.

## 15.4 Register Map

Table 15-1 on page 425 lists the SSI registers. The offset listed is a hexadecimal increment to the register's address, relative to that SSI module's base address:

- SSI0: `0x4000.8000`

**Note:** The SSI must be disabled (see the `SSE` bit in the **SSICR1** register) before any of the control registers are reprogrammed.

**Table 15-1. SSI Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	SSICR0	R/W	0x0000.0000	SSI Control 0	427
0x004	SSICR1	R/W	0x0000.0000	SSI Control 1	429
0x008	SSIDR	R/W	0x0000.0000	SSI Data	431
0x00C	SSISR	RO	0x0000.0003	SSI Status	432
0x010	SSICPSR	R/W	0x0000.0000	SSI Clock Prescale	434
0x014	SSIIM	R/W	0x0000.0000	SSI Interrupt Mask	435
0x018	SSIRIS	RO	0x0000.0008	SSI Raw Interrupt Status	437
0x01C	SSIMIS	RO	0x0000.0000	SSI Masked Interrupt Status	438
0x020	SSIICR	W1C	0x0000.0000	SSI Interrupt Clear	439
0x024	SSIDMACTL	R/W	0x0000.0000	SSI DMA Control	440
0xFD0	SSIPeriphID4	RO	0x0000.0000	SSI Peripheral Identification 4	441
0xFD4	SSIPeriphID5	RO	0x0000.0000	SSI Peripheral Identification 5	442
0xFD8	SSIPeriphID6	RO	0x0000.0000	SSI Peripheral Identification 6	443
0xFDC	SSIPeriphID7	RO	0x0000.0000	SSI Peripheral Identification 7	444
0xFE0	SSIPeriphID0	RO	0x0000.0022	SSI Peripheral Identification 0	445
0xFE4	SSIPeriphID1	RO	0x0000.0000	SSI Peripheral Identification 1	446
0xFE8	SSIPeriphID2	RO	0x0000.0018	SSI Peripheral Identification 2	447
0xFEC	SSIPeriphID3	RO	0x0000.0001	SSI Peripheral Identification 3	448

Offset	Name	Type	Reset	Description	See page
0xFF0	SSIPCellID0	RO	0x0000.000D	SSI PrimeCell Identification 0	449
0xFF4	SSIPCellID1	RO	0x0000.00F0	SSI PrimeCell Identification 1	450
0xFF8	SSIPCellID2	RO	0x0000.0005	SSI PrimeCell Identification 2	451
0xFFC	SSIPCellID3	RO	0x0000.00B1	SSI PrimeCell Identification 3	452

## 15.5 Register Descriptions

The remainder of this section lists and describes the SSI registers, in numerical order by address offset.

## Register 1: SSI Control 0 (SSICR0), offset 0x000

**SSICR0** is control register 0 and contains bit fields that control various functions within the SSI module. Functionality such as protocol mode, clock rate, and data size are configured in this register.

### SSI Control 0 (SSICR0)

SSI0 base: 0x4000.8000  
Offset 0x000  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SCR								SPH		SPO		FRF		DSS	
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	SCR	R/W	0x0000	SSI Serial Clock Rate  The value <b>SCR</b> is used to generate the transmit and receive bit rate of the SSI. The bit rate is:  $BR = FSSIClk / (CPSDVSR * (1 + SCR))$ where <b>CPSDVSR</b> is an even value from 2-254 programmed in the <b>SSICPSR</b> register, and <b>SCR</b> is a value from 0-255.
7	SPH	R/W	0	SSI Serial Clock Phase  This bit is only applicable to the Freescale SPI Format.  The <b>SPH</b> control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge.  When the <b>SPH</b> bit is 0, data is captured on the first clock edge transition. If <b>SPH</b> is 1, data is captured on the second clock edge transition.
6	SPO	R/W	0	SSI Serial Clock Polarity  This bit is only applicable to the Freescale SPI Format.  When the <b>SPO</b> bit is 0, it produces a steady state Low value on the <b>SSIClk</b> pin. If <b>SPO</b> is 1, a steady state High value is placed on the <b>SSIClk</b> pin when data is not being transferred.

Bit/Field	Name	Type	Reset	Description
5:4	FRF	R/W	0x0	SSI Frame Format Select The FRF values are defined as follows:  Value Frame Format 0x0 Freescale SPI Frame Format 0x1 Texas Instruments Synchronous Serial Frame Format 0x2 MICROWIRE Frame Format 0x3 Reserved
3:0	DSS	R/W	0x00	SSI Data Size Select The DSS values are defined as follows:  Value Data Size 0x0-0x2 Reserved 0x3 4-bit data 0x4 5-bit data 0x5 6-bit data 0x6 7-bit data 0x7 8-bit data 0x8 9-bit data 0x9 10-bit data 0xA 11-bit data 0xB 12-bit data 0xC 13-bit data 0xD 14-bit data 0xE 15-bit data 0xF 16-bit data

## Register 2: SSI Control 1 (SSICR1), offset 0x004

**SSICR1** is control register 1 and contains bit fields that control various functions within the SSI module. Master and slave mode functionality is controlled by this register.

### SSI Control 1 (SSICR1)

SSI0 base: 0x4000.8000  
Offset 0x004  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved												SOD	MS	SSE	LBM	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

3	SOD	R/W	0	SSI Slave Mode Output Disable  This bit is relevant only in the Slave mode ( $MS=1$ ). In multiple-slave systems, it is possible for the SSI master to broadcast a message to all slaves in the system while ensuring that only one slave drives data onto the serial output line. In such systems, the TXD lines from multiple slaves could be tied together. To operate in such a system, the SOD bit can be configured so that the SSI slave does not drive the SSITx pin.
---	-----	-----	---	---

The SOD values are defined as follows:

Value	Description
0	SSI can drive SSITx output in Slave Output mode.
1	SSI must not drive the SSITx output in Slave mode.

2	MS	R/W	0	SSI Master/Slave Select  This bit selects Master or Slave mode and can be modified only when SSI is disabled ( $SSE=0$ ).
---	----	-----	---	---

The MS values are defined as follows:

Value	Description
0	Device configured as a master.
1	Device configured as a slave.

Bit/Field	Name	Type	Reset	Description						
1	SSE	R/W	0	<p>SSI Synchronous Serial Port Enable</p> <p>Setting this bit enables SSI operation.</p> <p>The <code>SSE</code> values are defined as follows:</p> <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0</td><td>SSI operation disabled.</td></tr><tr><td>1</td><td>SSI operation enabled.</td></tr></tbody></table> <p><b>Note:</b> This bit must be set to 0 before any control registers are reprogrammed.</p>	Value	Description	0	SSI operation disabled.	1	SSI operation enabled.
Value	Description									
0	SSI operation disabled.									
1	SSI operation enabled.									
0	LBM	R/W	0	<p>SSI Loopback Mode</p> <p>Setting this bit enables Loopback Test mode.</p> <p>The <code>LBM</code> values are defined as follows:</p> <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0</td><td>Normal serial port operation enabled.</td></tr><tr><td>1</td><td>Output of the transmit serial shift register is connected internally to the input of the receive serial shift register.</td></tr></tbody></table>	Value	Description	0	Normal serial port operation enabled.	1	Output of the transmit serial shift register is connected internally to the input of the receive serial shift register.
Value	Description									
0	Normal serial port operation enabled.									
1	Output of the transmit serial shift register is connected internally to the input of the receive serial shift register.									

### Register 3: SSI Data (SSIDR), offset 0x008

**SSIDR** is the data register and is 16-bits wide. When **SSIDR** is read, the entry in the receive FIFO (pointed to by the current FIFO read pointer) is accessed. As data values are removed by the SSI receive logic from the incoming data frame, they are placed into the entry in the receive FIFO (pointed to by the current FIFO write pointer).

When **SSIDR** is written to, the entry in the transmit FIFO (pointed to by the write pointer) is written to. Data values are removed from the transmit FIFO one value at a time by the transmit logic. It is loaded into the transmit serial shifter, then serially shifted out onto the **SSITx** pin at the programmed bit rate.

When a data size of less than 16 bits is selected, the user must right-justify data written to the transmit FIFO. The transmit logic ignores the unused bits. Received data less than 16 bits is automatically right-justified in the receive buffer.

When the SSI is programmed for MICROWIRE frame format, the default size for transmit data is eight bits (the most significant byte is ignored). The receive data size is controlled by the programmer. The transmit FIFO and the receive FIFO are not cleared even when the **SSE** bit in the **SSICR1** register is set to zero. This allows the software to fill the transmit FIFO before enabling the SSI.

#### SSI Data (SSIDR)

SSI0 base: 0x4000.8000  
Offset 0x008  
Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DATA	R/W	0x0000	SSI Receive/Transmit Data  A read operation reads the receive FIFO. A write operation writes the transmit FIFO.  Software must right-justify data when the SSI is programmed for a data size that is less than 16 bits. Unused bits at the top are ignored by the transmit logic. The receive logic automatically right-justifies the data.

**Register 4: SSI Status (SSISR), offset 0x00C**

**SSISR** is a status register that contains bits that indicate the FIFO fill status and the SSI busy status.

**SSI Status (SSISR)**

SSI0 base: 0x4000.8000

Offset 0x00C

Type RO, reset 0x0000.0003

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved												BSY	RFF	RNE	TNF	TFE
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	

Bit/Field	Name	Type	Reset	Description						
31:5	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
4	BSY	RO	0	SSI Busy Bit  The <b>BSY</b> values are defined as follows:  <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>SSI is idle.</td> </tr> <tr> <td>1</td> <td>SSI is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty.</td> </tr> </tbody> </table>	Value	Description	0	SSI is idle.	1	SSI is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty.
Value	Description									
0	SSI is idle.									
1	SSI is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty.									
3	RFF	RO	0	SSI Receive FIFO Full  The <b>RFF</b> values are defined as follows:  <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Receive FIFO is not full.</td> </tr> <tr> <td>1</td> <td>Receive FIFO is full.</td> </tr> </tbody> </table>	Value	Description	0	Receive FIFO is not full.	1	Receive FIFO is full.
Value	Description									
0	Receive FIFO is not full.									
1	Receive FIFO is full.									
2	RNE	RO	0	SSI Receive FIFO Not Empty  The <b>RNE</b> values are defined as follows:  <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Receive FIFO is empty.</td> </tr> <tr> <td>1</td> <td>Receive FIFO is not empty.</td> </tr> </tbody> </table>	Value	Description	0	Receive FIFO is empty.	1	Receive FIFO is not empty.
Value	Description									
0	Receive FIFO is empty.									
1	Receive FIFO is not empty.									
1	TNF	RO	1	SSI Transmit FIFO Not Full  The <b>TNF</b> values are defined as follows:  <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Transmit FIFO is full.</td> </tr> <tr> <td>1</td> <td>Transmit FIFO is not full.</td> </tr> </tbody> </table>	Value	Description	0	Transmit FIFO is full.	1	Transmit FIFO is not full.
Value	Description									
0	Transmit FIFO is full.									
1	Transmit FIFO is not full.									



---

Bit/Field	Name	Type	Reset	Description
0	TFE	R0	1	SSI Transmit FIFO Empty The TFE values are defined as follows:  Value Description 0 Transmit FIFO is not empty. 1 Transmit FIFO is empty.

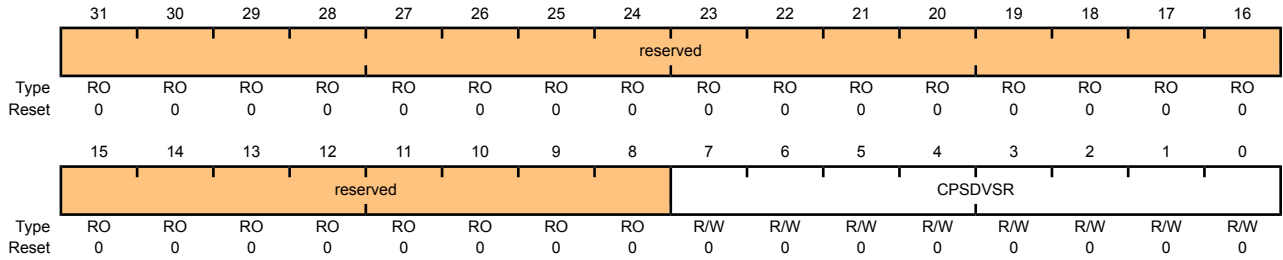
### Register 5: SSI Clock Prescale (SSICPSR), offset 0x010

**SSICPSR** is the clock prescale register and specifies the division factor by which the system clock must be internally divided before further use.

The value programmed into this register must be an even number between 2 and 254. The least-significant bit of the programmed number is hard-coded to zero. If an odd number is written to this register, data read back from this register has the least-significant bit as zero.

#### SSI Clock Prescale (SSICPSR)

SSI0 base: 0x4000.8000  
 Offset 0x010  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CPSDVSR	R/W	0x00	SSI Clock Prescale Divisor  This value must be an even number from 2 to 254, depending on the frequency of SSI0CLK. The LSB always returns 0 on reads.

## Register 6: SSI Interrupt Mask (SSIIM), offset 0x014

The **SSIIM** register is the interrupt mask set or clear register. It is a read/write register and all bits are cleared to 0 on reset.

On a read, this register gives the current value of the mask on the relevant interrupt. A write of 1 to the particular bit sets the mask, enabling the interrupt to be read. A write of 0 clears the corresponding mask.

### SSI Interrupt Mask (SSIIM)

SSI0 base: 0x4000.8000  
Offset 0x014  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved												TXIM	RXIM	RTIM	RORIM	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXIM	R/W	0	SSI Transmit FIFO Interrupt Mask  The <b>TXIM</b> values are defined as follows:  Value Description 0 TX FIFO half-full or less condition interrupt is masked. 1 TX FIFO half-full or less condition interrupt is not masked.
2	RXIM	R/W	0	SSI Receive FIFO Interrupt Mask  The <b>RXIM</b> values are defined as follows:  Value Description 0 RX FIFO half-full or more condition interrupt is masked. 1 RX FIFO half-full or more condition interrupt is not masked.
1	RTIM	R/W	0	SSI Receive Time-Out Interrupt Mask  The <b>RTIM</b> values are defined as follows:  Value Description 0 RX FIFO time-out interrupt is masked. 1 RX FIFO time-out interrupt is not masked.

Bit/Field	Name	Type	Reset	Description
0	RORIM	R/W	0	SSI Receive Overrun Interrupt Mask The RORIM values are defined as follows:  Value Description 0 RX FIFO overrun interrupt is masked. 1 RX FIFO overrun interrupt is not masked.

## Register 7: SSI Raw Interrupt Status (SSIRIS), offset 0x018

The **SSIRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect.

### SSI Raw Interrupt Status (SSIRIS)

SSI0 base: 0x4000.8000  
Offset 0x018  
Type RO, reset 0x0000.0008

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												TXRIS	RXRIS	RTRIS	RORRIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXRIS	RO	1	SSI Transmit FIFO Raw Interrupt Status Indicates that the transmit FIFO is half full or less, when set.
2	RXRIS	RO	0	SSI Receive FIFO Raw Interrupt Status Indicates that the receive FIFO is half full or more, when set.
1	RTRIS	RO	0	SSI Receive Time-Out Raw Interrupt Status Indicates that the receive time-out has occurred, when set.
0	RORRIS	RO	0	SSI Receive Overrun Raw Interrupt Status Indicates that the receive FIFO has overflowed, when set.

### Register 8: SSI Masked Interrupt Status (SSIMIS), offset 0x01C

The **SSIMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

#### SSI Masked Interrupt Status (SSIMIS)

SSI0 base: 0x4000.8000  
 Offset 0x01C  
 Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved													TXMIS	RXMIS	RTMIS	RORMIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXMIS	RO	0	SSI Transmit FIFO Masked Interrupt Status Indicates that the transmit FIFO is half full or less, when set.
2	RXMIS	RO	0	SSI Receive FIFO Masked Interrupt Status Indicates that the receive FIFO is half full or more, when set.
1	RTMIS	RO	0	SSI Receive Time-Out Masked Interrupt Status Indicates that the receive time-out has occurred, when set.
0	RORMIS	RO	0	SSI Receive Overrun Masked Interrupt Status Indicates that the receive FIFO has overflowed, when set.

## Register 9: SSI Interrupt Clear (SSIICR), offset 0x020

The **SSIICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

### SSI Interrupt Clear (SSIICR)

SSI0 base: 0x4000.8000

Offset 0x020

Type W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved														RTIC	RORIC	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

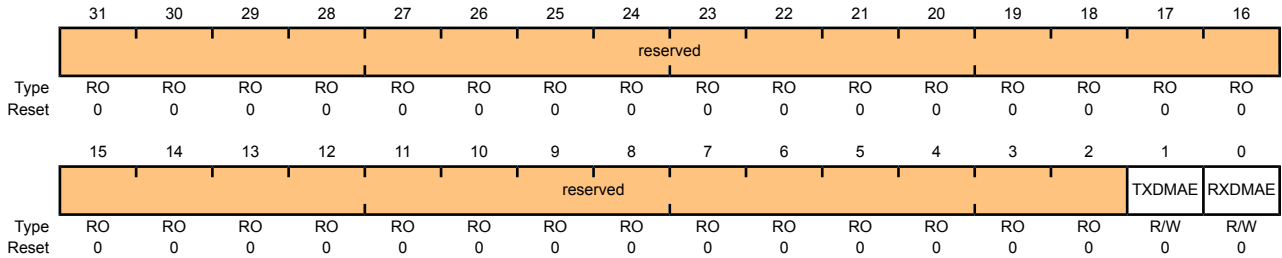
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	RTIC	W1C	0	SSI Receive Time-Out Interrupt Clear The <b>RTIC</b> values are defined as follows:  Value Description 0 No effect on interrupt. 1 Clears interrupt.
0	RORIC	W1C	0	SSI Receive Overrun Interrupt Clear The <b>RORIC</b> values are defined as follows:  Value Description 0 No effect on interrupt. 1 Clears interrupt.

### Register 10: SSI DMA Control (SSIDMACTL), offset 0x024

The **SSIDMACTL** register is the DMA control register.

#### SSI DMA Control (SSIDMACTL)

SSI0 base: 0x4000.8000  
 Offset 0x024  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	TXDMAE	R/W	0	Transmit DMA Enable If this bit is set to 1, DMA for the transmit FIFO is enabled.
0	RXDMAE	R/W	0	Receive DMA Enable If this bit is set to 1, DMA for the receive FIFO is enabled.



## Register 11: SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### SSI Peripheral Identification 4 (SSIPeriphID4)

SSI0 base: 0x4000.8000

Offset 0xFD0

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

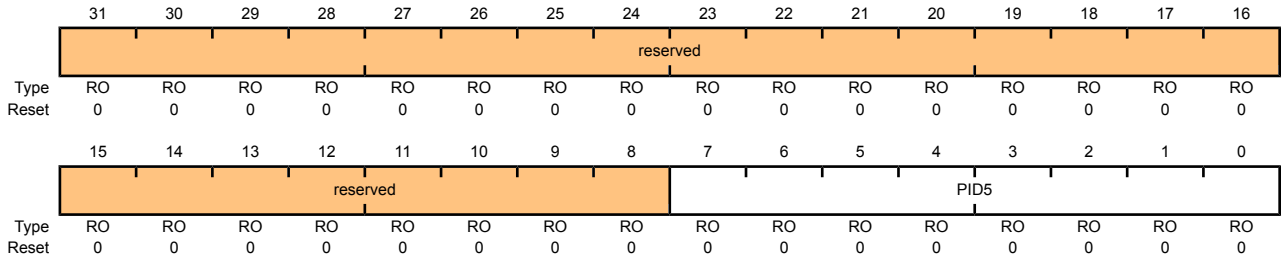
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	SSI Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

### Register 12: SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

#### SSI Peripheral Identification 5 (SSIPeriphID5)

SSI0 base: 0x4000.8000  
 Offset 0xFD4  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	SSI Peripheral ID Register[15:8] Can be used by software to identify the presence of this peripheral.

### Register 13: SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

#### SSI Peripheral Identification 6 (SSIPeriphID6)

SSI0 base: 0x4000.8000

Offset 0xFD8

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID6							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

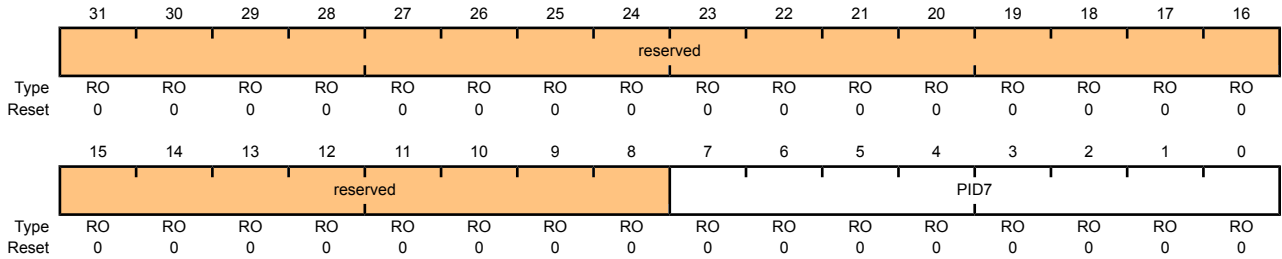
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	SSI Peripheral ID Register[23:16] Can be used by software to identify the presence of this peripheral.

### Register 14: SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

#### SSI Peripheral Identification 7 (SSIPeriphID7)

SSI0 base: 0x4000.8000  
 Offset 0xFDC  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	SSI Peripheral ID Register[31:24] Can be used by software to identify the presence of this peripheral.

## Register 15: SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### SSI Peripheral Identification 0 (SSIPeriphID0)

SSI0 base: 0x4000.8000

Offset 0xFE0

Type RO, reset 0x0000.0022

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0

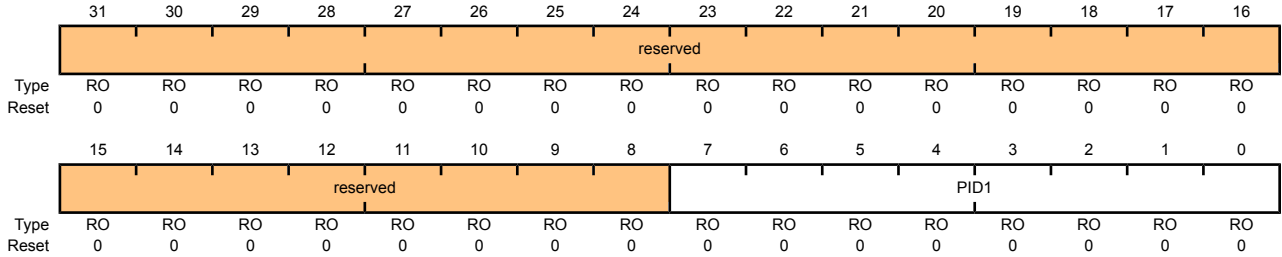
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x22	SSI Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

### Register 16: SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

#### SSI Peripheral Identification 1 (SSIPeriphID1)

SSI0 base: 0x4000.8000  
 Offset 0xFE4  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	SSI Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

## Register 17: SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### SSI Peripheral Identification 2 (SSIPeriphID2)

SSI0 base: 0x4000.8000

Offset 0xFE8

Type RO, reset 0x0000.0018

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

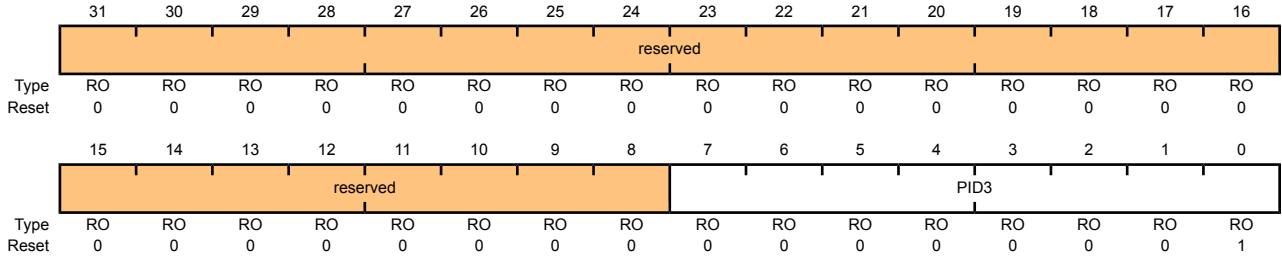
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	SSI Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

### Register 18: SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

#### SSI Peripheral Identification 3 (SSIPeriphID3)

SSI0 base: 0x4000.8000  
 Offset 0xFEC  
 Type RO, reset 0x0000.0001



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	SSI Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.



## Register 19: SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0

The **SSIPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

### SSI PrimeCell Identification 0 (SSIPCellID0)

SSI0 base: 0x4000.8000  
Offset 0xFF0  
Type RO, reset 0x0000.000D

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

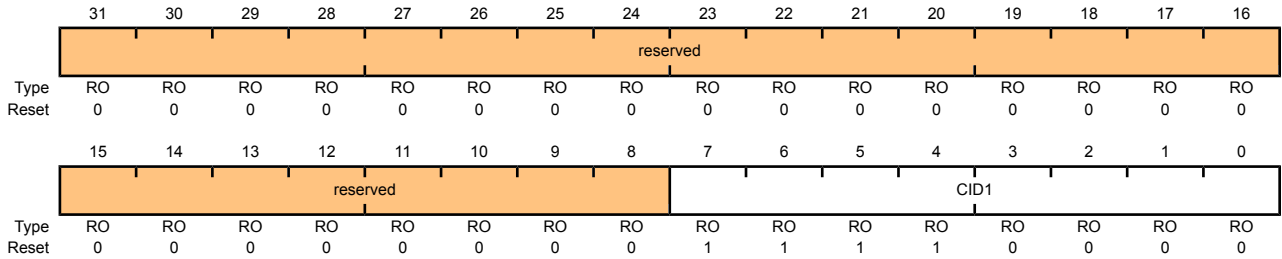
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	SSI PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

### Register 20: SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4

The **SSIPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

#### SSI PrimeCell Identification 1 (SSIPCellID1)

SSI0 base: 0x4000.8000  
 Offset 0xFF4  
 Type RO, reset 0x0000.00F0



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	SSI PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system.

## Register 21: SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8

The **SSIPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

### SSI PrimeCell Identification 2 (SSIPCellID2)

SSI0 base: 0x4000.8000

Offset 0xFF8

Type RO, reset 0x0000.0005

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

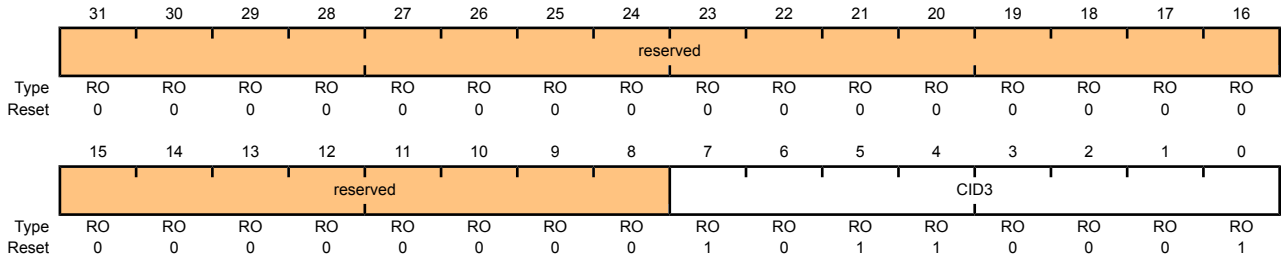
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	SSI PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system.

### Register 22: SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC

The **SSIPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

#### SSI PrimeCell Identification 3 (SSIPCellID3)

SSI0 base: 0x4000.8000  
 Offset 0xFFC  
 Type RO, reset 0x0000.00B1



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	SSI PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system.

## 16 Inter-Integrated Circuit (I<sup>2</sup>C) Interface

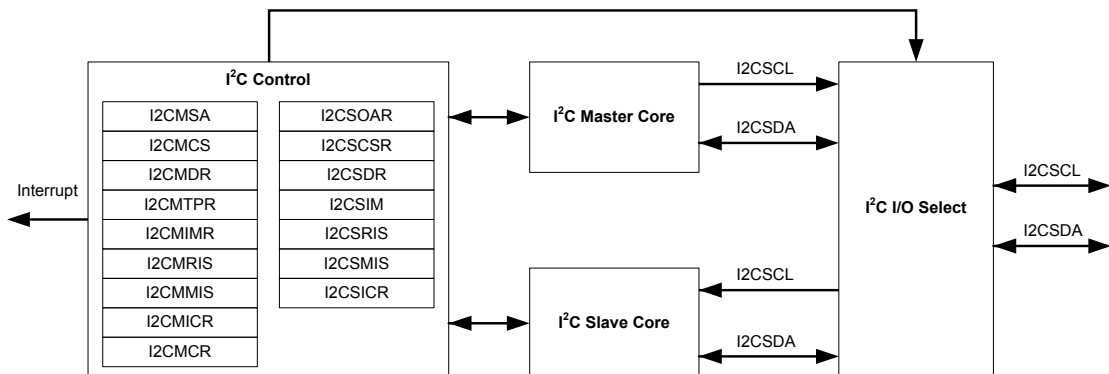
The Inter-Integrated Circuit (I<sup>2</sup>C) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL), and interfaces to external I<sup>2</sup>C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The I<sup>2</sup>C bus may also be used for system testing and diagnostic purposes in product development and manufacture. The LM3S1607 microcontroller includes two I<sup>2</sup>C modules, providing the ability to interact (both send and receive) with other I<sup>2</sup>C devices on the bus.

Devices on the I<sup>2</sup>C bus can be designated as either a master or a slave. Each Stellaris<sup>®</sup> I<sup>2</sup>C module supports both sending and receiving data as either a master or a slave, and also supports the simultaneous operation as both a master and a slave. There are a total of four I<sup>2</sup>C modes: Master Transmit, Master Receive, Slave Transmit, and Slave Receive. The Stellaris<sup>®</sup> I<sup>2</sup>C modules can operate at two speeds: Standard (100 Kbps) and Fast (400 Kbps).

Both the I<sup>2</sup>C master and slave can generate interrupts; the I<sup>2</sup>C master generates interrupts when a transmit or receive operation completes (or aborts due to an error) and the I<sup>2</sup>C slave generates interrupts when data has been sent or requested by a master.

### 16.1 Block Diagram

Figure 16-1. I<sup>2</sup>C Block Diagram

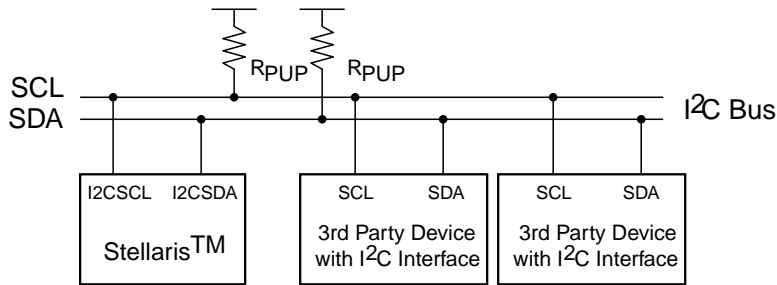


### 16.2 Functional Description

Each I<sup>2</sup>C module is comprised of both master and slave functions which are implemented as separate peripherals. For proper operation, the SDA and SCL pins must be connected to bi-directional open-drain pads. A typical I<sup>2</sup>C bus configuration is shown in Figure 16-2 on page 454.

See "I<sup>2</sup>C" on page 505 for I<sup>2</sup>C timing diagrams.

Figure 16-2. I<sup>2</sup>C Bus Configuration



### 16.2.1 I<sup>2</sup>C Bus Functional Overview

The I<sup>2</sup>C bus uses only two signals: SDA and SCL, named I2CSDA and I2CSCL on Stellaris<sup>®</sup> microcontrollers. SDA is the bi-directional serial data line and SCL is the bi-directional serial clock line. The bus is considered idle when both lines are high.

Every transaction on the I<sup>2</sup>C bus is nine bits long, consisting of eight data bits and a single acknowledge bit. The number of bytes per transfer (defined as the time between a valid START and STOP condition, described in “START and STOP Conditions” on page 454) is unrestricted, but each byte has to be followed by an acknowledge bit, and data must be transferred MSB first. When a receiver cannot receive another complete byte, it can hold the clock line SCL Low and force the transmitter into a wait state. The data transfer continues when the receiver releases the clock SCL.

#### 16.2.1.1 START and STOP Conditions

The protocol of the I<sup>2</sup>C bus defines two states to begin and end a transaction: START and STOP. A high-to-low transition on the SDA line while the SCL is high is defined as a START condition, and a low-to-high transition on the SDA line while SCL is high is defined as a STOP condition. The bus is considered busy after a START condition and free after a STOP condition. See Figure 16-3 on page 454.

Figure 16-3. START and STOP Conditions



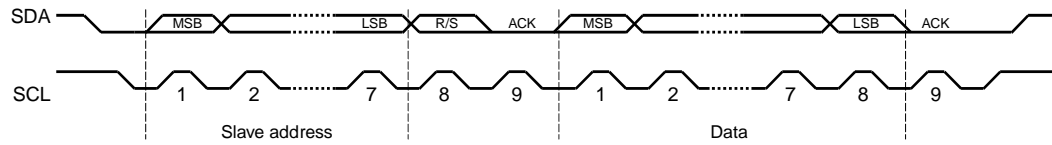
When operating in slave mode, two bits in the I2CRIS register indicate detection of start and stop conditions on the bus; while two bits in the I2CSMIS register allow start and stop conditions to be promoted to controller interrupts (when interrupts are enabled).

#### 16.2.1.2 Data Format with 7-Bit Address

Data transfers follow the format shown in Figure 16-4 on page 455. After the START condition, a slave address is sent. This address is 7-bits long followed by an eighth bit, which is a data direction bit (R/S bit in the I2CMSA register). A zero indicates a transmit operation (send), and a one indicates a request for data (receive). A data transfer is always terminated by a STOP condition generated by the master, however, a master can initiate communications with another device on the bus by generating a repeated START condition and addressing another slave without first generating a

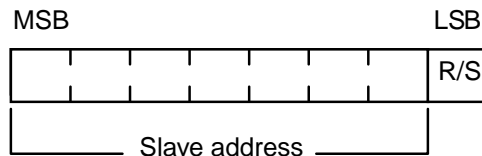
STOP condition. Various combinations of receive/send formats are then possible within a single transfer.

**Figure 16-4. Complete Data Transfer with a 7-Bit Address**



The first seven bits of the first byte make up the slave address (see Figure 16-5 on page 455). The eighth bit determines the direction of the message. A zero in the R/S position of the first byte means that the master will write (send) data to the selected slave, and a one in this position means that the master will receive data from the slave.

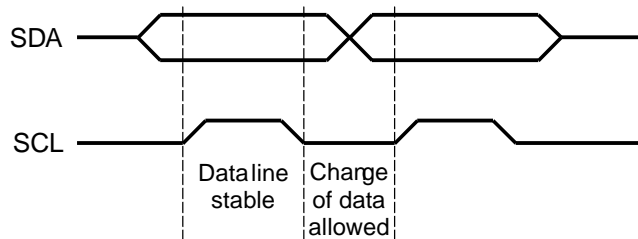
**Figure 16-5. R/S Bit in First Byte**



### 16.2.1.3 Data Validity

The data on the SDA line must be stable during the high period of the clock, and the data line can only change when SCL is low (see Figure 16-6 on page 455).

**Figure 16-6. Data Validity During Bit Transfer on the I<sup>2</sup>C Bus**



### 16.2.1.4 Acknowledge

All bus transactions have a required acknowledge clock cycle that is generated by the master. During the acknowledge cycle, the transmitter (which can be the master or slave) releases the SDA line. To acknowledge the transaction, the receiver must pull down SDA during the acknowledge clock cycle. The data sent out by the receiver during the acknowledge cycle must comply with the data validity requirements described in "Data Validity" on page 455.

When a slave receiver does not acknowledge the slave address, SDA must be left high by the slave so that the master can generate a STOP condition and abort the current transfer. If the master device is acting as a receiver during a transfer, it is responsible for acknowledging each transfer made by the slave. Since the master controls the number of bytes in the transfer, it signals the end of data to the slave transmitter by not generating an acknowledge on the last data byte. The slave transmitter must then release SDA to allow the master to generate the STOP or a repeated START condition.

### 16.2.1.5 Arbitration

A master may start a transfer only if the bus is idle. It's possible for two or more masters to generate a START condition within minimum hold time of the START condition. In these situations, an arbitration scheme takes place on the SDA line, while SCL is high. During arbitration, the first of the competing master devices to place a '1' (high) on SDA while another master transmits a '0' (low) will switch off its data output stage and retire until the bus is idle again.

Arbitration can take place over several bits. Its first stage is a comparison of address bits, and if both masters are trying to address the same device, arbitration continues on to the comparison of data bits.

### 16.2.2 Available Speed Modes

The I<sup>2</sup>C clock rate is determined by the parameters: CLK\_PRD, TIMER\_PRD, SCL\_LP, and SCL\_HP. where:

CLK\_PRD is the system clock period

SCL\_LP is the low phase of SCL (fixed at 6)

SCL\_HP is the high phase of SCL (fixed at 4)

TIMER\_PRD is the programmed value in the **I<sup>2</sup>C Master Timer Period (I2CMTPR)** register (see page 473).

The I<sup>2</sup>C clock period is calculated as follows:

$$SCL\_PERIOD = 2 * (1 + TIMER\_PRD) * (SCL\_LP + SCL\_HP) * CLK\_PRD$$

For example:

CLK\_PRD = 50 ns

TIMER\_PRD = 2

SCL\_LP=6

SCL\_HP=4

yields a SCL frequency of:

$$1/T = 333 \text{ Khz}$$

Table 16-1 on page 456 gives examples of timer period, system clock, and speed mode (Standard or Fast).

**Table 16-1. Examples of I<sup>2</sup>C Master Timer Period versus Speed Mode**

System Clock	Timer Period	Standard Mode	Timer Period	Fast Mode
4 Mhz	0x01	100 Kbps	-	-
6 Mhz	0x02	100 Kbps	-	-
12.5 Mhz	0x06	89 Kbps	0x01	312 Kbps
16.7 Mhz	0x08	93 Kbps	0x02	278 Kbps
20 Mhz	0x09	100 Kbps	0x02	333 Kbps
25 Mhz	0x0C	96.2 Kbps	0x03	312 Kbps
33Mhz	0x10	97.1 Kbps	0x04	330 Kbps
40Mhz	0x13	100 Kbps	0x04	400 Kbps



System Clock	Timer Period	Standard Mode	Timer Period	Fast Mode
50MHz	0x18	100 Kbps	0x06	357 Kbps

### 16.2.3 Interrupts

The I<sup>2</sup>C can generate interrupts when the following conditions are observed:

- Master transaction completed
- Master transaction error
- Slave transaction received
- Slave transaction requested
- Stop condition on bus detected
- Start condition on bus detected

There is a separate interrupt signal for the I<sup>2</sup>C master and I<sup>2</sup>C slave modules. While both modules can generate interrupts for multiple conditions, only a single interrupt signal is sent to the interrupt controller.

#### 16.2.3.1 I<sup>2</sup>C Master Interrupts

The I<sup>2</sup>C master module generates an interrupt when a transaction completes (either transmit or receive), or when an error occurs during a transaction. To enable the I<sup>2</sup>C master interrupt, software must write a '1' to the **I<sup>2</sup>C Master Interrupt Mask (I2CMIMR)** register. When an interrupt condition is met, software must check the `ERROR` bit in the **I<sup>2</sup>C Master Control/Status (I2CMCS)** register to verify that an error didn't occur during the last transaction. An error condition is asserted if the last transaction wasn't acknowledge by the slave or if the master was forced to give up ownership of the bus due to a lost arbitration round with another master. If an error is not detected, the application can proceed with the transfer. The interrupt is cleared by writing a '1' to the **I<sup>2</sup>C Master Interrupt Clear (I2CMICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I<sup>2</sup>C Master Raw Interrupt Status (I2CMRIS)** register.

#### 16.2.3.2 I<sup>2</sup>C Slave Interrupts

The slave module generates interrupts as it receives data and transmit requests from an I<sup>2</sup>C master. The slave module also generates interrupts when a start and stop condition is detected. To enable an I<sup>2</sup>C slave interrupt, write a '1' to the appropriate bit in the **I<sup>2</sup>C Slave Interrupt Mask (I2CSIMR)** register. Software determines whether the module should write (transmit) or read (receive) data from the **I<sup>2</sup>C Slave Data (I2CSDR)** register, by checking the `RREQ` and `TREQ` bits of the **I<sup>2</sup>C Slave Control/Status (I2CSCSR)** register. If the slave module is in receive mode and the first byte of a transfer is received, the `FBR` bit is set along with the `RREQ` bit. The interrupt is cleared by writing a '1' to the **I<sup>2</sup>C Slave Interrupt Clear (I2CSICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I<sup>2</sup>C Slave Raw Interrupt Status (I2CSRIS)** register.

### 16.2.4 Loopback Operation

The I<sup>2</sup>C modules can be placed into an internal loopback mode for diagnostic or debug work. This is accomplished by setting the `LPBK` bit in the **I<sup>2</sup>C Master Configuration (I2CMCR)** register. In loopback mode, the SDA and SCL signals from the master and slave modules are tied together.

### 16.2.5 Command Sequence Flow Charts

This section details the steps required to perform the various I<sup>2</sup>C transfer types in both master and slave mode.

#### 16.2.5.1 I<sup>2</sup>C Master Command Sequences

The figures that follow show the command sequences available for the I<sup>2</sup>C master.

**Figure 16-7. Master Single SEND**

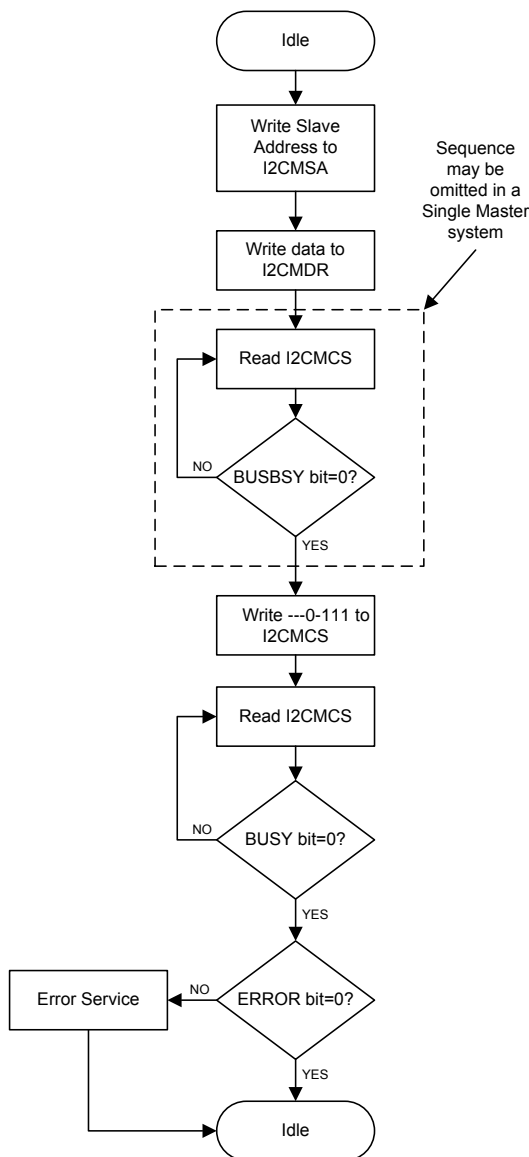


Figure 16-8. Master Single RECEIVE

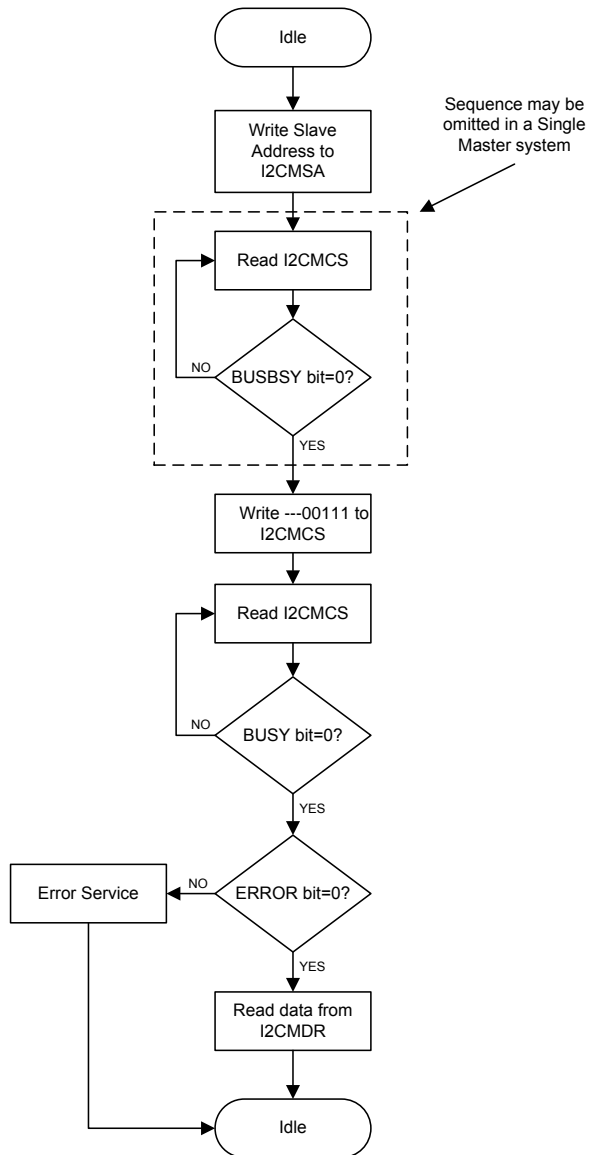


Figure 16-9. Master Burst SEND

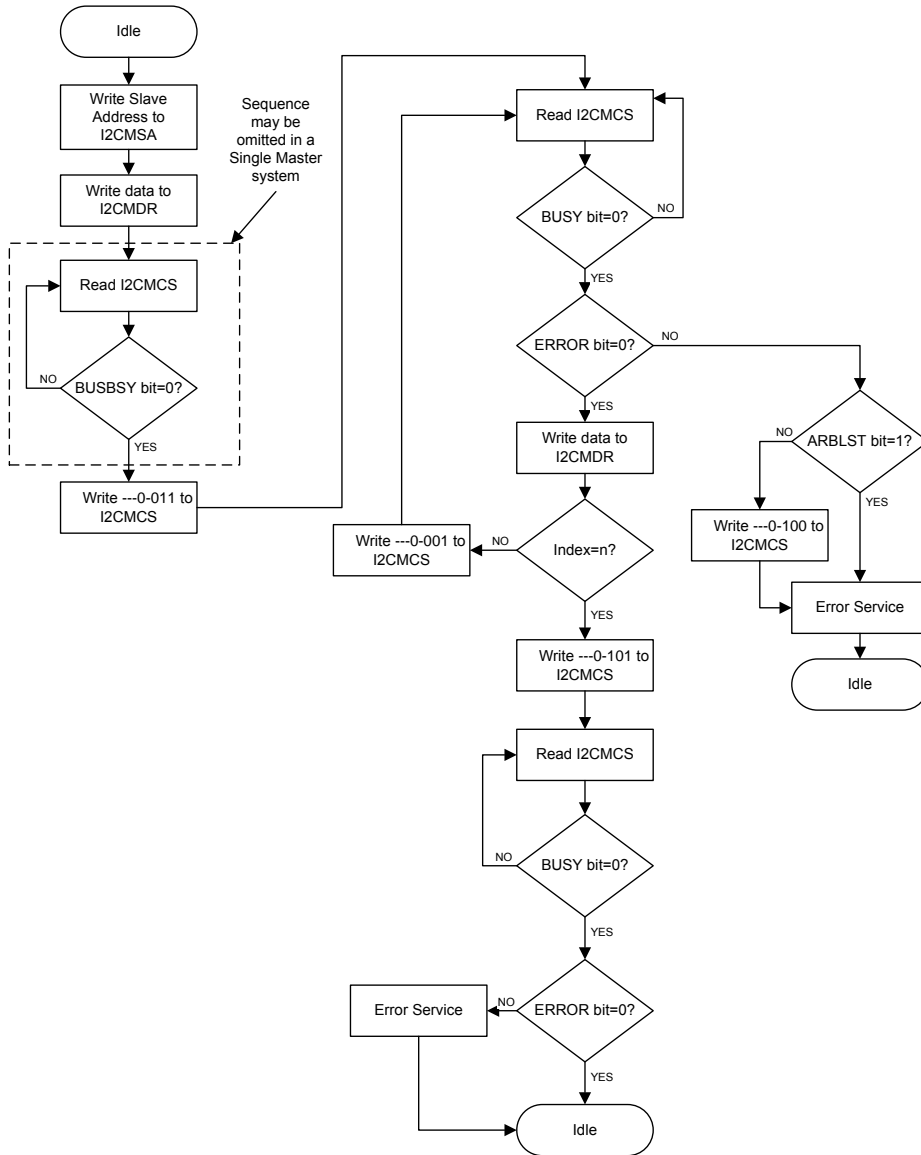


Figure 16-10. Master Burst RECEIVE

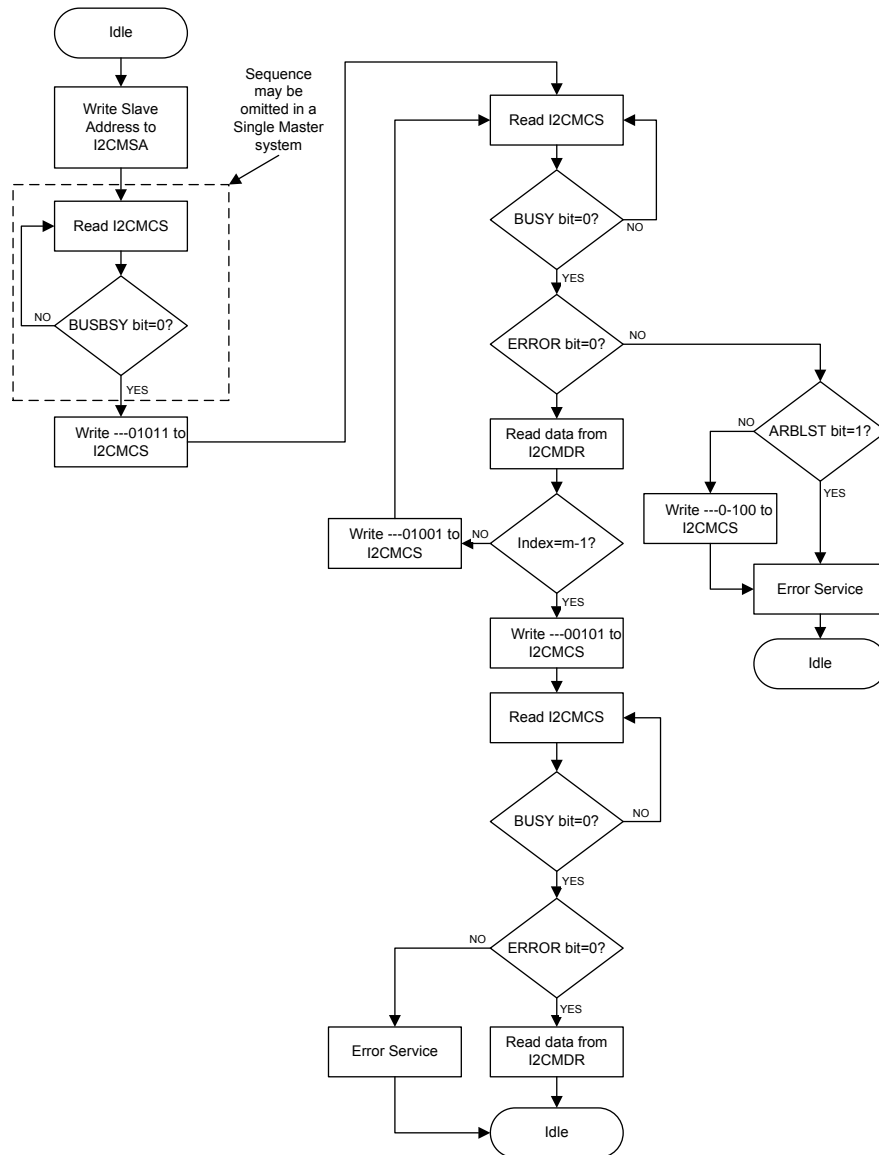


Figure 16-11. Master Burst RECEIVE after Burst SEND

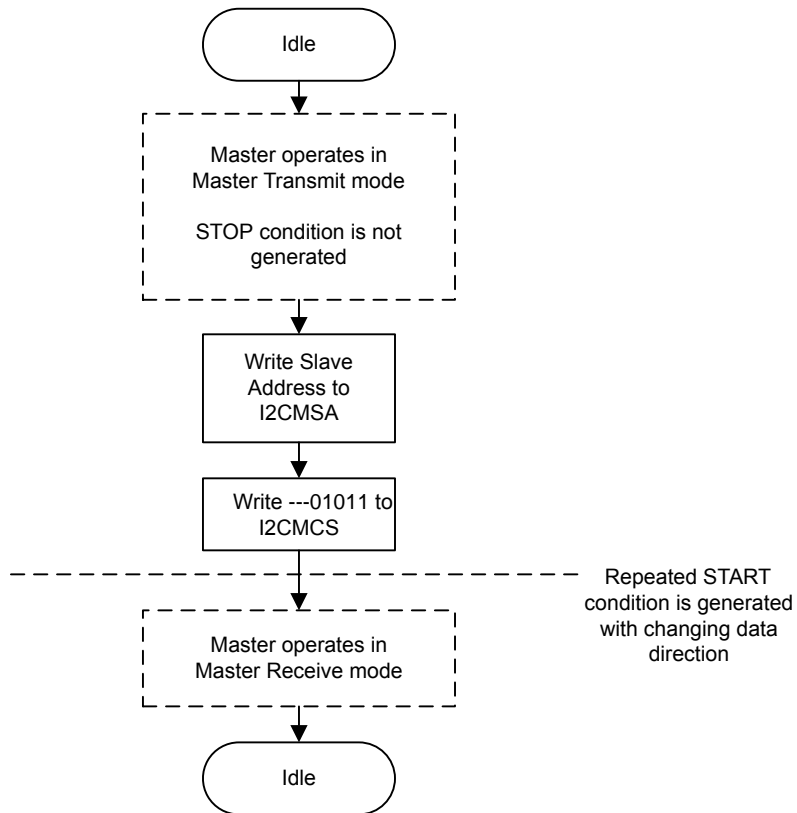
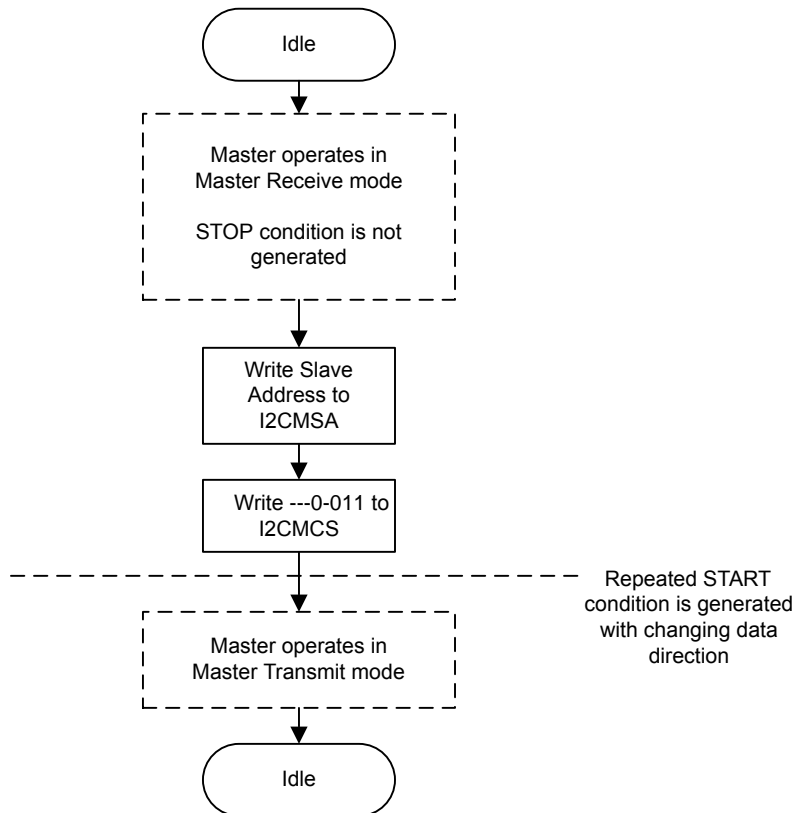


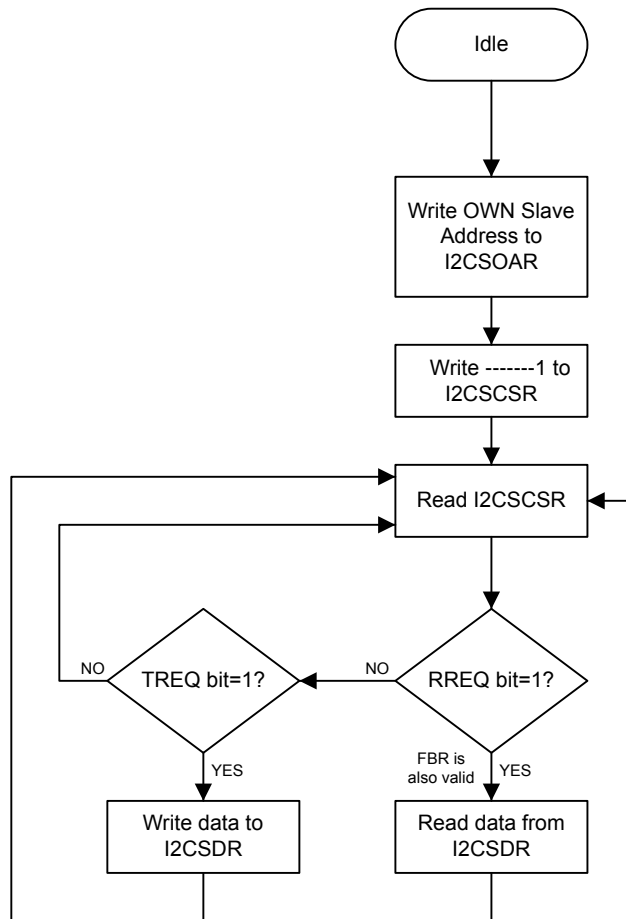
Figure 16-12. Master Burst SEND after Burst RECEIVE



### 16.2.5.2 I<sup>2</sup>C Slave Command Sequences

Figure 16-13 on page 464 presents the command sequence available for the I<sup>2</sup>C slave.

Figure 16-13. Slave Command Sequence



### 16.3 Initialization and Configuration

The following example shows how to configure the I<sup>2</sup>C module to send a single byte as a master. This assumes the system clock is 20 MHz.

1. Enable the I<sup>2</sup>C clock by writing a value of 0x0000.1000 to the **RCGC1** register in the System Control module.
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module.
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register. Also, be sure to enable the same pins for Open Drain operation.
4. Initialize the I<sup>2</sup>C Master by writing the **I2CMCR** register with a value of 0x0000.0020.
5. Set the desired SCL clock speed of 100 Kbps by writing the **I2CMTPR** register with the correct value. The value written to the **I2CMTPR** register represents the number of system clock periods in one SCL clock period. The TPR value is determined by the following equation:



```

TPR = (System Clock / (2 * (SCL_LP + SCL_HP) * SCL_CLK)) - 1;
TPR = (20MHz / (2 * (6 + 4) * 100000)) - 1;
TPR = 9

```

Write the **I2CMTPR** register with the value of 0x0000.0009.

6. Specify the slave address of the master and that the next operation will be a Send by writing the **I2CMSA** register with a value of 0x0000.0076. This sets the slave address to 0x3B.
7. Place data (byte) to be sent in the data register by writing the **I2CMDR** register with the desired data.
8. Initiate a single byte send of the data from Master to Slave by writing the **I2CMCS** register with a value of 0x0000.0007 (STOP, START, RUN).
9. Wait until the transmission completes by polling the **I2CMCS** register's **BUSBSY** bit until it has been cleared.

## 16.4 I<sup>2</sup>C Register Map

Table 16-2 on page 465 lists the I<sup>2</sup>C registers. All addresses given are relative to the I<sup>2</sup>C base addresses for the master and slave:

- I<sup>2</sup>C Master 0: 0x4002.0000
- I<sup>2</sup>C Slave 0: 0x4002.0800
- I<sup>2</sup>C Master 1: 0x4002.1000
- I<sup>2</sup>C Slave 1: 0x4002.1800

**Table 16-2. Inter-Integrated Circuit (I<sup>2</sup>C) Interface Register Map**

Offset	Name	Type	Reset	Description	See page
<b>I<sup>2</sup>C Master</b>					
0x000	I2CMSA	R/W	0x0000.0000	I2C Master Slave Address	467
0x004	I2CMCS	R/W	0x0000.0000	I2C Master Control/Status	468
0x008	I2CMDR	R/W	0x0000.0000	I2C Master Data	472
0x00C	I2CMTPR	R/W	0x0000.0001	I2C Master Timer Period	473
0x010	I2CMIMR	R/W	0x0000.0000	I2C Master Interrupt Mask	474
0x014	I2CMRIS	RO	0x0000.0000	I2C Master Raw Interrupt Status	475
0x018	I2CMMIS	RO	0x0000.0000	I2C Master Masked Interrupt Status	476
0x01C	I2CMICR	WO	0x0000.0000	I2C Master Interrupt Clear	477
0x020	I2CMCR	R/W	0x0000.0000	I2C Master Configuration	478
<b>I<sup>2</sup>C Slave</b>					
0x000	I2CSOAR	R/W	0x0000.0000	I2C Slave Own Address	480

Offset	Name	Type	Reset	Description	See page
0x004	I2CSCSR	RO	0x0000.0000	I2C Slave Control/Status	481
0x008	I2CSDR	R/W	0x0000.0000	I2C Slave Data	483
0x00C	I2CSIMR	R/W	0x0000.0000	I2C Slave Interrupt Mask	484
0x010	I2CSRIS	RO	0x0000.0000	I2C Slave Raw Interrupt Status	485
0x014	I2CSMIS	RO	0x0000.0000	I2C Slave Masked Interrupt Status	486
0x018	I2CSICR	WO	0x0000.0000	I2C Slave Interrupt Clear	487

## 16.5 Register Descriptions (I<sup>2</sup>C Master)

The remainder of this section lists and describes the I<sup>2</sup>C master registers, in numerical order by address offset. See also “Register Descriptions (I2C Slave)” on page 479.

## Register 1: I<sup>2</sup>C Master Slave Address (I2CMSA), offset 0x000

This register consists of eight bits: seven address bits (A6-A0), and a Receive/Send bit, which determines if the next operation is a Receive (High), or Send (Low).

### I2C Master Slave Address (I2CMSA)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x000

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:1	SA	R/W	0	I <sup>2</sup> C Slave Address  This field specifies bits A6 through A0 of the slave address.
0	R/S	R/W	0	Receive/Send  The R/S bit specifies if the next operation is a Receive (High) or Send (Low).  Value Description 0 Send. 1 Receive.

## Register 2: I<sup>2</sup>C Master Control/Status (I2CMCS), offset 0x004

This register accesses four control bits when written, and accesses seven status bits when read.

The status register consists of seven bits, which when read determine the state of the I<sup>2</sup>C bus controller.

The control register consists of four bits: the RUN, START, STOP, and ACK bits. The START bit causes the generation of the START, or REPEATED START condition.

The STOP bit determines if the cycle stops at the end of the data cycle, or continues on to a burst. To generate a single send cycle, the I<sup>2</sup>C Master Slave Address (I2CMSA) register is written with the desired address, the R/S bit is set to 0, and the Control register is written with ACK=X (0 or 1), STOP=1, START=1, and RUN=1 to perform the operation and stop. When the operation is completed (or aborted due an error), the interrupt pin becomes active and the data may be read from the I2CMDR register. When the I<sup>2</sup>C module operates in Master receiver mode, the ACK bit must be set normally to logic 1. This causes the I<sup>2</sup>C bus controller to send an acknowledge automatically after each byte. This bit must be reset when the I<sup>2</sup>C bus controller requires no further data to be sent from the slave transmitter.

### Read-Only Status Register

#### I2C Master Control/Status (I2CMCS)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x004

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved										BUSBSY	IDLE	ARBLST	DATAACK	ADRACK	ERROR	BUSY
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	BUSBSY	RO	0	Bus Busy  This bit specifies the state of the I <sup>2</sup> C bus. If set, the bus is busy; otherwise, the bus is idle. The bit changes based on the START and STOP conditions.
5	IDLE	RO	0	I <sup>2</sup> C Idle  This bit specifies the I <sup>2</sup> C controller state. If set, the controller is idle; otherwise the controller is not idle.
4	ARBLST	RO	0	Arbitration Lost  This bit specifies the result of bus arbitration. If set, the controller lost arbitration; otherwise, the controller won arbitration.

Bit/Field	Name	Type	Reset	Description
3	DATAACK	RO	0	Acknowledge Data  This bit specifies the result of the last data operation. If set, the transmitted data was not acknowledged; otherwise, the data was acknowledged.
2	ADRACK	RO	0	Acknowledge Address  This bit specifies the result of the last address operation. If set, the transmitted address was not acknowledged; otherwise, the address was acknowledged.
1	ERROR	RO	0	Error  This bit specifies the result of the last bus operation. If set, an error occurred on the last operation; otherwise, no error was detected. The error can be from the slave address not being acknowledged, the transmit data not being acknowledged, or because the controller lost arbitration.
0	BUSY	RO	0	I <sup>2</sup> C Busy  This bit specifies the state of the controller. If set, the controller is busy; otherwise, the controller is idle. When the <code>BUSY</code> bit is set, the other status bits are not valid.

## Write-Only Control Register

### I2C Master Control/Status (I2CMCS)

I2C Master 0 base: 0x4002.0000  
 I2C Master 1 base: 0x4002.1000  
 Offset 0x004  
 Type WO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved													ACK	STOP	START	RUN
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bit/Field	Name	Type	Reset	Description
31:4	reserved	WO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	ACK	WO	0	Data Acknowledge Enable  When set, causes received data byte to be acknowledged automatically by the master. See field decoding in Table 16-3 on page 470.
2	STOP	WO	0	Generate STOP  When set, causes the generation of the STOP condition. See field decoding in Table 16-3 on page 470.

Bit/Field	Name	Type	Reset	Description
1	START	WO	0	Generate START When set, causes the generation of a START or repeated START condition. See field decoding in Table 16-3 on page 470.
0	RUN	WO	0	I <sup>2</sup> C Master Enable When set, allows the master to send or receive data. See field decoding in Table 16-3 on page 470.

**Table 16-3. Write Field Decoding for I2CMCS[3:0] Field (Sheet 1 of 3)**

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
	R/S	ACK	STOP	START	RUN	
Idle	0	X <sup>a</sup>	0	1	1	START condition followed by SEND (master goes to the Master Transmit state).
	0	X	1	1	1	START condition followed by a SEND and STOP condition (master remains in Idle state).
	1	0	0	1	1	START condition followed by RECEIVE operation with negative ACK (master goes to the Master Receive state).
	1	0	1	1	1	START condition followed by RECEIVE and STOP condition (master remains in Idle state).
	1	1	0	1	1	START condition followed by RECEIVE (master goes to the Master Receive state).
	1	1	1	1	1	Illegal.
	All other combinations not listed are non-operations.					
Master Transmit	X	X	0	0	1	SEND operation (master remains in Master Transmit state).
	X	X	1	0	0	STOP condition (master goes to Idle state).
	X	X	1	0	1	SEND followed by STOP condition (master goes to Idle state).
	0	X	0	1	1	Repeated START condition followed by a SEND (master remains in Master Transmit state).
	0	X	1	1	1	Repeated START condition followed by SEND and STOP condition (master goes to Idle state).
	1	0	0	1	1	Repeated START condition followed by a RECEIVE operation with a negative ACK (master goes to Master Receive state).
	1	0	1	1	1	Repeated START condition followed by a SEND and STOP condition (master goes to Idle state).
	1	1	0	1	1	Repeated START condition followed by RECEIVE (master goes to Master Receive state).
	1	1	1	1	1	Illegal.
	All other combinations not listed are non-operations.					

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
	R/S	ACK	STOP	START	RUN	
Master Receive	X	0	0	0	1	RECEIVE operation with negative ACK (master remains in Master Receive state).
	X	X	1	0	0	STOP condition (master goes to Idle state). <sup>b</sup>
	X	0	1	0	1	RECEIVE followed by STOP condition (master goes to Idle state).
	X	1	0	0	1	RECEIVE operation (master remains in Master Receive state).
	X	1	1	0	1	Illegal.
	1	0	0	1	1	Repeated START condition followed by RECEIVE operation with a negative ACK (master remains in Master Receive state).
	1	0	1	1	1	Repeated START condition followed by RECEIVE and STOP condition (master goes to Idle state).
	1	1	0	1	1	Repeated START condition followed by RECEIVE (master remains in Master Receive state).
	0	X	0	1	1	Repeated START condition followed by SEND (master goes to Master Transmit state).
	0	X	1	1	1	Repeated START condition followed by SEND and STOP condition (master goes to Idle state).
All other combinations not listed are non-operations.						NOP.

a. An X in a table cell indicates the bit can be 0 or 1.

b. In Master Receive mode, a STOP condition should be generated only after a Data Negative Acknowledge executed by the master or an Address Negative Acknowledge executed by the slave.

### Register 3: I<sup>2</sup>C Master Data (I2CMDR), offset 0x008

This register contains the data to be transmitted when in the Master Transmit state, and the data received when in the Master Receive state.

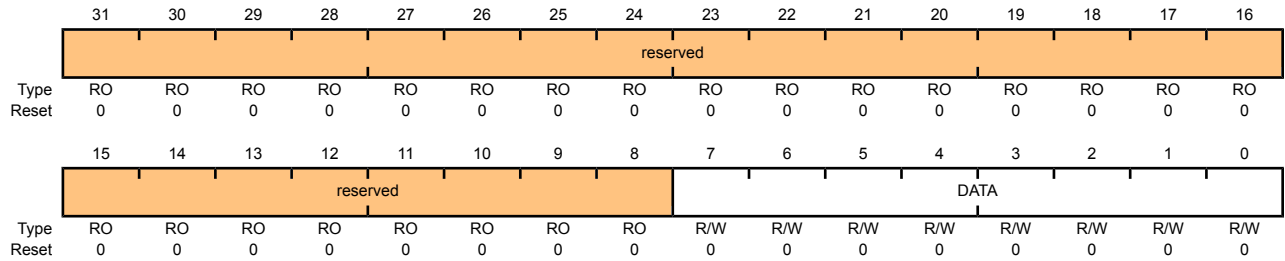
#### I2C Master Data (I2CMDR)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x008

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x00	Data Transferred Data transferred during transaction.



## Register 4: I<sup>2</sup>C Master Timer Period (I2CMTPR), offset 0x00C

This register specifies the period of the SCL clock.

### I2C Master Timer Period (I2CMTPR)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x00C

Type R/W, reset 0x0000.0001

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TPR	R/W	0x1	SCL Clock Period

This field specifies the period of the SCL clock.

$$SCL\_PRD = 2 * (1 + TPR) * (SCL\_LP + SCL\_HP) * CLK\_PRD$$

where:

SCL\_PRD is the SCL line period (I<sup>2</sup>C clock).

TPR is the Timer Period register value (range of 1 to 255).

SCL\_LP is the SCL Low period (fixed at 6).

SCL\_HP is the SCL High period (fixed at 4).

### Register 5: I<sup>2</sup>C Master Interrupt Mask (I2CMIMR), offset 0x010

This register controls whether a raw interrupt is promoted to a controller interrupt.

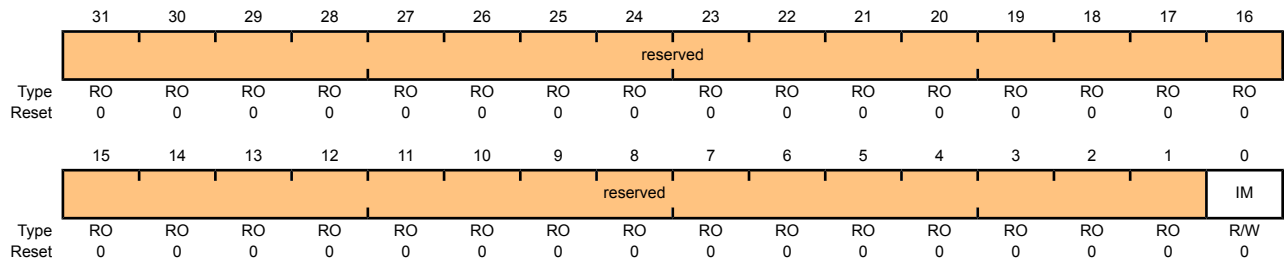
#### I2C Master Interrupt Mask (I2CMIMR)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x010

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	IM	R/W	0	Interrupt Mask  This bit controls whether a raw interrupt is promoted to a controller interrupt. If set, the interrupt is not masked and the interrupt is promoted; otherwise, the interrupt is masked.

## Register 6: I<sup>2</sup>C Master Raw Interrupt Status (I2CMRIS), offset 0x014

This register specifies whether an interrupt is pending.

### I2C Master Raw Interrupt Status (I2CMRIS)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x014

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	RIS	RO	0	Raw Interrupt Status  This bit specifies the raw interrupt state (prior to masking) of the I <sup>2</sup> C master block. If set, an interrupt is pending; otherwise, an interrupt is not pending.

## Register 7: I<sup>2</sup>C Master Masked Interrupt Status (I2CMMIS), offset 0x018

This register specifies whether an interrupt was signaled.

### I2C Master Masked Interrupt Status (I2CMMIS)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x018

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	MIS	RO	0	Masked Interrupt Status  This bit specifies the raw interrupt state (after masking) of the I <sup>2</sup> C master block. If set, an interrupt was signaled; otherwise, an interrupt has not been generated since the bit was last cleared.

## Register 8: I<sup>2</sup>C Master Interrupt Clear (I2CMICR), offset 0x01C

This register clears the raw interrupt.

### I2C Master Interrupt Clear (I2CMICR)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x01C

Type WO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
																IC
Type																WO
Reset																0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	IC	WO	0	Interrupt Clear  This bit controls the clearing of the raw interrupt. A write of 1 clears the interrupt; otherwise, a write of 0 has no affect on the interrupt state. A read of this register returns no meaningful data.

### Register 9: I<sup>2</sup>C Master Configuration (I2CMCR), offset 0x020

This register configures the mode (Master or Slave) and sets the interface for test mode loopback.

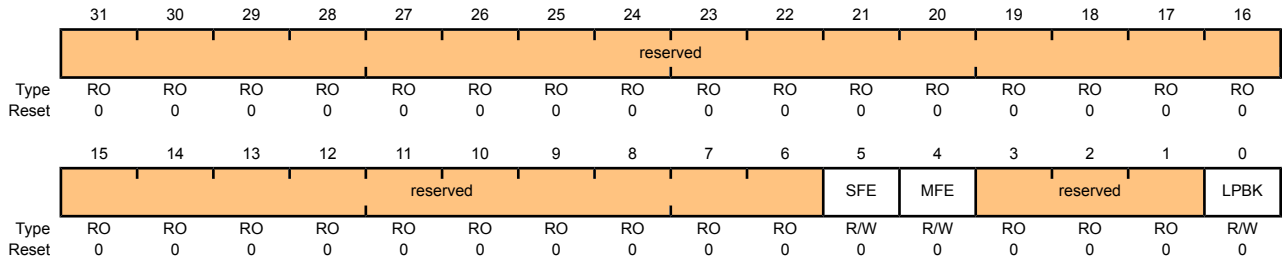
#### I2C Master Configuration (I2CMCR)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x020

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SFE	R/W	0	I <sup>2</sup> C Slave Function Enable  This bit specifies whether the interface may operate in Slave mode. If set, Slave mode is enabled; otherwise, Slave mode is disabled.
4	MFE	R/W	0	I <sup>2</sup> C Master Function Enable  This bit specifies whether the interface may operate in Master mode. If set, Master mode is enabled; otherwise, Master mode is disabled and the interface clock is disabled.
3:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	LPBK	R/W	0	I <sup>2</sup> C Loopback  This bit specifies whether the interface is operating normally or in Loopback mode. If set, the device is put in a test mode loopback configuration; otherwise, the device operates normally.

## 16.6 Register Descriptions (I2C Slave)

The remainder of this section lists and describes the I<sup>2</sup>C slave registers, in numerical order by address offset. See also “Register Descriptions (I<sup>2</sup>C Master)” on page 466.

### Register 10: I<sup>2</sup>C Slave Own Address (I2CSOAR), offset 0x000

This register consists of seven address bits that identify the Stellaris<sup>®</sup> I<sup>2</sup>C device on the I<sup>2</sup>C bus.

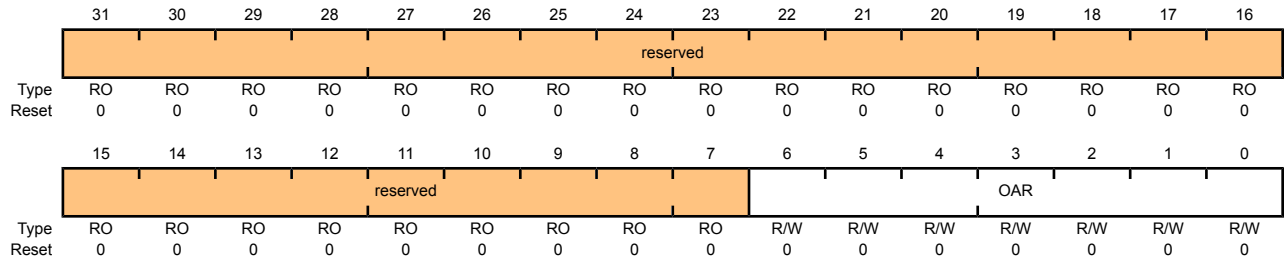
#### I2C Slave Own Address (I2CSOAR)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x000

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	OAR	R/W	0x00	I <sup>2</sup> C Slave Own Address This field specifies bits A6 through A0 of the slave address.



## Register 11: I<sup>2</sup>C Slave Control/Status (I2CCSR), offset 0x004

This register accesses one control bit when written, and three status bits when read.

The read-only Status register consists of three bits: the `FBR`, `RREQ`, and `TREQ` bits. The `First Byte Received (FBR)` bit is set only after the Stellaris<sup>®</sup> device detects its own slave address and receives the first data byte from the I<sup>2</sup>C master. The `Receive Request (RREQ)` bit indicates that the Stellaris<sup>®</sup> I<sup>2</sup>C device has received a data byte from an I<sup>2</sup>C master. Read one data byte from the **I<sup>2</sup>C Slave Data (I2CSDR)** register to clear the `RREQ` bit. The `Transmit Request (TREQ)` bit indicates that the Stellaris<sup>®</sup> I<sup>2</sup>C device is addressed as a Slave Transmitter. Write one data byte into the **I<sup>2</sup>C Slave Data (I2CSDR)** register to clear the `TREQ` bit.

The write-only Control register consists of one bit: the `DA` bit. The `DA` bit enables and disables the Stellaris<sup>®</sup> I<sup>2</sup>C slave operation.

### Read-Only Status Register

#### I2C Slave Control/Status (I2CCSR)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x004

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													FBR	TREQ	RREQ
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	FBR	RO	0	<p>First Byte Received</p> <p>Indicates that the first byte following the slave's own address is received. This bit is only valid when the <code>RREQ</code> bit is set, and is automatically cleared when data has been read from the <b>I2CSDR</b> register.</p> <p><b>Note:</b> This bit is not used for slave transmit operations.</p>
1	TREQ	RO	0	<p>Transmit Request</p> <p>This bit specifies the state of the I<sup>2</sup>C slave with regards to outstanding transmit requests. If set, the I<sup>2</sup>C unit has been addressed as a slave transmitter and uses clock stretching to delay the master until data has been written to the <b>I2CSDR</b> register. Otherwise, there is no outstanding transmit request.</p>

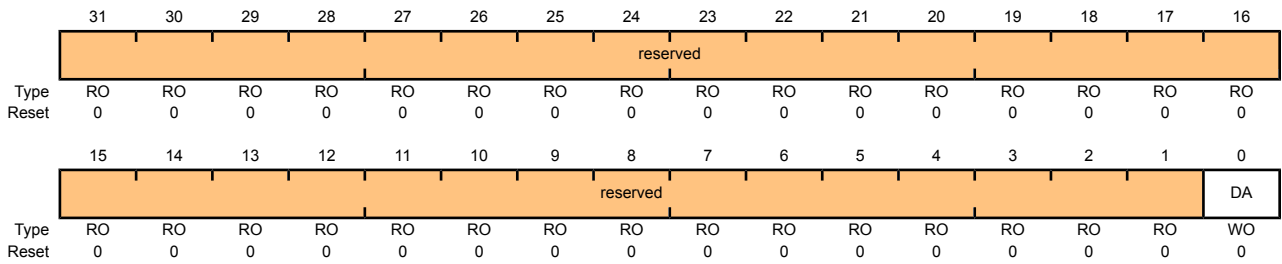
Bit/Field	Name	Type	Reset	Description
0	RREQ	RO	0	Receive Request

This bit specifies the status of the I<sup>2</sup>C slave with regards to outstanding receive requests. If set, the I<sup>2</sup>C unit has outstanding receive data from the I<sup>2</sup>C master and uses clock stretching to delay the master until the data has been read from the **I2CSDR** register. Otherwise, no receive data is outstanding.

### Write-Only Control Register

#### I2C Slave Control/Status (I2CSCSR)

I2C Slave 0 base: 0x4002.0800  
 I2C Slave 1 base: 0x4002.1800  
 Offset 0x004  
 Type WO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	DA	WO	0	Device Active

Value Description  
 0 Disables the I<sup>2</sup>C slave operation.  
 1 Enables the I<sup>2</sup>C slave operation.

## Register 12: I<sup>2</sup>C Slave Data (I2CSDR), offset 0x008

This register contains the data to be transmitted when in the Slave Transmit state, and the data received when in the Slave Receive state.

### I2C Slave Data (I2CSDR)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x008

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x0	Data for Transfer  This field contains the data for transfer during a slave receive or transmit operation.

### Register 13: I<sup>2</sup>C Slave Interrupt Mask (I2CSIMR), offset 0x00C

This register controls whether a raw interrupt is promoted to a controller interrupt.

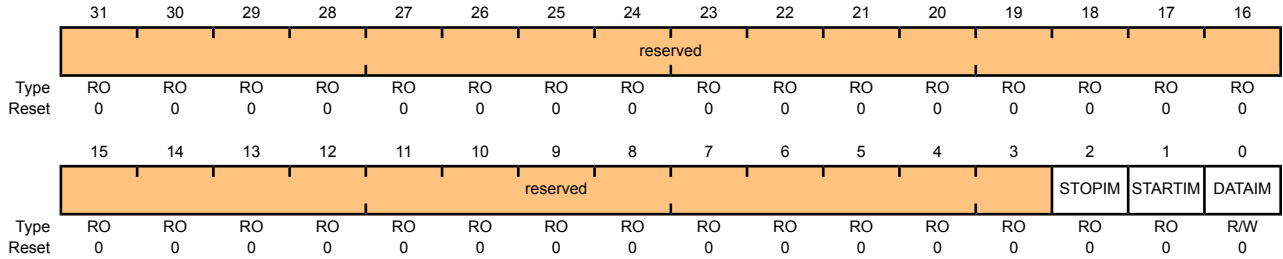
#### I2C Slave Interrupt Mask (I2CSIMR)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x00C

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPIM	RO	0	Stop Condition Interrupt Mask  This bit controls whether the raw interrupt for detection of a stop condition on the I <sup>2</sup> C bus is promoted to a controller interrupt. If set, the interrupt is not masked and the interrupt is promoted; otherwise, the interrupt is masked.
1	STARTIM	RO	0	Start Condition Interrupt Mask  This bit controls whether the raw interrupt for detection of a start condition on the I <sup>2</sup> C bus is promoted to a controller interrupt. If set, the interrupt is not masked and the interrupt is promoted; otherwise, the interrupt is masked.
0	DATAIM	R/W	0	Data Interrupt Mask  This bit controls whether the raw interrupt for data received and data requested is promoted to a controller interrupt. If set, the interrupt is not masked and the interrupt is promoted; otherwise, the interrupt is masked.

## Register 14: I<sup>2</sup>C Slave Raw Interrupt Status (I2CSRIS), offset 0x010

This register specifies whether an interrupt is pending.

### I2C Slave Raw Interrupt Status (I2CSRIS)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x010

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													STOPRIS	STARTRIS	DATARIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPRIS	RO	0	Stop Condition Raw Interrupt Status  This bit specifies the raw interrupt state for stop condition detect (prior to masking) of the I <sup>2</sup> C slave block. If set, an interrupt is pending; otherwise, an interrupt is not pending.
1	STARTRIS	RO	0	Start Condition Raw Interrupt Status  This bit specifies the raw interrupt state for start condition detect (prior to masking) of the I <sup>2</sup> C slave block. If set, an interrupt is pending; otherwise, an interrupt is not pending.
0	DATARIS	RO	0	Data Raw Interrupt Status  This bit specifies the raw interrupt state for data received and data requested (prior to masking) of the I <sup>2</sup> C slave block. If set, an interrupt is pending; otherwise, an interrupt is not pending.

## Register 15: I<sup>2</sup>C Slave Masked Interrupt Status (I2CSMIS), offset 0x014

This register specifies whether an interrupt was signaled.

### I2C Slave Masked Interrupt Status (I2CSMIS)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x014

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved													STOPMIS	STARTMIS	DATAMIS	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPMIS	RW	0	Stop Condition Masked Interrupt Status  This bit specifies the interrupt state for stop condition detect (after masking) of the I <sup>2</sup> C slave block. If set, an interrupt was signaled; otherwise, an interrupt has not been generated since the bit was last cleared.
1	STARTMIS	RW	0	Start Condition Masked Interrupt Status  This bit specifies the interrupt state for start condition detect (after masking) of the I <sup>2</sup> C slave block. If set, an interrupt was signaled; otherwise, an interrupt has not been generated since the bit was last cleared.
0	DATAMIS	RO	0	Data Masked Interrupt Status  This bit specifies the interrupt state for data received and data requested (after masking) of the I <sup>2</sup> C slave block. If set, an interrupt was signaled; otherwise, an interrupt has not been generated since the bit was last cleared.

## Register 16: I<sup>2</sup>C Slave Interrupt Clear (I2CSICR), offset 0x018

This register clears the raw interrupt. A read of this register returns no meaningful data.

### I2C Slave Interrupt Clear (I2CSICR)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x018

Type WO, reset 0x0000.0000

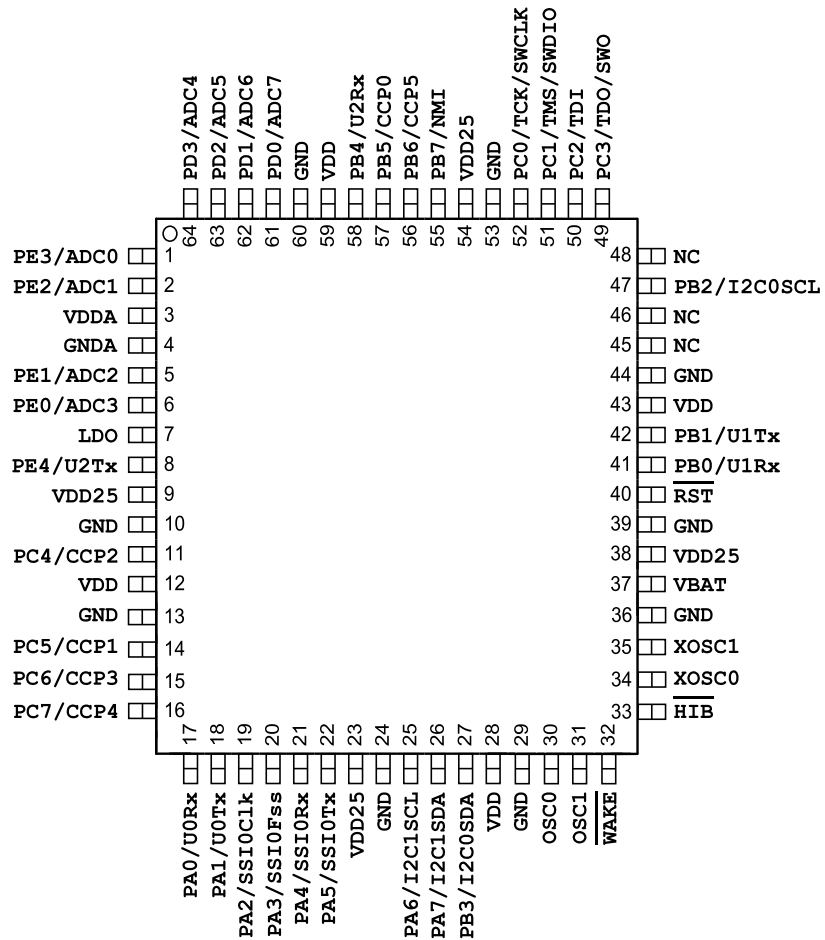
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved													STOPIC	STARTIC	DATAIC	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPIC	WO	0	Stop Condition Interrupt Clear  This bit controls the clearing of the raw interrupt for stop condition detect. When set, it clears the STOPRIS interrupt bit; otherwise, it has no effect on the STOPRIS bit value.
1	STARTIC	WO	0	Start Condition Interrupt Clear  This bit controls the clearing of the raw interrupt for start condition detect. When set, it clears the STARTRIS interrupt bit; otherwise, it has no effect on the STARTRIS bit value.
0	DATAIC	WO	0	Data Interrupt Clear  This bit controls the clearing of the raw interrupt for data received and data requested. When set, it clears the DATARIS interrupt bit; otherwise, it has no effect on the DATARIS bit value.

# 17 Pin Diagram

The LM3S1607 microcontroller pin diagram is shown below.

Figure 17-1. 64-Pin LQFP Package Pin Diagram



LM3S1607



## 18 Signal Tables

The following tables list the signals available for each pin. Functionality is enabled by software with the **GPIOAFSEL** register.

**Important:** All multiplexed pins are GPIOs by default, with the exception of the four JTAG pins (PC[3:0]) which default to the JTAG functionality.

Table 18-1 on page 489 shows the pin-to-signal-name mapping, including functional characteristics of the signals. Table 18-2 on page 492 lists the signals in alphabetical order by signal name.

Table 18-3 on page 495 groups the signals by functionality, except for GPIOs. Table 18-4 on page 497 lists the GPIO pins and their alternate functionality.

**Table 18-1. Signals by Pin Number**

Pin Number	Pin Name	Pin Type	Buffer Type	Description
1	PE3	I/O	Analog	GPIO port E bit 3
	ADC0	I	Analog	ADC 0 input
2	PE2	I/O	Analog	GPIO port E bit 2
	ADC1	I	Analog	ADC 1 input
3	VDDA	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions.
4	GNDA	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
5	PE1	I/O	Analog	GPIO port E bit 1
	ADC2	I	Analog	ADC 2 input
6	PE0	I/O	Analog	GPIO port E bit 0
	ADC3	I	Analog	ADC 3 input
7	LDO	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 $\mu$ F or greater. The LDO pin must also be connected to the VDD25 pins at the board level in addition to the decoupling capacitor(s).
8	PE4	I/O	TTL	GPIO port E bit 4
	U2Tx	O	TTL	UART2 transmit
9	VDD25	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
10	GND	-	Power	Ground reference for logic and I/O pins.
11	PC4	I/O	TTL	GPIO port C bit 4
	CCP2	I/O	TTL	Capture/Compare/PWM 2
12	VDD	-	Power	Positive supply for I/O and some logic.
13	GND	-	Power	Ground reference for logic and I/O pins.

Pin Number	Pin Name	Pin Type	Buffer Type	Description
14	PC5	I/O	TTL	GPIO port C bit 5
	CCP1	I/O	TTL	Capture/Compare/PWM 1
15	PC6	I/O	TTL	GPIO port C bit 6
	CCP3	I/O	TTL	Capture/Compare/PWM 3
16	PC7	I/O	TTL	GPIO port C bit 7
	CCP4	I/O	TTL	Capture/Compare/PWM 4
17	PA0	I/O	TTL	GPIO port A bit 0
	U0Rx	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
18	PA1	I/O	TTL	GPIO port A bit 1
	U0Tx	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
19	PA2	I/O	TTL	GPIO port A bit 2
	SSI0Clk	I/O	TTL	SSI module 0 clock
20	PA3	I/O	TTL	GPIO port A bit 3
	SSI0Fss	I/O	TTL	SSI module 0 frame
21	PA4	I/O	TTL	GPIO port A bit 4
	SSI0Rx	I	TTL	SSI module 0 receive
22	PA5	I/O	TTL	GPIO port A bit 5
	SSI0Tx	O	TTL	SSI module 0 transmit
23	VDD25	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
24	GND	-	Power	Ground reference for logic and I/O pins.
25	PA6	I/O	TTL	GPIO port A bit 6
	I2C1SCL	I/O	OD	I2C module 1 clock
26	PA7	I/O	TTL	GPIO port A bit 7
	I2C1SDA	I/O	OD	I2C module 1 data
27	PB3	I/O	TTL	GPIO port B bit 3
	I2C0SDA	I/O	OD	I2C module 0 data
28	VDD	-	Power	Positive supply for I/O and some logic.
29	GND	-	Power	Ground reference for logic and I/O pins.
30	OSC0	I	Analog	Main oscillator crystal input or an external clock reference input.
31	OSC1	O	Analog	Main oscillator crystal output.
32	WAKE	I	-	An external input that brings the processor out of hibernate mode when asserted.
33	HIB	O	OD	An output that indicates the processor is in hibernate mode.
34	XOSC0	I	Analog	Hibernation Module oscillator crystal input or an external clock reference input. Note that this is either a 4.19-MHz crystal or a 32.768-kHz oscillator for the Hibernation Module RTC. See the CLKSEL bit in the HIBCTL register.
35	XOSC1	O	Analog	Hibernation Module oscillator crystal output.

Pin Number	Pin Name	Pin Type	Buffer Type	Description
36	GND	-	Power	Ground reference for logic and I/O pins.
37	VBAT	-	Power	Power source for the Hibernation Module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation Module power-source supply.
38	VDD25	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
39	GND	-	Power	Ground reference for logic and I/O pins.
40	$\overline{\text{RST}}$	I/O	TTL	System reset input.
41	PB0	I/O	TTL	GPIO port B bit 0
	U1Rx	I	TTL	UART 1 receive
42	PB1	I/O	TTL	GPIO port B bit 1
	U1Tx	O	TTL	UART 1 transmit
43	VDD	-	Power	Positive supply for I/O and some logic.
44	GND	-	Power	Ground reference for logic and I/O pins.
45	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
46	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
47	PB2	I/O	TTL	GPIO port B bit 2
	I2C0SCL	I/O	OD	I2C module 0 clock
48	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
49	PC3	I/O	TTL	GPIO port C bit 3
	TDO	O	TTL	JTAG TDO and SWO
	SWO	O	TTL	JTAG TDO and SWO
50	PC2	I/O	TTL	GPIO port C bit 2
	TDI	I	TTL	JTAG TDI
51	PC1	I/O	TTL	GPIO port C bit 1
	TMS	I/O	TTL	JTAG TMS and SWDIO
	SWDIO	I/O	TTL	JTAG TMS and SWDIO
52	PC0	I/O	TTL	GPIO port C bit 0
	TCK	I	TTL	JTAG/SWD CLK
	SWCLK	I	TTL	JTAG/SWD CLK
53	GND	-	Power	Ground reference for logic and I/O pins.
54	VDD25	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
55	PB7	I/O	TTL	GPIO port B bit 7
	NMI	I	TTL	Non maskable interrupt
56	PB6	I/O	TTL	GPIO port B bit 6
	CCP5	I/O	TTL	Capture/Compare/PWM 5
57	PB5	I/O	TTL	GPIO port B bit 5
	CCP0	I/O	TTL	Capture/Compare/PWM 0

Pin Number	Pin Name	Pin Type	Buffer Type	Description
58	PB4	I/O	TTL	GPIO port B bit 4
	U2Rx	I	TTL	UART 2 receive
59	VDD	-	Power	Positive supply for I/O and some logic.
60	GND	-	Power	Ground reference for logic and I/O pins.
61	PD0	I/O	Analog	GPIO port D bit 0
	ADC7	I	Analog	ADC 7 input
62	PD1	I/O	Analog	GPIO port D bit 1
	ADC6	I	Analog	ADC 6 input
63	PD2	I/O	Analog	GPIO port D bit 2
	ADC5	I	Analog	ADC 5 input
64	PD3	I/O	Analog	GPIO port D bit 3
	ADC4	I	Analog	ADC 4 input

Table 18-2. Signals by Signal Name

Pin Name	Pin Number	Pin Type	Buffer Type	Description
ADC0	1	I	Analog	ADC 0 input
ADC1	2	I	Analog	ADC 1 input
ADC2	5	I	Analog	ADC 2 input
ADC3	6	I	Analog	ADC 3 input
ADC4	64	I	Analog	ADC 4 input
ADC5	63	I	Analog	ADC 5 input
ADC6	62	I	Analog	ADC 6 input
ADC7	61	I	Analog	ADC 7 input
CCP0	57	I/O	TTL	Capture/Compare/PWM 0
CCP1	14	I/O	TTL	Capture/Compare/PWM 1
CCP2	11	I/O	TTL	Capture/Compare/PWM 2
CCP3	15	I/O	TTL	Capture/Compare/PWM 3
CCP4	16	I/O	TTL	Capture/Compare/PWM 4
CCP5	56	I/O	TTL	Capture/Compare/PWM 5
GND	10	-	Power	Ground reference for logic and I/O pins.
GND	13	-	Power	Ground reference for logic and I/O pins.
GND	24	-	Power	Ground reference for logic and I/O pins.
GND	29	-	Power	Ground reference for logic and I/O pins.
GND	36	-	Power	Ground reference for logic and I/O pins.
GND	39	-	Power	Ground reference for logic and I/O pins.
GND	44	-	Power	Ground reference for logic and I/O pins.
GND	53	-	Power	Ground reference for logic and I/O pins.
GND	60	-	Power	Ground reference for logic and I/O pins.
GNDA	4	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.

Pin Name	Pin Number	Pin Type	Buffer Type	Description
$\overline{\text{HIB}}$	33	O	OD	An output that indicates the processor is in hibernate mode.
I2C0SCL	47	I/O	OD	I2C module 0 clock
I2C0SDA	27	I/O	OD	I2C module 0 data
I2C1SCL	25	I/O	OD	I2C module 1 clock
I2C1SDA	26	I/O	OD	I2C module 1 data
LDO	7	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 $\mu\text{F}$ or greater. The LDO pin must also be connected to the VDD25 pins at the board level in addition to the decoupling capacitor(s).
NC	45	-	-	No connect. Leave the pin electrically unconnected/isolated.
NC	46	-	-	No connect. Leave the pin electrically unconnected/isolated.
NC	48	-	-	No connect. Leave the pin electrically unconnected/isolated.
NMI	55	I	TTL	Non maskable interrupt
OSC0	30	I	Analog	Main oscillator crystal input or an external clock reference input.
OSC1	31	O	Analog	Main oscillator crystal output.
PA0	17	I/O	TTL	GPIO port A bit 0
PA1	18	I/O	TTL	GPIO port A bit 1
PA2	19	I/O	TTL	GPIO port A bit 2
PA3	20	I/O	TTL	GPIO port A bit 3
PA4	21	I/O	TTL	GPIO port A bit 4
PA5	22	I/O	TTL	GPIO port A bit 5
PA6	25	I/O	TTL	GPIO port A bit 6
PA7	26	I/O	TTL	GPIO port A bit 7
PB0	41	I/O	TTL	GPIO port B bit 0
PB1	42	I/O	TTL	GPIO port B bit 1
PB2	47	I/O	TTL	GPIO port B bit 2
PB3	27	I/O	TTL	GPIO port B bit 3
PB4	58	I/O	TTL	GPIO port B bit 4
PB5	57	I/O	TTL	GPIO port B bit 5
PB6	56	I/O	TTL	GPIO port B bit 6
PB7	55	I/O	TTL	GPIO port B bit 7
PC0	52	I/O	TTL	GPIO port C bit 0
PC1	51	I/O	TTL	GPIO port C bit 1
PC2	50	I/O	TTL	GPIO port C bit 2
PC3	49	I/O	TTL	GPIO port C bit 3
PC4	11	I/O	TTL	GPIO port C bit 4
PC5	14	I/O	TTL	GPIO port C bit 5
PC6	15	I/O	TTL	GPIO port C bit 6
PC7	16	I/O	TTL	GPIO port C bit 7

Pin Name	Pin Number	Pin Type	Buffer Type	Description
PD0	61	I/O	Analog	GPIO port D bit 0
PD1	62	I/O	Analog	GPIO port D bit 1
PD2	63	I/O	Analog	GPIO port D bit 2
PD3	64	I/O	Analog	GPIO port D bit 3
PE0	6	I/O	Analog	GPIO port E bit 0
PE1	5	I/O	Analog	GPIO port E bit 1
PE2	2	I/O	Analog	GPIO port E bit 2
PE3	1	I/O	Analog	GPIO port E bit 3
PE4	8	I/O	TTL	GPIO port E bit 4
$\overline{\text{RST}}$	40	I/O	TTL	System reset input.
SSI0Clk	19	I/O	TTL	SSI module 0 clock
SSI0Fss	20	I/O	TTL	SSI module 0 frame
SSI0Rx	21	I	TTL	SSI module 0 receive
SSI0Tx	22	O	TTL	SSI module 0 transmit
SWCLK	52	I	TTL	JTAG/SWD CLK
SWDIO	51	I/O	TTL	JTAG TMS and SWDIO
SWO	49	O	TTL	JTAG TDO and SWO
TCK	52	I	TTL	JTAG/SWD CLK
TDI	50	I	TTL	JTAG TDI
TDO	49	O	TTL	JTAG TDO and SWO
TMS	51	I/O	TTL	JTAG TMS and SWDIO
U0Rx	17	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
U0Tx	18	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
U1Rx	41	I	TTL	UART 1 receive
U1Tx	42	O	TTL	UART 1 transmit
U2Rx	58	I	TTL	UART 2 receive
U2Tx	8	O	TTL	UART2 transmit
VBAT	37	-	Power	Power source for the Hibernation Module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation Module power-source supply.
VDD	12	-	Power	Positive supply for I/O and some logic.
VDD	28	-	Power	Positive supply for I/O and some logic.
VDD	43	-	Power	Positive supply for I/O and some logic.
VDD	59	-	Power	Positive supply for I/O and some logic.
VDD25	9	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VDD25	23	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.

Pin Name	Pin Number	Pin Type	Buffer Type	Description
VDD25	38	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VDD25	54	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VDDA	3	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions.
$\overline{\text{WAKE}}$	32	I	-	An external input that brings the processor out of hibernate mode when asserted.
XOSC0	34	I	Analog	Hibernation Module oscillator crystal input or an external clock reference input. Note that this is either a 4.19-MHz crystal or a 32.768-kHz oscillator for the Hibernation Module RTC. See the <code>CLKSEL</code> bit in the <code>HIBCTL</code> register.
XOSC1	35	O	Analog	Hibernation Module oscillator crystal output.

Table 18-3. Signals by Function, Except for GPIO

Function	Pin Name	Pin Number	Pin Type	Buffer Type	Description
ADC	ADC0	1	I	Analog	ADC 0 input
	ADC1	2	I	Analog	ADC 1 input
	ADC2	5	I	Analog	ADC 2 input
	ADC3	6	I	Analog	ADC 3 input
	ADC4	64	I	Analog	ADC 4 input
	ADC5	63	I	Analog	ADC 5 input
	ADC6	62	I	Analog	ADC 6 input
	ADC7	61	I	Analog	ADC 7 input
Controller Area Network	U2Tx	8	O	TTL	UART2 transmit
General-Purpose Timers	CCP0	57	I/O	TTL	Capture/Compare/PWM 0
	CCP1	14	I/O	TTL	Capture/Compare/PWM 1
	CCP2	11	I/O	TTL	Capture/Compare/PWM 2
	CCP3	15	I/O	TTL	Capture/Compare/PWM 3
	CCP4	16	I/O	TTL	Capture/Compare/PWM 4
	CCP5	56	I/O	TTL	Capture/Compare/PWM 5
I2C	I2C0SCL	47	I/O	OD	I2C module 0 clock
	I2C0SDA	27	I/O	OD	I2C module 0 data
	I2C1SCL	25	I/O	OD	I2C module 1 clock
	I2C1SDA	26	I/O	OD	I2C module 1 data
JTAG/SWD/SWO	SWCLK	52	I	TTL	JTAG/SWD CLK
	SWDIO	51	I/O	TTL	JTAG TMS and SWDIO
	SWO	49	O	TTL	JTAG TDO and SWO

Function	Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TCK	52	I	TTL	JTAG/SWD CLK
	TDI	50	I	TTL	JTAG TDI
	TDO	49	O	TTL	JTAG TDO and SWO
	TMS	51	I/O	TTL	JTAG TMS and SWDIO
Power	GND	10	-	Power	Ground reference for logic and I/O pins.
	GND	13	-	Power	Ground reference for logic and I/O pins.
	GND	24	-	Power	Ground reference for logic and I/O pins.
	GND	29	-	Power	Ground reference for logic and I/O pins.
	GND	36	-	Power	Ground reference for logic and I/O pins.
	GND	39	-	Power	Ground reference for logic and I/O pins.
	GND	44	-	Power	Ground reference for logic and I/O pins.
	GND	53	-	Power	Ground reference for logic and I/O pins.
	GND	60	-	Power	Ground reference for logic and I/O pins.
	GNDA	4	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
	HTB	33	O	OD	An output that indicates the processor is in hibernate mode.
	LDO	7	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 $\mu$ F or greater. The LDO pin must also be connected to the VDD25 pins at the board level in addition to the decoupling capacitor(s).
	VBAT	37	-	Power	Power source for the Hibernation Module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation Module power-source supply.
	VDD	12	-	Power	Positive supply for I/O and some logic.
	VDD	28	-	Power	Positive supply for I/O and some logic.
	VDD	43	-	Power	Positive supply for I/O and some logic.
	VDD	59	-	Power	Positive supply for I/O and some logic.
	VDD25	9	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
	VDD25	23	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
	VDD25	38	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VDD25	54	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.	
VDDA	3	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions.	
	$\overline{\text{WAKE}}$	32	I	-	An external input that brings the processor out of hibernate mode when asserted.
SSI	SSI0Clk	19	I/O	TTL	SSI module 0 clock



Function	Pin Name	Pin Number	Pin Type	Buffer Type	Description
	SSIOFss	20	I/O	TTL	SSI module 0 frame
	SSIORx	21	I	TTL	SSI module 0 receive
	SSIOTx	22	O	TTL	SSI module 0 transmit
System Control & Clocks	NMI	55	I	TTL	Non maskable interrupt
	OSC0	30	I	Analog	Main oscillator crystal input or an external clock reference input.
	OSC1	31	O	Analog	Main oscillator crystal output.
	RST	40	I/O	TTL	System reset input.
	XOSC0	34	I	Analog	Hibernation Module oscillator crystal input or an external clock reference input. Note that this is either a 4.19-MHz crystal or a 32.768-kHz oscillator for the Hibernation Module RTC. See the CLKSEL bit in the HIBCTL register.
	XOSC1	35	O	Analog	Hibernation Module oscillator crystal output.
UART	U0Rx	17	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
	U0Tx	18	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
	U1Rx	41	I	TTL	UART 1 receive
	U1Tx	42	O	TTL	UART 1 transmit
	U2Rx	58	I	TTL	UART 2 receive

Table 18-4. GPIO Pins and Alternate Functions

GPIO Pin	Pin Number	Multiplexed Function	Multiplexed Function
PA0	17	U0Rx	
PA1	18	U0Tx	
PA2	19	SSIOClk	
PA3	20	SSIOFss	
PA4	21	SSIORx	
PA5	22	SSIOTx	
PA6	25	I2C1SCL	
PA7	26	I2C1SDA	
PB0	41	U1Rx	
PB1	42	U1Tx	
PB2	47	I2C0SCL	
PB3	27	I2C0SDA	
PB4	58	U2Rx	
PB5	57	CCP0	
PB6	56	CCP5	
PB7	55	NMI	
PC0	52	TCK	SWCLK
PC1	51	TMS	SWDIO
PC2	50	TDI	
PC3	49	TDO	SWO

<b>GPIO Pin</b>	<b>Pin Number</b>	<b>Multiplexed Function</b>	<b>Multiplexed Function</b>
PC4	11	CCP2	
PC5	14	CCP1	
PC6	15	CCP3	
PC7	16	CCP4	
PD0	61	ADC7	
PD1	62	ADC6	
PD2	63	ADC5	
PD3	64	ADC4	
PE0	6	ADC3	
PE1	5	ADC2	
PE2	2	ADC1	
PE3	1	ADC0	
PE4	8	U2Tx	

## 19 Operating Characteristics

**Table 19-1. Temperature Characteristics**

Characteristic <sup>a</sup>	Symbol	Value	Unit
Industrial operating temperature range	$T_A$	-40 to +85	°C

a. Maximum storage temperature is 150°C.

**Table 19-2. Thermal Characteristics**

Characteristic	Symbol	Value	Unit
Thermal resistance (junction to ambient) <sup>a</sup>	$\Theta_{JA}$	37	°C/W
Average junction temperature <sup>b</sup>	$T_J$	$T_A + (P_{AVG} \cdot \Theta_{JA})$	°C

a. Junction to ambient thermal resistance  $\Theta_{JA}$  numbers are determined by a package simulator.

b. Power dissipation is a function of temperature.

## 20 Electrical Characteristics

### 20.1 DC Characteristics

#### 20.1.1 Maximum Ratings

The maximum ratings are the limits to which the device can be subjected without permanently damaging the device.

**Note:** The device is not guaranteed to operate properly at the maximum ratings.

**Table 20-1. Maximum Ratings**

Characteristic <sup>a</sup>	Symbol	Value		Unit
		Min	Max	
I/O supply voltage ( $V_{DD}$ )	$V_{DD}$	0	4	V
Core supply voltage ( $V_{DD25}$ )	$V_{DD25}$	0	3	V
Analog supply voltage ( $V_{DDA}$ )	$V_{DDA}$	0	4	V
Battery supply voltage ( $V_{BAT}$ )	$V_{BAT}$	0	4	V
Input voltage	$V_{IN}$	-0.3	5.5	V
Maximum current per output pins	I	-	25	mA

a. Voltages are measured with respect to GND.

**Important:** This device contains circuitry to protect the inputs against damage due to high-static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (for example, either GND or  $V_{DD}$ ).

#### 20.1.2 Recommended DC Operating Conditions

For special high-current applications, the GPIO output buffers may be used with the following restrictions. With the GPIO pins configured as 8-mA output drivers, a total of four GPIO outputs may be used to sink current loads up to 18 mA each. At 18-mA sink current loading, the  $V_{OL}$  value is specified as 1.2 V. The high-current GPIO package pins must be selected such that there are only a maximum of two per side of the physical package with the total number of high-current GPIO outputs not exceeding four for the entire package.

**Table 20-2. Recommended DC Operating Conditions**

Parameter	Parameter Name	Min	Nom	Max	Unit
$V_{DD}$	I/O supply voltage	3.0	3.3	3.6	V
$V_{DD25}$	Core supply voltage	2.25	2.5	2.75	V
$V_{DDA}$	Analog supply voltage	3.0	3.3	3.6	V
$V_{BAT}$	Battery supply voltage	2.3	3.0	3.6	V
$V_{IH}$	High-level input voltage	2.0	-	5.0	V
$V_{IL}$	Low-level input voltage	-0.3	-	1.3	V
$V_{SIH}$	High-level input voltage for Schmitt trigger inputs	$0.8 * V_{DD}$	-	$V_{DD}$	V
$V_{SIL}$	Low-level input voltage for Schmitt trigger inputs	0	-	$0.2 * V_{DD}$	V

Parameter	Parameter Name	Min	Nom	Max	Unit
$V_{OH}^a$	High-level output voltage	2.4	-	-	V
$V_{OL}^a$	Low-level output voltage	-	-	0.4	V
$I_{OH}$	High-level source current, $V_{OH}=2.4$ V				
	2-mA Drive	2.0	-	-	mA
	4-mA Drive	4.0	-	-	mA
	8-mA Drive	8.0	-	-	mA
$I_{OL}$	Low-level sink current, $V_{OL}=0.4$ V				
	2-mA Drive	2.0	-	-	mA
	4-mA Drive	4.0	-	-	mA
	8-mA Drive	8.0	-	-	mA

a.  $V_{OL}$  and  $V_{OH}$  shift to 1.2 V when using high-current GPIOs.

### 20.1.3 On-Chip Low Drop-Out (LDO) Regulator Characteristics

Table 20-3. LDO Regulator Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
$V_{LDOOUT}$	Programmable internal (logic) power supply output value	2.25	2.5	2.75	V
	Output voltage accuracy	-	2%	-	%
$t_{PON}$	Power-on time	-	-	100	$\mu$ s
$t_{ON}$	Time on	-	-	200	$\mu$ s
$t_{OFF}$	Time off	-	-	100	$\mu$ s
$V_{STEP}$	Step programming incremental voltage	-	50	-	mV
$C_{LDO}$	External filter capacitor size for internal power supply	1.0	-	3.0	$\mu$ F

### 20.1.4 Power Specifications

The power measurements specified in the tables that follow are run on the core processor using SRAM with the following specifications (except as noted):

- $V_{DD} = 3.3$  V
- $V_{DD25} = 2.50$  V
- $V_{BAT} = 3.0$  V
- $V_{DDA} = 3.3$  V
- Temperature = 25°C
- Clock Source (MOSC) = 3.579545 MHz Crystal Oscillator
- Main oscillator (MOSC) = enabled
- Internal oscillator (IOSC) = disabled

Table 20-4. Detailed Power Specifications

Parameter	Parameter Name	Conditions	3.3 V $V_{DD}$ , $V_{DDA}$		2.5 V $V_{DD25}$		3.0 V $V_{BAT}$		Unit
			Nom	Max	Nom	Max	Nom	Max	
$I_{DD\_RUN}$	Run mode 1 (Flash loop)	$V_{DD25} = 2.50\text{ V}$ Code= while(1){} executed in Flash Peripherals = All ON System Clock = 50 MHz (with PLL)	9.5	pending <sup>a</sup>	108	pending <sup>a</sup>	0	pending <sup>a</sup>	mA
	Run mode 2 (Flash loop)	$V_{DD25} = 2.50\text{ V}$ Code= while(1){} executed in Flash Peripherals = All OFF System Clock = 50 MHz (with PLL)	<0.001	pending <sup>a</sup>	53	pending <sup>a</sup>	0	pending <sup>a</sup>	mA
	Run mode 1 (SRAM loop)	$V_{DD25} = 2.50\text{ V}$ Code= while(1){} executed in SRAM Peripherals = All ON System Clock = 50 MHz (with PLL)	9.5	pending <sup>a</sup>	102	pending <sup>a</sup>	0	pending <sup>a</sup>	mA
	Run mode 2 (SRAM loop)	$V_{DD25} = 2.50\text{ V}$ Code= while(1){} executed in SRAM Peripherals = All OFF System Clock = 50 MHz (with PLL)	<0.001	pending <sup>a</sup>	47	pending <sup>a</sup>	0	pending <sup>a</sup>	mA
$I_{DD\_SLEEP}$	Sleep mode	$V_{DD25} = 2.50\text{ V}$ Peripherals = All OFF System Clock = 50 MHz (with PLL)	<0.001	pending <sup>a</sup>	17	pending <sup>a</sup>	0	pending <sup>a</sup>	mA
$I_{DD\_DEEPSLEEP}$	Deep-Sleep mode	LDO = 2.25 V Peripherals = All OFF System Clock = IOS30KHZ/64	0.14	pending <sup>a</sup>	0.18	pending <sup>a</sup>	0	pending <sup>a</sup>	mA
$I_{DD\_HIBERNATE}$	Hibernate mode	$V_{BAT} = 3.0\text{ V}$ $V_{DD} = 0\text{ V}$ $V_{DD25} = 0\text{ V}$ $V_{DDA} = 0\text{ V}$ $V_{DDPHY} = 0\text{ V}$ Peripherals = All OFF System Clock = OFF Hibernate Module = 32 kHz	0	0	0	0	16	pending <sup>a</sup>	$\mu\text{A}$

a. Pending characterization completion.

## 20.1.5 Flash Memory Characteristics

**Table 20-5. Flash Memory Characteristics**

Parameter	Parameter Name	Min	Nom	Max	Unit
PE <sub>CYC</sub>	Number of guaranteed program/erase cycles before failure <sup>a</sup>	10,000	100,000	-	cycles
T <sub>RET</sub>	Data retention at average operating temperature of 85°C	10	-	-	years
T <sub>PROG</sub>	Word program time	20	-	-	µs
T <sub>ERASE</sub>	Page erase time	20	-	-	ms
T <sub>ME</sub>	Mass erase time	200	-	-	ms

a. A program/erase cycle is defined as switching the bits from 1-> 0 -> 1.

## 20.1.6 Hibernation

**Table 20-6. Hibernation Module DC Characteristics**

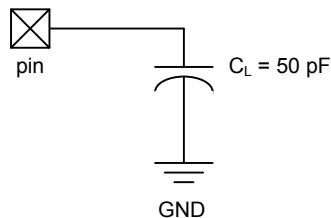
Parameter	Parameter Name	Value	Unit
V <sub>LOWBAT</sub>	Low battery detect voltage	2.35	V

## 20.2 AC Characteristics

### 20.2.1 Load Conditions

Unless otherwise specified, the following conditions are true for all timing measurements. Timing measurements are for 4-mA drive strength.

**Figure 20-1. Load Conditions**



### 20.2.2 Clocks

**Table 20-7. Phase Locked Loop (PLL) Characteristics**

Parameter	Parameter Name	Min	Nom	Max	Unit
f <sub>ref_crystal</sub>	Crystal reference <sup>a</sup>	3.579545	-	16.384	MHz
f <sub>ref_ext</sub>	External clock reference <sup>a</sup>	3.579545	-	16.384	MHz
f <sub>pll</sub>	PLL frequency <sup>b</sup>	-	400	-	MHz
T <sub>READY</sub>	PLL lock time	-	-	0.5	ms

a. The exact value is determined by the crystal value programmed into the XTAL field of the **Run-Mode Clock Configuration (RCC)** register.

b. PLL frequency is automatically calculated by the hardware based on the XTAL field of the **RCC** register.

Table 20-8. Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
f <sub>IOSC</sub>	Internal 12 MHz oscillator frequency	8.4	12	15.6	MHz
f <sub>IOSC30KHZ</sub>	Internal 30 KHz oscillator frequency	21	30	39	KHz
f <sub>XOSC</sub>	Hibernation module oscillator frequency	-	4.194304	-	MHz
f <sub>XOSC_XTAL</sub>	Crystal reference for hibernation oscillator	-	4.194304	-	MHz
f <sub>XOSC_EXT</sub>	External clock reference for hibernation module	-	32.768	-	KHz
f <sub>MOSC</sub>	Main oscillator frequency	1	-	16.384	MHz
t <sub>MOSC_per</sub>	Main oscillator period	61	-	1000	ns
f <sub>ref_crystal_bypass</sub> <sup>a</sup>	Crystal reference using the main oscillator (PLL in BYPASS mode)	1	-	16.384	MHz
f <sub>ref_ext_bypass</sub>	External clock reference (PLL in BYPASS mode) <sup>a</sup>	0	-	50	MHz
f <sub>system_clock</sub>	System clock	0	-	50	MHz

a. The ADC must be clocked from the PLL or directly from a 14-MHz to 18-MHz clock source to operate properly.

Table 20-9. Crystal Characteristics

Parameter Name	Value						Units
Frequency	16	12	8	6	4	3.5	MHz
Frequency tolerance	±50	±50	±50	±50	±50	±50	ppm
Aging	±5	±5	±5	±5	±5	±5	ppm/yr
Oscillation mode	Parallel	Parallel	Parallel	Parallel	Parallel	Parallel	-
Temperature stability (-40°C to 85°C)	±25	±25	±25	±25	±25	±25	ppm
Motional capacitance (typ)	13.9	18.5	27.8	37.0	55.6	63.5	pF
Motional inductance (typ)	7.15	9.5	14.3	19.1	28.6	32.7	mH
Equivalent series resistance (max)	80	100	120	160	200	220	Ω
Shunt capacitance (max)	10	10	10	10	10	10	pF
Load capacitance (typ)	16	16	16	16	16	16	pF
Drive level (typ)	100	100	100	100	100	100	μW

## 20.2.3 Analog-to-Digital Converter

Table 20-10. ADC Characteristics<sup>a</sup>

Parameter	Parameter Name	Min	Nom	Max	Unit
V <sub>ADCIN</sub>	Maximum single-ended, full-scale analog input voltage	-	-	3.0	V
	Minimum single-ended, full-scale analog input voltage	-	-	0	V
	Maximum differential, full-scale analog input voltage	-	-	1.5	V
	Minimum differential, full-scale analog input voltage	-	-	-1.5	V
C <sub>ADCIN</sub>	Equivalent input capacitance	-	1	-	pF
N	Resolution	-	10	-	bits
f <sub>ADC</sub>	ADC internal clock frequency	7	8	9	MHz
t <sub>ADCCONV</sub>	Conversion time	-	-	16	t <sub>ADC</sub> cycles <sup>b</sup>
f <sub>ADCCONV</sub>	Conversion rate	438	500	563	k samples/s
INL	Integral nonlinearity	-	-	±1	LSB
DNL	Differential nonlinearity	-	-	±1	LSB



Parameter	Parameter Name	Min	Nom	Max	Unit
OFF	Offset	-	-	±1	LSB
GAIN	Gain	-	-	±1	LSB

a. The ADC reference voltage is 3.0 V. This reference voltage is internally generated from the 3.3 VDDA supply by a band gap circuit.

b.  $t_{\text{ADC}} = 1/f_{\text{ADC clock}}$

## 20.2.4 I<sup>2</sup>C

Table 20-11. I<sup>2</sup>C Characteristics

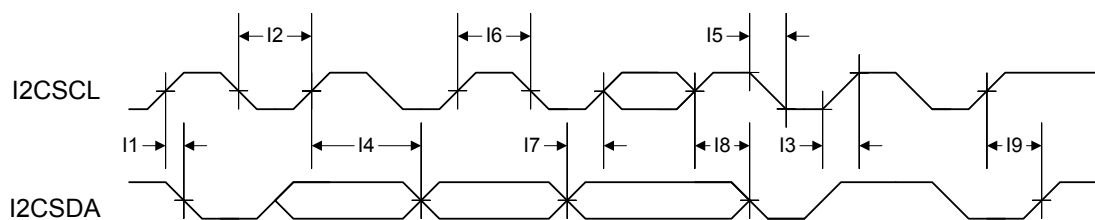
Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
I1 <sup>a</sup>	$t_{\text{SCH}}$	Start condition hold time	36	-	-	system clocks
I2 <sup>a</sup>	$t_{\text{LP}}$	Clock Low period	36	-	-	system clocks
I3 <sup>b</sup>	$t_{\text{SRT}}$	I <sup>2</sup> C <sub>SCL</sub> /I <sup>2</sup> C <sub>SDA</sub> rise time ( $V_{\text{IL}}=0.5\text{ V}$ to $V_{\text{IH}}=2.4\text{ V}$ )	-	-	(see note b)	ns
I4 <sup>a</sup>	$t_{\text{DH}}$	Data hold time	2	-	-	system clocks
I5 <sup>c</sup>	$t_{\text{SFT}}$	I <sup>2</sup> C <sub>SCL</sub> /I <sup>2</sup> C <sub>SDA</sub> fall time ( $V_{\text{IH}}=2.4\text{ V}$ to $V_{\text{IL}}=0.5\text{ V}$ )	-	9	10	ns
I6 <sup>a</sup>	$t_{\text{HT}}$	Clock High time	24	-	-	system clocks
I7 <sup>a</sup>	$t_{\text{DS}}$	Data setup time	18	-	-	system clocks
I8 <sup>a</sup>	$t_{\text{SCSR}}$	Start condition setup time (for repeated start condition only)	36	-	-	system clocks
I9 <sup>a</sup>	$t_{\text{SCS}}$	Stop condition setup time	24	-	-	system clocks

a. Values depend on the value programmed into the TPR bit in the I<sup>2</sup>C Master Timer Period (I2CMTPR) register; a TPR programmed for the maximum I<sup>2</sup>C<sub>SCL</sub> frequency (TPR=0x2) results in a minimum output timing as shown in the table above. The I<sup>2</sup>C interface is designed to scale the actual data transition time to move it to the middle of the I<sup>2</sup>C<sub>SCL</sub> Low period. The actual position is affected by the value programmed into the TPR; however, the numbers given in the above values are minimum values.

b. Because I<sup>2</sup>C<sub>SCL</sub> and I<sup>2</sup>C<sub>SDA</sub> are open-drain-type outputs, which the controller can only actively drive Low, the time I<sup>2</sup>C<sub>SCL</sub> or I<sup>2</sup>C<sub>SDA</sub> takes to reach a high level depends on external signal capacitance and pull-up resistor values.

c. Specified at a nominal 50 pF load.

Figure 20-2. I<sup>2</sup>C Timing



## 20.2.5 Hibernation Module

The Hibernation Module requires special system implementation considerations since it is intended to power-down all other sections of its host device. The system power-supply distribution and interfaces to the device must be driven to 0 V<sub>DC</sub> or powered down with the same external voltage regulator controlled by  $\overline{\text{HIB}}$ .

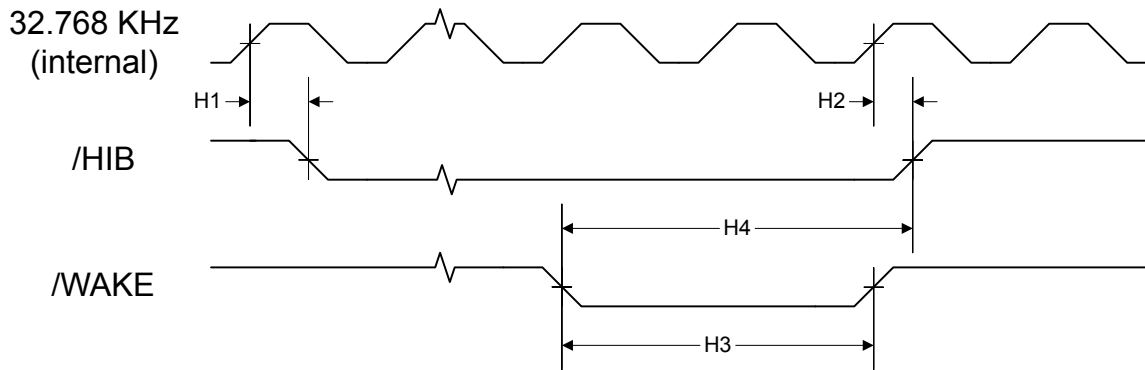
The external voltage regulators controlled by  $\overline{\text{HIB}}$  must have a settling time of 250  $\mu\text{s}$  or less.

**Table 20-12. Hibernation Module AC Characteristics**

Parameter No	Parameter	Parameter Name	Min	Nom	Max	Unit
H1	$t_{HIB\_LOW}$	Internal 32.768 KHz clock reference rising edge to /HIB asserted	-	200	-	$\mu s$
H2	$t_{HIB\_HIGH}$	Internal 32.768 KHz clock reference rising edge to /HIB deasserted	-	30	-	$\mu s$
H3	$t_{WAKE\_ASSERT}$	/WAKE assertion time	62	-	-	$\mu s$
H4	$t_{WAKETOHIB}$	/WAKE assert to /HIB desassert	62	-	124	$\mu s$
H5	$t_{XOSC\_SETTLE}$	XOSC settling time <sup>a</sup>	20	-	-	ms
H6	$t_{HIB\_REG\_WRITE}$	Time for a write to non-volatile registers in HIB module to complete	92	-	-	$\mu s$
H7	$t_{HIB\_TO\_VDD}$	$\overline{HIB}$ deassert to VDD and VDD25 at minimum operational level	-	-	250	$\mu s$
H8	$R_{WAKEPU}$	$\overline{WAKE}$ internal pull-up resistor	-	200	-	k $\Omega$

a. This parameter is highly sensitive to PCB layout and trace lengths, which may make this parameter time longer. Care must be taken in PCB design to minimize trace lengths and RLC (resistance, inductance, capacitance).

**Figure 20-3. Hibernation Module Timing**



## 20.2.6 Synchronous Serial Interface (SSI)

**Table 20-13. SSI Characteristics**

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
S1	$t_{clk\_per}$	SSIClk cycle time	2	-	65024	system clocks
S2	$t_{clk\_high}$	SSIClk high time	-	1/2	-	t clk_per
S3	$t_{clk\_low}$	SSIClk low time	-	1/2	-	t clk_per
S4	$t_{clkrf}$	SSIClk rise/fall time	-	7.4	26	ns
S5	$t_{DMd}$	Data from master valid delay time	0	-	20	ns
S6	$t_{DMs}$	Data from master setup time	20	-	-	ns
S7	$t_{DMh}$	Data from master hold time	40	-	-	ns
S8	$t_{DSs}$	Data from slave setup time	20	-	-	ns
S9	$t_{DSh}$	Data from slave hold time	40	-	-	ns

Figure 20-4. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement

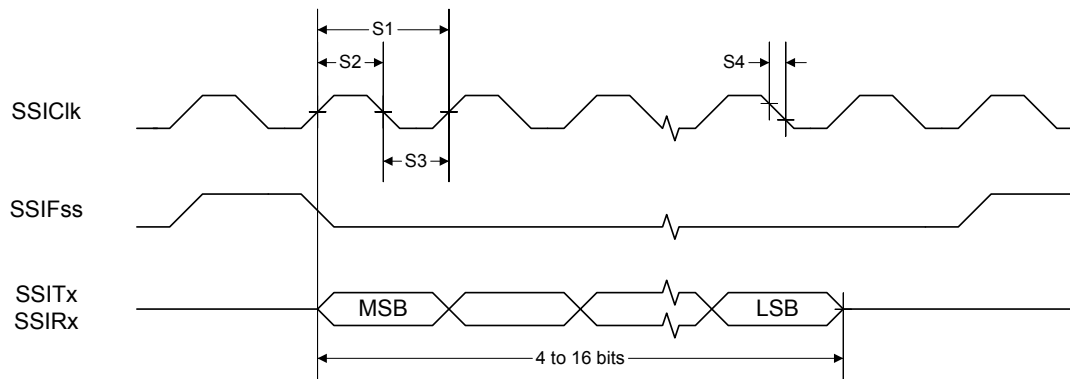


Figure 20-5. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer

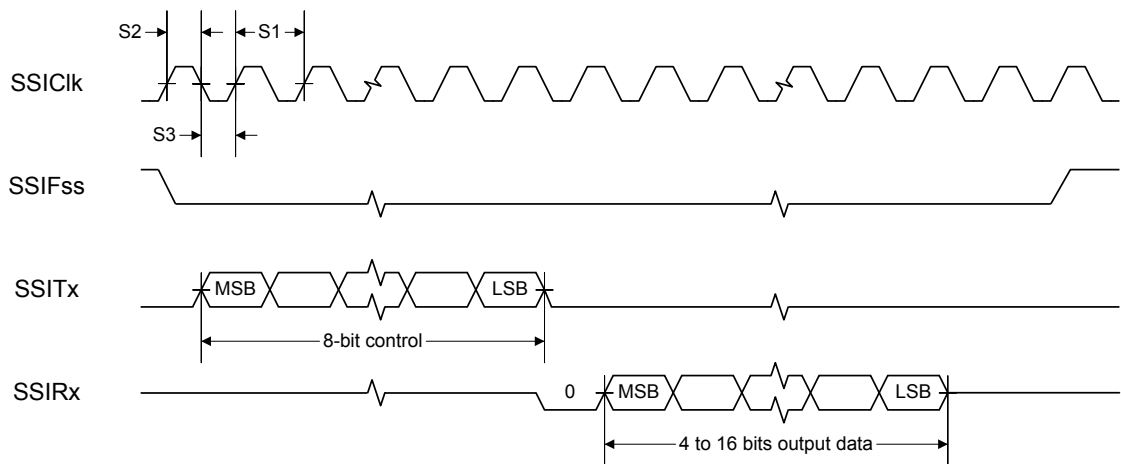
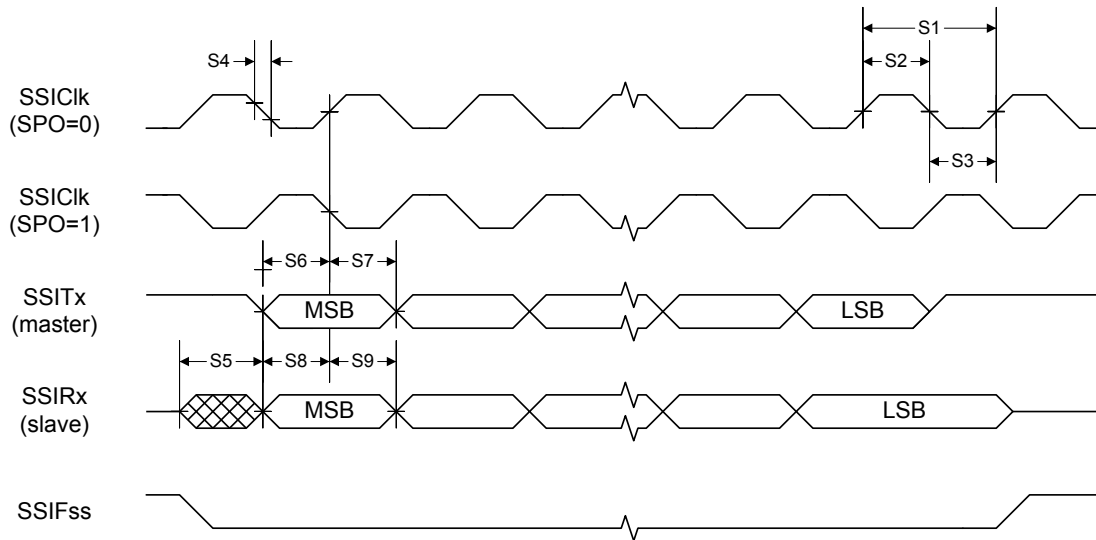


Figure 20-6. SSI Timing for SPI Frame Format (FRF=00), with SPH=1



## 20.2.7 JTAG and Boundary Scan

Table 20-14. JTAG Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
J1	$f_{TCK}$	TCK operational clock frequency	0	-	10	MHz
J2	$t_{TCK}$	TCK operational clock period	100	-	-	ns
J3	$t_{TCK\_LOW}$	TCK clock Low time	-	$t_{TCK}$	-	ns
J4	$t_{TCK\_HIGH}$	TCK clock High time	-	$t_{TCK}$	-	ns
J5	$t_{TCK\_R}$	TCK rise time	0	-	10	ns
J6	$t_{TCK\_F}$	TCK fall time	0	-	10	ns
J7	$t_{TMS\_SU}$	TMS setup time to TCK rise	20	-	-	ns
J8	$t_{TMS\_HLD}$	TMS hold time from TCK rise	20	-	-	ns
J9	$t_{TDI\_SU}$	TDI setup time to TCK rise	25	-	-	ns
J10	$t_{TDI\_HLD}$	TDI hold time from TCK rise	25	-	-	ns
J11	$t_{TDO\_ZDV}$	TCK fall to Data Valid from High-Z	-	23	35	ns
		2-mA drive		15	26	ns
		4-mA drive		14	25	ns
		8-mA drive		18	29	ns
J12	$t_{TDO\_DV}$	TCK fall to Data Valid from Data Valid	-	21	35	ns
		2-mA drive		14	25	ns
		4-mA drive		13	24	ns
		8-mA drive		18	28	ns

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
J13 $t_{TDO\_DVZ}$	TCK fall to High-Z from Data Valid	2-mA drive	-	9	11	ns
		4-mA drive		7	9	ns
		8-mA drive		6	8	ns
		8-mA drive with slew rate control		7	9	ns

Figure 20-7. JTAG Test Clock Input Timing

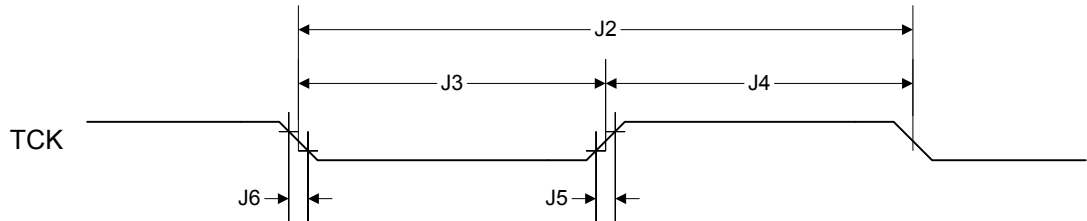
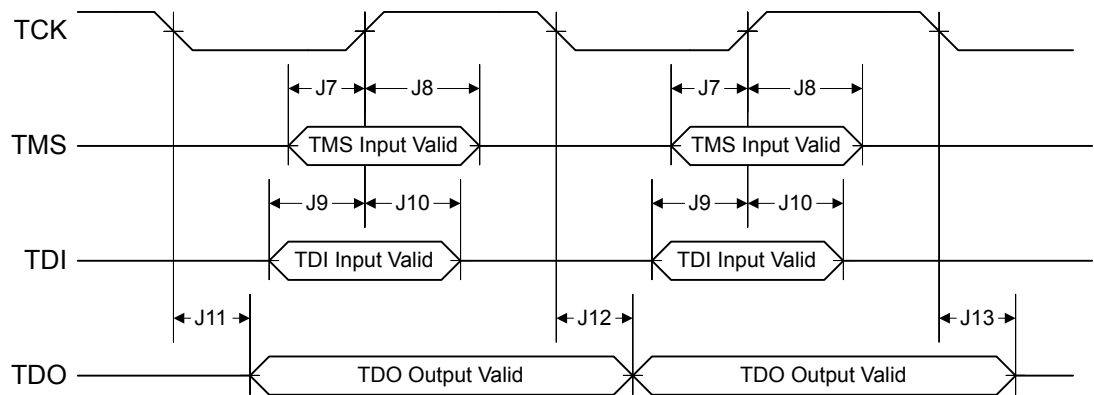


Figure 20-8. JTAG Test Access Port (TAP) Timing



### 20.2.8 General-Purpose I/O

**Note:** All GPIOs are 5 V-tolerant.

Table 20-15. GPIO Characteristics

Parameter	Parameter Name	Condition	Min	Nom	Max	Unit
$t_{GPIOR}$	GPIO Rise Time (from 20% to 80% of $V_{DD}$ )	2-mA drive	-	17	26	ns
		4-mA drive		9	13	ns
		8-mA drive		6	9	ns
		8-mA drive with slew rate control		10	12	ns
$t_{GPIOF}$	GPIO Fall Time (from 80% to 20% of $V_{DD}$ )	2-mA drive	-	17	25	ns
		4-mA drive		8	12	ns
		8-mA drive		6	10	ns
		8-mA drive with slew rate control		11	13	ns
$R_{GPIOPU}$	GPIO internal pull-up resistor	Pull-up enabled	50	-	110	k $\Omega$

Parameter	Parameter Name	Condition	Min	Nom	Max	Unit
R <sub>GPIOPD</sub>	GPIO internal pull-down resistor	Pull-down enabled	55	-	180	kΩ

## 20.2.9 Reset

Table 20-16. Reset Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
R1	V <sub>TH</sub>	Reset threshold	-	2.0	-	V
R2	V <sub>BTH</sub>	Brown-Out threshold	2.85	2.9	2.95	V
R3	T <sub>POR</sub>	Power-On Reset timeout	-	10	-	ms
R4	T <sub>BOR</sub>	Brown-Out timeout	-	500	-	μs
R5	T <sub>IRPOR</sub>	Internal reset timeout after POR	6	-	11	ms
R6	T <sub>IRBOR</sub>	Internal reset timeout after BOR <sup>a</sup>	0	-	1	μs
R7	T <sub>IRHWR</sub>	Internal reset timeout after hardware reset ( $\overline{\text{RST}}$ pin)	0	-	1	ms
R8	T <sub>IRSWR</sub>	Internal reset timeout after software-initiated system reset <sup>a</sup>	2.5	-	20	μs
R9	T <sub>IRWDR</sub>	Internal reset timeout after watchdog reset <sup>a</sup>	2.5	-	20	μs
R10	T <sub>VDDRISE</sub>	Supply voltage (V <sub>DD</sub> ) rise time (0V-3.3V)	-	-	250	ms
R11	T <sub>MIN</sub>	Minimum $\overline{\text{RST}}$ pulse width	2	-	-	μs

a.  $20 * t_{\text{MOSC\_per}}$

Figure 20-9. External Reset Timing ( $\overline{\text{RST}}$ )

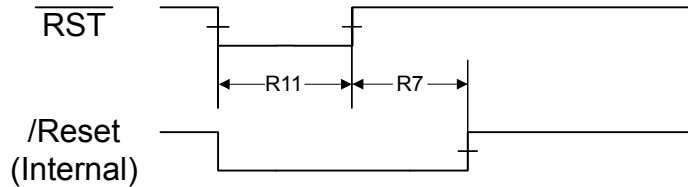


Figure 20-10. Power-On Reset Timing

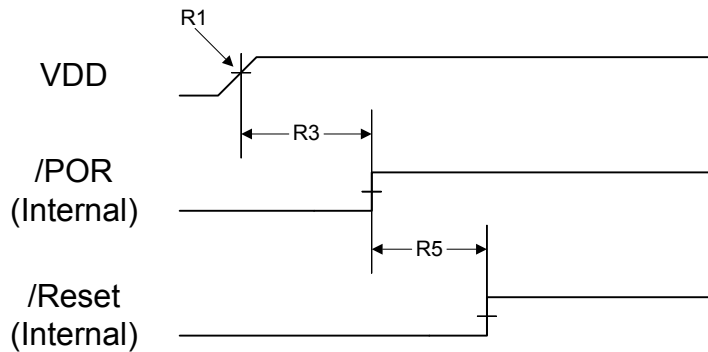


Figure 20-11. Brown-Out Reset Timing

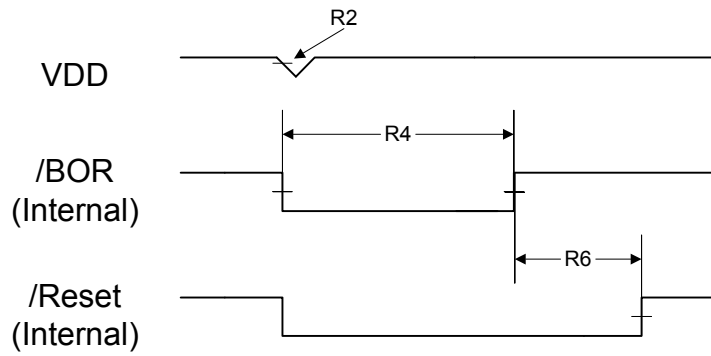


Figure 20-12. Software Reset Timing

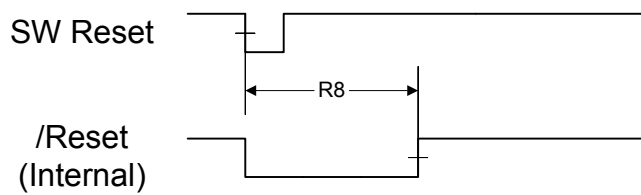
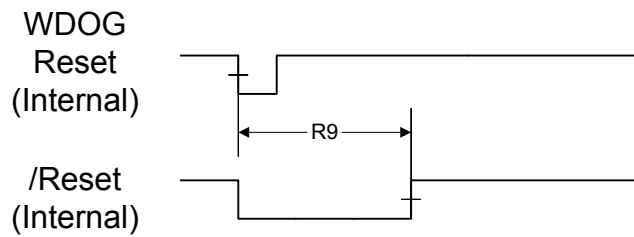
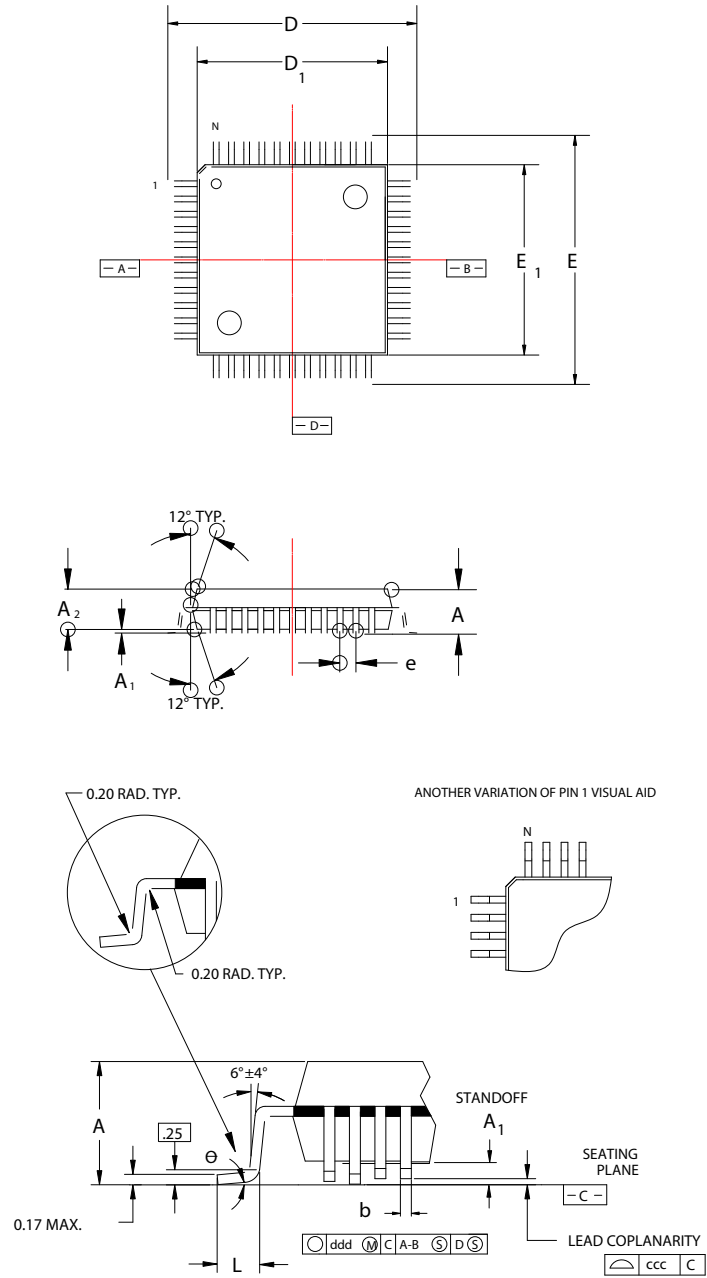


Figure 20-13. Watchdog Reset Timing



# 21 Package Information

Figure 21-1. 64-Pin LQFP Package



**Note:** The following notes apply to the package drawing.

1. All dimensions shown in mm.



2. Dimensions shown are nominal with tolerances indicated.
3. Foot length 'L' is measured at gage plane 0.25 mm above seating plane.
4. L/F: Eftec 64T Cu or equivalent, 0.127mm (0.005") thick.

Body +2.00 mm Footprint, 1.4 mm package thickness		
Symbols	Leads	64L
A	Max.	1.60
A <sub>1</sub>	-	0.05 Min./0.15 Max.
A <sub>2</sub>	±0.05	1.40
D	±0.20	12.00
D <sub>1</sub>	±0.10	10.00
E	±0.20	12.00
E <sub>1</sub>	±0.10	10.00
L	+0.15/-0.10	0.60
e	Basic	0.50
b	±0.05	0.22
θ	-	0°-7°
ddd	Max.	0.08
ccc	Max.	0.08
JEDEC Reference Drawing		MS-026
Variation Designator		BCD

## A Boot Loader

### A.1 Boot Loader

The Stellaris<sup>®</sup> boot loader is executed from the ROM when flash is empty and is used to download code to the flash memory of a device without the use of a debug interface. The boot loader uses a simple packet interface to provide synchronous communication with the device. The boot loader runs off the internal oscillator and does not enable the PLL, so its speed is determined by the speed of the internal oscillator. The UART0, SSI0 and I<sup>2</sup>C0 serial interfaces can be used. For simplicity, both the data format and communication protocol are identical for all serial interfaces.

### A.2 Interfaces

Once communication with the boot loader is established via one of the serial interfaces, that interface is used until the boot loader is reset or new code takes over. For example, once you start communicating using the SSI port, communications with the boot loader via the UART are disabled until the device is reset.

#### A.2.1 UART

The Universal Asynchronous Receivers/Transmitters (UART) communication uses a fixed serial format of 8 bits of data, no parity, and 1 stop bit. The baud rate used for communication is automatically detected by the boot loader and can be any valid baud rate supported by the host and the device. The auto detection sequence requires that the baud rate should be no more than 1/32 the internal oscillator frequency of the board that is running the boot loader (which is at least 8.4 MHz, providing support for up to 262,500 baud). This is actually the same as the hardware limitation for the maximum baud rate for any UART on a Stellaris<sup>®</sup> device which is calculated as follows:

$$\text{Max Baud Rate} = \text{System Clock Frequency} / 16$$

In order to determine the baud rate, the boot loader needs to determine the relationship between the internal oscillator and the baud rate. This is enough information for the boot loader to configure its UART to the same baud rate as the host. This automatic baud-rate detection allows the host to use any valid baud rate that it wants to communicate with the device.

The method used to perform this automatic synchronization relies on the host sending the boot loader two bytes that are both 0x55. This generates a series of pulses to the boot loader that it can use to calculate the ratios needed to program the UART to match the host's baud rate. After the host sends the pattern, it attempts to read back one byte of data from the UART. The boot loader returns the value of 0xCC to indicate successful detection of the baud rate. If this byte is not received after at least twice the time required to transfer the two bytes, the host can resend another pattern of 0x55, 0x55, and wait for the 0xCC byte again until the boot loader acknowledges that it has received a synchronization pattern correctly. For example, the time to wait for data back from the boot loader should be calculated as at least  $2 \times (20 \text{ (bits/sync)} / \text{baud rate (bits/sec)})$ . For a baud rate of 115200, this time is  $2 \times (20 / 115200)$  or 0.35 ms.

#### A.2.2 SSI

The Synchronous Serial Interface (SSI) port also uses a fixed serial format for communications, with the framing defined as Motorola format with SPH set to 1 and SPO set to 1. See "Frame Formats" on page 416 in the SSI chapter for more information on formats for this transfer protocol. Like the UART, this interface has hardware requirements that limit the maximum speed that the SSI clock can run. This allows the SSI clock to be at most 1/12 the the internal oscillator frequency of

the board running the boot loader (which is at least 8.4 MHz, providing support for up to 700 KHz).. Since the host device is the master, the SSI on the boot loader device does not need to determine the clock as it is provided directly by the host.

### A.2.3 I<sup>2</sup>C

The Inter-Integrated Circuit (I<sup>2</sup>C) port operates in slave mode with a slave address of 0x42. The I<sup>2</sup>C port will work at both 100 KHz and 400 KHz I<sup>2</sup>C clock frequency. Since the host device is the master, the I<sup>2</sup>C on the boot loader device does not need to determine the clock as it is provided directly by the host.

## A.3 Packet Handling

All communications, with the exception of the UART auto-baud, are done via defined packets that are acknowledged (ACK) or not acknowledged (NAK) by the devices. The packets use the same format for receiving and sending packets, including the method used to acknowledge successful or unsuccessful reception of a packet.

### A.3.1 Packet Format

All packets sent and received from the device use the following byte-packed format.

```
struct
{
  unsigned char ucSize;
  unsigned char ucChecksum;
  unsigned char Data[];
};
```

ucSize	The first byte received holds the total size of the transfer including the size and checksum bytes.
ucChecksum	This holds a simple checksum of the bytes in the data buffer only. The algorithm is Data[0]+Data[1]+...+ Data[ucSize-3].
Data	This is the raw data intended for the device, which is formatted in some form of command interface. There should be ucSize-2 bytes of data provided in this buffer to or from the device.

### A.3.2 Sending Packets

The actual bytes of the packet can be sent individually or all at once; the only limitation is that commands that cause flash memory access should limit the download sizes to prevent losing bytes during flash programming. This limitation is discussed further in the section that describes the boot loader command, COMMAND\_SEND\_DATA (see "COMMAND\_SEND\_DATA (0x24)" on page 517).

Once the packet has been formatted correctly by the host, it should be sent out over the UART or SSI interface. Then the host should poll the UART *or* SSI interface for the first non-zero data returned from the device. The first non-zero byte will either be an ACK (0xCC) or a NAK (0x33) byte from the device indicating the packet was received successfully (ACK) or unsuccessfully (NAK). This does not indicate that the actual contents of the command issued in the data portion of the packet were valid, just that the packet was received correctly.

### A.3.3 Receiving Packets

The boot loader sends a packet of data in the same format that it receives a packet. The boot loader may transfer leading zero data before the first actual byte of data is sent out. The first non-zero byte is the size of the packet followed by a checksum byte, and finally followed by the data itself. There is no break in the data after the first non-zero byte is sent from the boot loader. Once the device communicating with the boot loader receives all the bytes, it must either ACK or NAK the packet to indicate that the transmission was successful. The appropriate response after sending a NAK to the boot loader is to resend the command that failed and request the data again. If needed, the host may send leading zeros before sending down the ACK/NAK signal to the boot loader, as the boot loader only accepts the first non-zero data as a valid response. This zero padding is needed by the SSI interface in order to receive data to or from the boot loader.

## A.4 Commands

The next section defines the list of commands that can be sent to the boot loader. The first byte of the data should always be one of the defined commands, followed by data or parameters as determined by the command that is sent.

### A.4.1 COMMAND\_PING (0X20)

This command simply accepts the command and sets the global status to success. The format of the packet is as follows:

```
Byte[0] = 0x03;  
Byte[1] = checksum(Byte[2]);  
Byte[2] = COMMAND_PING;
```

The ping command has 3 bytes and the value for `COMMAND_PING` is 0x20 and the checksum of one byte is that same byte, making `Byte[1]` also 0x20. Since the ping command has no real return status, the receipt of an ACK can be interpreted as a successful ping to the boot loader.

### A.4.2 COMMAND\_GET\_STATUS (0x23)

This command returns the status of the last command that was issued. Typically, this command should be sent after every command to ensure that the previous command was successful or to properly respond to a failure. The command requires one byte in the data of the packet and should be followed by reading a packet with one byte of data that contains a status code. The last step is to ACK or NAK the received data so the boot loader knows that the data has been read.

```
Byte[0] = 0x03  
Byte[1] = checksum(Byte[2])  
Byte[2] = COMMAND_GET_STATUS
```

### A.4.3 COMMAND\_DOWNLOAD (0x21)

This command is sent to the boot loader to indicate where to store data and how many bytes will be sent by the `COMMAND_SEND_DATA` commands that follow. The command consists of two 32-bit values that are both transferred MSB first. The first 32-bit value is the address to start programming data into, while the second is the 32-bit size of the data that will be sent. This command also triggers an erase of the full area to be programmed so this command takes longer than other commands. This results in a longer time to receive the ACK/NAK back from the board. This command should be followed by a `COMMAND_GET_STATUS` to ensure that the Program Address and Program size are valid for the device running the boot loader.

The format of the packet to send this command is as follows:

```

Byte[0] = 11
Byte[1] = checksum(Bytes[2:10])
Byte[2] = COMMAND_DOWNLOAD
Byte[3] = Program Address [31:24]
Byte[4] = Program Address [23:16]
Byte[5] = Program Address [15:8]
Byte[6] = Program Address [7:0]
Byte[7] = Program Size [31:24]
Byte[8] = Program Size [23:16]
Byte[9] = Program Size [15:8]
Byte[10] = Program Size [7:0]

```

#### A.4.4 COMMAND\_SEND\_DATA (0x24)

This command should only follow a COMMAND\_DOWNLOAD command or another COMMAND\_SEND\_DATA command if more data is needed. Consecutive send data commands automatically increment address and continue programming from the previous location. For packets which do not contain the final portion of the downloaded data, a multiple of four bytes should always be transferred. The command terminates programming once the number of bytes indicated by the COMMAND\_DOWNLOAD command has been received. Each time this function is called it should be followed by a COMMAND\_GET\_STATUS to ensure that the data was successfully programmed into the flash. If the boot loader sends a NAK to this command, the boot loader does not increment the current address to allow retransmission of the previous data. The following example shows a COMMAND\_SEND\_DATA packet with 8 bytes of packet data:

```

Byte[0] = 11
Byte[1] = checksum(Bytes[2:10])
Byte[2] = COMMAND_SEND_DATA
Byte[3] = Data[0]
Byte[4] = Data[1]
Byte[5] = Data[2]
Byte[6] = Data[3]
Byte[7] = Data[4]
Byte[8] = Data[5]
Byte[9] = Data[6]
Byte[10] = Data[7]

```

#### A.4.5 COMMAND\_RUN (0x22)

This command is used to tell the boot loader to execute from the address passed as the parameter in this command. This command consists of a single 32-bit value that is interpreted as the address to execute. The 32-bit value is transmitted MSB first and the boot loader responds with an ACK signal back to the host device before actually executing the code at the given address. This allows the host to know that the command was received successfully and the code is now running.

```

Byte[0] = 7
Byte[1] = checksum(Bytes[2:6])
Byte[2] = COMMAND_RUN
Byte[3] = Execute Address[31:24]
Byte[4] = Execute Address[23:16]
Byte[5] = Execute Address[15:8]
Byte[6] = Execute Address[7:0]

```

#### A.4.6 COMMAND\_RESET (0x25)

This command is used to tell the boot loader device to reset. Unlike the `COMMAND_RUN` command, this allows the initial stack pointer to be read by the hardware and set up for the new code. It can also be used to reset the boot loader if a critical error occurs and the host device wants to restart communication with the boot loader.

```
Byte[0] = 3  
Byte[1] = checksum(Byte[2])  
Byte[2] = COMMAND_RESET
```

The boot loader responds with an ACK signal back to the host device before actually executing the software reset to the device running the boot loader. This allows the host to know that the command was received successfully and the part will be reset.

## B ROM DriverLib Functions

### B.1 DriverLib Functions Included in the Integrated ROM

The Peripheral Driver Library (DriverLib) APIs that are available in the integrated ROM of the Stellaris® family of devices are listed below. The detailed description of each function is available in the *Stellaris® ROM User's Guide*.

```
ROM_ADCHardwareOversampleConfigure
    // Configures the hardware oversampling factor of the ADC.

ROM_ADCIntClear
    // Clears sample sequence interrupt source.

ROM_ADCIntDisable
    // Disables a sample sequence interrupt.

ROM_ADCIntEnable
    // Enables a sample sequence interrupt.

ROM_ADCIntStatus
    // Gets the current interrupt status.

ROM_ADCProcessorTrigger
    // Causes a processor trigger for a sample sequence.

ROM_ADCSequenceConfigure
    // Configures the trigger source and priority of a sample sequence.

ROM_ADCSequenceDataGet
    // Gets the captured data for a sample sequence.

ROM_ADCSequenceDisable
    // Disables a sample sequence.

ROM_ADCSequenceEnable
    // Enables a sample sequence.

ROM_ADCSequenceOverflow
    // Determines if a sample sequence overflow occurred.

ROM_ADCSequenceOverflowClear
    // Clears the overflow condition on a sample sequence.

ROM_ADCSequenceStepConfigure
    // Configure a step of the sample sequencer.

ROM_ADCSequenceUnderflow
    // Determines if a sample sequence underflow occurred.

ROM_ADCSequenceUnderflowClear
    // Clears the underflow condition on a sample sequence.
```

ROM\_FlashErase  
// Erases a block of flash.

ROM\_FlashIntClear  
// Clears flash controller interrupt sources.

ROM\_FlashIntDisable  
// Disables individual flash controller interrupt sources.

ROM\_FlashIntEnable  
// Enables individual flash controller interrupt sources.

ROM\_FlashIntGetStatus  
// Gets the current interrupt status.

ROM\_FlashProgram  
// Programs flash.

ROM\_FlashProtectGet  
// Gets the protection setting for a block of flash.

ROM\_FlashProtectSave  
// Saves the flash protection settings.

ROM\_FlashProtectSet  
// Sets the protection setting for a block of flash.

ROM\_FlashUsecGet  
// Gets the number of processor clocks per micro-second.

ROM\_FlashUsecSet  
// Sets the number of processor clocks per micro-second.

ROM\_FlashUserGet  
// Gets the User Registers

ROM\_FlashUserSave  
// Saves the User Registers

ROM\_FlashUserSet  
// Sets the User Registers

ROM\_GPIODirModeGet  
// Gets the direction and mode of a pin.

ROM\_GPIODirModeSet  
// Sets the direction and mode of the specified pin(s).

ROM\_GPIOIntTypeGet  
// Gets the interrupt type for a pin.

ROM\_GPIOIntTypeSet  
// Sets the interrupt type for the specified pin(s).



```
ROM_GPIOPadConfigGet
    // Gets the pad configuration for a pin.

ROM_GPIOPadConfigSet
    // Sets the pad configuration for the specified pin(s).

ROM_GPIOPinIntClear
    // Clears the interrupt for the specified pin(s).

ROM_GPIOPinIntDisable
    // Disables interrupts for the specified pin(s).

ROM_GPIOPinIntEnable
    // Enables interrupts for the specified pin(s).

ROM_GPIOPinIntStatus
    // Gets interrupt status for the specified GPIO port.

ROM_GPIOPinRead
    // Reads the values present of the specified pin(s).

ROM_GPIOPinTypeGPIOInput
    // Configures pin(s) for use as GPIO inputs.

ROM_GPIOPinTypeGPIOOutput
    // Configures pin(s) for use as GPIO outputs.

ROM_GPIOPinTypeGPIOOutputOD
    // Configures pin(s) for use as GPIO open drain outputs.

ROM_GPIOPinTypeI2C
    // Configures pin(s) for use by the I2C peripheral.

ROM_GPIOPinTypeSSI
    // Configures pin(s) for use by the SSI peripheral.

ROM_GPIOPinTypeTimer
    // Configures pin(s) for use by the Timer peripheral.

ROM_GPIOPinTypeUART
    // Configures pin(s) for use by the UART peripheral.

ROM_GPIOPinWrite
    // Writes a value to the specified pin(s).

ROM_I2CMasterBusBusy
    // Indicates whether or not the I2C bus is busy.

ROM_I2CMasterBusy
    // Indicates whether or not the I2C Master is busy.

ROM_I2CMasterControl
    // Controls the state of the I2C Master module.
```

ROM\_I2CMasterDataGet  
// Receives a byte that has been sent to the I2C Master.

ROM\_I2CMasterDataPut  
// Transmits a byte from the I2C Master.

ROM\_I2CMasterDisable  
// Disables the I2C master block.

ROM\_I2CMasterEnable  
// Enables the I2C Master block.

ROM\_I2CMasterErr  
// Gets the error status of the I2C Master module.

ROM\_I2CMasterInitExpClk  
// Initializes the I2C Master block.

ROM\_I2CMasterIntClear  
// Clears I2C Master interrupt sources.

ROM\_I2CMasterIntDisable  
// Disables the I2C Master interrupt.

ROM\_I2CMasterIntEnable  
// Enables the I2C Master interrupt.

ROM\_I2CMasterIntStatus  
// Gets the current I2C Master interrupt status.

ROM\_I2CMasterSlaveAddrSet  
// Sets the address that the I2C Master will place on the bus.

ROM\_I2CSlaveDataGet  
// Receives a byte that has been sent to the I2C Slave.

ROM\_I2CSlaveDataPut  
// Transmits a byte from the I2C Slave.

ROM\_I2CSlaveDisable  
// Disables the I2C slave block.

ROM\_I2CSlaveEnable  
// Enables the I2C Slave block.

ROM\_I2CSlaveInit  
// Initializes the I2C Slave block.

ROM\_I2CSlaveIntClear  
// Clears I2C Slave interrupt sources.

ROM\_I2CSlaveIntDisable  
// Disables the I2C Slave interrupt.

```
ROM_I2CSlaveIntEnable
    // Enables the I2C Slave interrupt.

ROM_I2CSlaveIntStatus
    // Gets the current I2C Slave interrupt status.

ROM_I2CSlaveStatus
    // Gets the I2C Slave module status.

ROM_IntDisable
    // Disables an interrupt.

ROM_IntEnable
    // Enables an interrupt.

ROM_IntMasterDisable
    // Disables the processor interrupt.

ROM_IntMasterEnable
    // Enables the processor interrupt.

ROM_IntPriorityGet
    // Gets the priority of an interrupt.

ROM_IntPriorityGroupingGet
    // Gets the priority grouping of the interrupt controller.

ROM_IntPriorityGroupingSet
    // Sets the priority grouping of the interrupt controller.

ROM_IntPrioritySet
    // Sets the priority of an interrupt.

ROM_SSIConfigSetExpClk
    // Configures the synchronous serial interface.

ROM_SSIDataGet
    // Gets a data element from the SSI receive FIFO.

ROM_SSIDataGetNonBlocking
    // Gets a data element from the SSI receive FIFO.

ROM_SSIDataPut
    // Puts a data element into the SSI transmit FIFO.

ROM_SSIDataPutNonBlocking
    // Puts a data element into the SSI transmit FIFO.

ROM_SSIDisable
    // Disables the synchronous serial interface.

ROM_SSIEnable
    // Enables the synchronous serial interface.
```

ROM\_SSIIntClear  
// Clears SSI interrupt sources.

ROM\_SSIIntDisable  
// Disables individual SSI interrupt sources.

ROM\_SSIIntEnable  
// Enables individual SSI interrupt sources.

ROM\_SSIIntStatus  
// Gets the current interrupt status.

ROM\_SysCtlADCSpeedGet  
// Gets the sample rate of the ADC.

ROM\_SysCtlADCSpeedSet  
// Sets the sample rate of the ADC.

ROM\_SysCtlClockGet  
// Gets the processor clock rate.

ROM\_SysCtlClockSet  
// Sets the clocking of the device.

ROM\_SysCtlDeepSleep  
// Puts the processor into deep-sleep mode.

ROM\_SysCtlFlashSizeGet  
// Gets the size of the flash.

ROM\_SysCtlGPIOAHBDisable  
// Disables a GPIO peripheral for access from the high speed bus.

ROM\_SysCtlGPIOAHBEnable  
// Enables a GPIO peripheral for access from the high speed bus.

ROM\_SysCtlIntClear  
// Clears system control interrupt sources.

ROM\_SysCtlIntDisable  
// Disables individual system control interrupt sources.

ROM\_SysCtlIntEnable  
// Enables individual system control interrupt sources.

ROM\_SysCtlIntStatus  
// Gets the current interrupt status.

ROM\_SysCtlLDOGet  
// Gets the output voltage of the LDO.

ROM\_SysCtlLDOSet  
// Sets the output voltage of the LDO.

ROM\_SysCtlPeripheralClockGating  
// Controls peripheral clock gating in sleep and deep-sleep mode.

ROM\_SysCtlPeripheralDeepSleepDisable  
// Disables a peripheral in deep-sleep mode.

ROM\_SysCtlPeripheralDeepSleepEnable  
// Enables a peripheral in deep-sleep mode.

ROM\_SysCtlPeripheralDisable  
// Disables a peripheral.

ROM\_SysCtlPeripheralEnable  
// Enables a peripheral.

ROM\_SysCtlPeripheralPresent  
// Determines if a peripheral is present.

ROM\_SysCtlPeripheralReset  
// Performs a software reset of a peripheral.

ROM\_SysCtlPeripheralSleepDisable  
// Disables a peripheral in sleep mode.

ROM\_SysCtlPeripheralSleepEnable  
// Enables a peripheral in sleep mode.

ROM\_SysCtlPinPresent  
// Determines if a pin is present.

ROM\_SysCtlReset  
// Resets the device.

ROM\_SysCtlResetCauseClear  
// Clears reset reasons.

ROM\_SysCtlResetCauseGet  
// Gets the reason for a reset.

ROM\_SysCtlSleep  
// Puts the processor into sleep mode.

ROM\_SysCtlSRAMSizeGet  
// Gets the size of the SRAM.

ROM\_SysTickDisable  
// Disables the SysTick counter.

ROM\_SysTickEnable  
// Enables the SysTick counter.

ROM\_SysTickIntDisable  
// Disables the SysTick interrupt.

ROM\_SysTickIntEnable  
// Enables the SysTick interrupt.

ROM\_SysTickPeriodGet  
// Gets the period of the SysTick counter.

ROM\_SysTickPeriodSet  
// Sets the period of the SysTick counter.

ROM\_SysTickValueGet  
// Gets the current value of the SysTick counter.

ROM\_TimerConfigure  
// Configures the timer(s).

ROM\_TimerControlEvent  
// Controls the event type.

ROM\_TimerControlLevel  
// Controls the output level.

ROM\_TimerControlStall  
// Controls the stall handling.

ROM\_TimerControlTrigger  
// Enables or disables the trigger output.

ROM\_TimerDisable  
// Disables the timer(s).

ROM\_TimerEnable  
// Enables the timer(s).

ROM\_TimerIntClear  
// Clears timer interrupt sources.

ROM\_TimerIntDisable  
// Disables individual timer interrupt sources.

ROM\_TimerIntEnable  
// Enables individual timer interrupt sources.

ROM\_TimerIntStatus  
// Gets the current interrupt status.

ROM\_TimerLoadGet  
// Gets the timer load value.

ROM\_TimerLoadSet  
// Sets the timer load value.

ROM\_TimerMatchGet  
// Gets the timer match value.

ROM\_TimerMatchSet  
// Sets the timer match value.

ROM\_TimerPrescaleGet  
// Get the timer prescale value.

ROM\_TimerPrescaleMatchGet  
// Get the timer prescale match value.

ROM\_TimerPrescaleMatchSet  
// Set the timer prescale match value.

ROM\_TimerPrescaleSet  
// Set the timer prescale value.

ROM\_TimerRTCDisable  
// Disable RTC counting.

ROM\_TimerRTCEnable  
// Enable RTC counting.

ROM\_TimerValueGet  
// Gets the current timer value.

ROM\_UARTBreakCtl  
// Causes a BREAK to be sent.

ROM\_UARTCharGet  
// Waits for a character from the specified port.

ROM\_UARTCharGetNonBlocking  
// Receives a character from the specified port.

ROM\_UARTCharPut  
// Waits to send a character from the specified port.

ROM\_UARTCharPutNonBlocking  
// Sends a character to the specified port.

ROM\_UARTCharsAvail  
// Determines if there are any characters in the receive FIFO.

ROM\_UARTConfigGetExpClk  
// Gets the current configuration of a UART.

ROM\_UARTConfigSetExpClk  
// Sets the configuration of a UART.

ROM\_UARTDisable  
// Disables transmitting and receiving.

ROM\_UARTDisableSIR  
// Disables SIR (IrDA) mode on the specified UART.

ROM\_UARTEnable  
// Enables transmitting and receiving.

ROM\_UARTEnableSIR  
// Enables SIR (IrDA) mode on specified UART.

ROM\_UARTFIFOLevelGet  
// Gets the FIFO level at which interrupts are generated.

ROM\_UARTFIFOLevelSet  
// Sets the FIFO level at which interrupts are generated.

ROM\_UARTIntClear  
// Clears UART interrupt sources.

ROM\_UARTIntDisable  
// Disables individual UART interrupt sources.

ROM\_UARTIntEnable  
// Enables individual UART interrupt sources.

ROM\_UARTIntStatus  
// Gets the current interrupt status.

ROM\_UARTParityModeGet  
// Gets the type of parity currently being used.

ROM\_UARTParityModeSet  
// Sets the type of parity.

ROM\_UARTSpaceAvail  
// Determines if there is any space in the transmit FIFO.

ROM\_UpdateI2C  
// Starts an update over the I2C0 interface.

ROM\_UpdateSSI  
// Starts an update over the SSI0 interface.

ROM\_UpdateUART  
// Starts an update over the UART0 interface.

ROM\_WatchdogEnable  
// Enables the watchdog timer.

ROM\_WatchdogIntClear  
// Clears the watchdog timer interrupt.

ROM\_WatchdogIntEnable  
// Enables the watchdog timer interrupt.

ROM\_WatchdogIntStatus  
// Gets the current watchdog timer interrupt status.



ROM\_WatchdogLock  
// Enables the watchdog timer lock mechanism.

ROM\_WatchdogLockState  
// Gets the state of the watchdog timer lock mechanism.

ROM\_WatchdogReloadGet  
// Gets the watchdog timer reload value.

ROM\_WatchdogReloadSet  
// Sets the watchdog timer reload value.

ROM\_WatchdogResetDisable  
// Disables the watchdog timer reset.

ROM\_WatchdogResetEnable  
// Enables the watchdog timer reset.

ROM\_WatchdogRunning  
// Determines if the watchdog timer is enabled.

ROM\_WatchdogStallDisable  
// Disables stalling of the watchdog timer during debug events.

ROM\_WatchdogStallEnable  
// Enables stalling of the watchdog timer during debug events.

ROM\_WatchdogUnlock  
// Disables the watchdog timer lock mechanism.

ROM\_WatchdogValueGet  
// Gets the current watchdog timer value.

## C Register Quick Reference

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16										
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0										
<b>System Control</b>																									
Base 0x400F.E000																									
<b>DID0, type RO, offset 0x000, reset -</b>																									
VER								CLASS																	
MAJOR								MINOR																	
<b>PBORCTL, type R/W, offset 0x030, reset 0x0000.7FFD</b>																									
														BORIOR											
<b>LDOPTL, type R/W, offset 0x034, reset 0x0000.0000</b>																									
														VADJ											
<b>RIS, type RO, offset 0x050, reset 0x0000.0000</b>																									
						MOSCPFRS		PLLRIS								BORRIS									
<b>IMC, type R/W, offset 0x054, reset 0x0000.0000</b>																									
						MOSCPFIM		PLLLIM								BORIM									
<b>MISC, type R/W1C, offset 0x058, reset 0x0000.0000</b>																									
						MOSCPFMS		PLLLMIS								BORMIS									
<b>RESC, type R/W, offset 0x05C, reset -</b>																									
										SW		WDT		BOR		POR		EXT		MOSCFAIL					
<b>RCC, type R/W, offset 0x060, reset 0x0780.3AD1</b>																									
PWRDN				ACG		SYSDIV				USESYS															
				BYPASS		XTAL				OSCSRC						IOSCDIS		MOSCDIS							
<b>PLLCFG, type RO, offset 0x064, reset -</b>																									
										F						R									
<b>GPIOHCTL, type R/W, offset 0x06C, reset 0x0000.0000</b>																									
										PORTEHS		PORTDHS		PORTCHS		PORTBHS		PORTAHS							
<b>RCC2, type R/W, offset 0x070, reset 0x0780.6810</b>																									
USERCC2				SYSDIV2																					
reserved-1		PWRDN2		BYPASS2						OSCSRC2															
<b>MOSCCTL, type R/W, offset 0x07C, reset 0x0000.0000</b>																									
														CVAL											
<b>DSLCLKCFG, type R/W, offset 0x144, reset 0x0780.0000</b>																									
DSDIVORIDE								DSOSCSRC																	
<b>DID1, type RO, offset 0x004, reset -</b>																									
VER				FAM				PARTNO																	
PINCOUNT								TEMP				PKG		ROHS		QUAL									
<b>DC0, type RO, offset 0x008, reset 0x007F.003F</b>																									
														SRAMSZ											
														FLASHSZ											
<b>DC1, type RO, offset 0x010, reset 0x0001.32FF</b>																									
MINSYS				MAXADCSPD				MPU		HIB		TEMPSNS		PLL		WDT		SWO		SWD		ADC		JTAG	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DC2, type RO, offset 0x014, reset 0x000F.5017</b>															
	I2C1		I2C0								SSIO	TIMER3	TIMER2	TIMER1	TIMER0
												UART2	UART1	UART0	
<b>DC3, type RO, offset 0x018, reset 0xBFFF.0000</b>															
32KHZ		CCP5	CCP4	CCP3	CCP2	CCP1	CCP0	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
<b>DC4, type RO, offset 0x01C, reset 0x0000.301F</b>															
		UDMA	ROM								GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
<b>DC5, type RO, offset 0x020, reset 0x0000.0000</b>															
<b>DC6, type RO, offset 0x024, reset 0x0000.0000</b>															
<b>DC7, type RO, offset 0x028, reset 0x00C0.0F00</b>															
								SSIO_TX	SSIO_RX	UART0_TX	UART0_RX				
<b>RCGC0, type R/W, offset 0x100, reset 0x00000040</b>															
								MAXADCSPD		HIB		WDT			ADC
<b>SCGC0, type R/W, offset 0x110, reset 0x00000040</b>															
								MAXADCSPD		HIB		WDT			ADC
<b>DCGC0, type R/W, offset 0x120, reset 0x00000040</b>															
								MAXADCSPD		HIB		WDT			ADC
<b>RCGC1, type R/W, offset 0x104, reset 0x00000000</b>															
	I2C1		I2C0								SSIO	TIMER3	TIMER2	TIMER1	TIMER0
												UART2	UART1	UART0	
<b>SCGC1, type R/W, offset 0x114, reset 0x00000000</b>															
	I2C1		I2C0								SSIO	TIMER3	TIMER2	TIMER1	TIMER0
												UART2	UART1	UART0	
<b>DCGC1, type R/W, offset 0x124, reset 0x00000000</b>															
	I2C1		I2C0								SSIO	TIMER3	TIMER2	TIMER1	TIMER0
												UART2	UART1	UART0	
<b>RCGC2, type R/W, offset 0x108, reset 0x00000000</b>															
		UDMA									GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
<b>SCGC2, type R/W, offset 0x118, reset 0x00000000</b>															
		UDMA									GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
<b>DCGC2, type R/W, offset 0x128, reset 0x00000000</b>															
		UDMA									GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
<b>SRRCR0, type R/W, offset 0x040, reset 0x00000000</b>															
									HIB			WDT			ADC
<b>SRRCR1, type R/W, offset 0x044, reset 0x00000000</b>															
	I2C1		I2C0								SSIO	TIMER3	TIMER2	TIMER1	TIMER0
												UART2	UART1	UART0	



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
<b>FMD, type R/W, offset 0x004, reset 0x0000.0000</b>																
DATA																
DATA																
<b>FMC, type R/W, offset 0x008, reset 0x0000.0000</b>																
WRKEY																
												COMT	MERASE	ERASE	WRITE	
<b>FCRIS, type RO, offset 0x00C, reset 0x0000.0000</b>																
														PRIS	ARIS	
<b>FCIM, type R/W, offset 0x010, reset 0x0000.0000</b>																
														PMASK	AMASK	
<b>FCMISC, type R/W1C, offset 0x014, reset 0x0000.0000</b>																
														PMISC	AMISC	
<b>Internal Memory</b>																
<b>Flash Registers (System Control Offset)</b>																
Base 0x400F.E000																
<b>USEACL, type R/W, offset 0x140, reset 0x31</b>																
USEC																
<b>RMVER, type RO, offset 0x0F4, reset 0x0000.0000</b>																
CONT								SIZE								
VER								REV								
<b>FMPRE0, type R/W, offset 0x130 and 0x200, reset 0xFFFF.FFFF</b>																
READ_ENABLE																
READ_ENABLE																
<b>FMPPE0, type R/W, offset 0x134 and 0x400, reset 0xFFFF.FFFF</b>																
PROG_ENABLE																
PROG_ENABLE																
<b>USER_DBG, type R/W, offset 0x1D0, reset 0xFFFF.FFFE</b>																
NW	DATA														DBG1	DBG0
<b>USER_REG0, type R/W, offset 0x1E0, reset 0xFFFF.FFFF</b>																
NW	DATA															
DATA																
<b>USER_REG1, type R/W, offset 0x1E4, reset 0xFFFF.FFFF</b>																
NW	DATA															
DATA																
<b>USER_REG2, type R/W, offset 0x1E8, reset 0xFFFF.FFFF</b>																
NW	DATA															
DATA																
<b>USER_REG3, type R/W, offset 0x1EC, reset 0xFFFF.FFFF</b>																
NW	DATA															
DATA																
<b>FMPRE1, type R/W, offset 0x204, reset 0xFFFF.FFFF</b>																
READ_ENABLE																
READ_ENABLE																
<b>FMPRE2, type R/W, offset 0x208, reset 0x0000.0000</b>																
READ_ENABLE																
READ_ENABLE																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>FMPRE3, type R/W, offset 0x20C, reset 0x0000.0000</b>															
READ_ENABLE															
READ_ENABLE															
<b>FMPPE1, type R/W, offset 0x404, reset 0xFFFF.FFFF</b>															
PROG_ENABLE															
PROG_ENABLE															
<b>FMPPE2, type R/W, offset 0x408, reset 0x0000.0000</b>															
PROG_ENABLE															
PROG_ENABLE															
<b>FMPPE3, type R/W, offset 0x40C, reset 0x0000.0000</b>															
PROG_ENABLE															
PROG_ENABLE															
<b>Micro Direct Memory Access (μDMA)</b>															
<b>μDMA Channel Control Structure</b>															
Base n/a															
<b>DMASRCENDP, type R/W, offset 0x000, reset -</b>															
ADDR															
ADDR															
<b>DMADSTENDP, type R/W, offset 0x004, reset -</b>															
ADDR															
ADDR															
<b>DMACHCTL, type R/W, offset 0x008, reset -</b>															
DSTINC		DSTSIZE		SRCINC		SRCSIZE								ARBSIZE	
ARBSIZE						XFERSIZE				NKUSEBURST				XFERMODE	
<b>Micro Direct Memory Access (μDMA)</b>															
<b>μDMA Registers</b>															
Base 0x400F.F000															
<b>DMASTAT, type RO, offset 0x000, reset 0x001F.0000</b>															
DMACHANS															
STATE															
MASTEN															
<b>DMACFG, type WO, offset 0x004, reset -</b>															
MASTEN															
<b>DMACTLBASE, type R/W, offset 0x008, reset 0x0000.0000</b>															
ADDR															
ADDR															
<b>DMAALTBASE, type RO, offset 0x00C, reset 0x0000.0200</b>															
ADDR															
ADDR															
<b>DMAWAITSTAT, type RO, offset 0x010, reset 0x0000.0000</b>															
WAITREQ[n]															
WAITREQ[n]															
<b>DMASWREQ, type WO, offset 0x014, reset -</b>															
SWREQ[n]															
SWREQ[n]															
<b>DMAUSEBURSTSET, type RO, offset 0x018, reset 0x0000.0000</b>															
SET[n]															
SET[n]															
<b>DMAUSEBURSTSET, type WO, offset 0x018, reset 0x0000.0000</b>															
SET[n]															
SET[n]															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DMAUSEBURSTCLR, type WO, offset 0x01C, reset -</b>															
CLR[n]															
CLR[n]															
<b>DMAREQMASKSET, type RO, offset 0x020, reset 0x0000.0000</b>															
SET[n]															
SET[n]															
<b>DMAREQMASKSET, type WO, offset 0x020, reset 0x0000.0000</b>															
SET[n]															
SET[n]															
<b>DMAREQMASKCLR, type WO, offset 0x024, reset -</b>															
CLR[n]															
CLR[n]															
<b>DMAENASET, type RO, offset 0x028, reset 0x0000.0000</b>															
SET[n]															
SET[n]															
<b>DMAENASET, type WO, offset 0x028, reset 0x0000.0000</b>															
CHENSET[n]															
CHENSET[n]															
<b>DMAENACL, type WO, offset 0x02C, reset -</b>															
CLR[n]															
CLR[n]															
<b>DMAALTSET, type RO, offset 0x030, reset 0x0000.0000</b>															
SET[n]															
SET[n]															
<b>DMAALTSET, type WO, offset 0x030, reset 0x0000.0000</b>															
SET[n]															
SET[n]															
<b>DMAALTCLR, type WO, offset 0x034, reset -</b>															
CLR[n]															
CLR[n]															
<b>DMAPRIOSET, type RO, offset 0x038, reset 0x0000.0000</b>															
SET[n]															
SET[n]															
<b>DMAPRIOSET, type WO, offset 0x038, reset 0x0000.0000</b>															
SET[n]															
SET[n]															
<b>DMAPRIOCLR, type WO, offset 0x03C, reset -</b>															
CLR[n]															
CLR[n]															
<b>DMAERRCLR, type RO, offset 0x04C, reset 0x0000.0000</b>															
															ERRCLR
<b>DMAERRCLR, type WO, offset 0x04C, reset 0x0000.0000</b>															
															ERRCLR
<b>DMAPeriphID0, type RO, offset 0xFE0, reset 0x0000.0030</b>															
															PID0
<b>DMAPeriphID1, type RO, offset 0xFE4, reset 0x0000.00B2</b>															
															PID1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DMAPeriphID2, type RO, offset 0xFE8, reset 0x0000.000B</b>															
												PID2			
<b>DMAPeriphID3, type RO, offset 0xFEC, reset 0x0000.0000</b>															
												PID3			
<b>DMAPeriphID4, type RO, offset 0xFD0, reset 0x0000.0004</b>															
												PID4			
<b>DMACellID0, type RO, offset 0xFF0, reset 0x0000.000D</b>															
												CID0			
<b>DMACellID1, type RO, offset 0xFF4, reset 0x0000.00F0</b>															
												CID1			
<b>DMACellID2, type RO, offset 0xFF8, reset 0x0000.0005</b>															
												CID2			
<b>DMACellID3, type RO, offset 0xFFC, reset 0x0000.00B1</b>															
												CID3			
<b>General-Purpose Input/Outputs (GPIOs)</b>															
GPIO Port A (legacy) base: 0x4000.4000															
GPIO Port A (high-speed) base: 0x4005.8000															
GPIO Port B (legacy) base: 0x4000.5000															
GPIO Port B (high-speed) base: 0x4005.9000															
GPIO Port C (legacy) base: 0x4000.6000															
GPIO Port C (high-speed) base: 0x4005.A000															
GPIO Port D (legacy) base: 0x4000.7000															
GPIO Port D (high-speed) base: 0x4005.B000															
GPIO Port E (legacy) base: 0x4002.4000															
GPIO Port E (high-speed) base: 0x4005.C000															
<b>GPIO_DATA, type R/W, offset 0x000, reset 0x0000.0000</b>															
												DATA			
<b>GPIO_DIR, type R/W, offset 0x400, reset 0x0000.0000</b>															
												DIR			
<b>GPIO_IS, type R/W, offset 0x404, reset 0x0000.0000</b>															
												IS			
<b>GPIO_IBE, type R/W, offset 0x408, reset 0x0000.0000</b>															
												IBE			
<b>GPIO_IEV, type R/W, offset 0x40C, reset 0x0000.0000</b>															
												IEV			
<b>GPIO_IM, type R/W, offset 0x410, reset 0x0000.0000</b>															
												IME			
<b>GPIO_RIS, type RO, offset 0x414, reset 0x0000.0000</b>															
												RIS			
<b>GPIO_MIS, type RO, offset 0x418, reset 0x0000.0000</b>															
												MIS			



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOICR, type W1C, offset 0x41C, reset 0x0000.0000															
												IC			
GPIOAFSEL, type R/W, offset 0x420, reset -															
												AFSEL			
GPIODR2R, type R/W, offset 0x500, reset 0x0000.00FF															
												DRV2			
GPIODR4R, type R/W, offset 0x504, reset 0x0000.0000															
												DRV4			
GPIODR8R, type R/W, offset 0x508, reset 0x0000.0000															
												DRV8			
GPIODR, type R/W, offset 0x50C, reset 0x0000.0000															
												ODE			
GPIOPUR, type R/W, offset 0x510, reset -															
												PUE			
GPIOPDR, type R/W, offset 0x514, reset 0x0000.0000															
												PDE			
GPIOCLR, type R/W, offset 0x518, reset 0x0000.0000															
												SRL			
GPIODEN, type R/W, offset 0x51C, reset -															
												DEN			
GPIOLOCK, type R/W, offset 0x520, reset 0x0000.0001															
												LOCK			
												LOCK			
GPIOCR, type -, offset 0x524, reset -															
												CR			
GPIOAMSEL, type R/W, offset 0x528, reset 0x0000.0000															
												GPIOAMSEL			
GPIOPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000															
												PID4			
GPIOPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000															
												PID5			
GPIOPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000															
												PID6			
GPIOPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000															
												PID7			

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
<b>GPIOPeriphID0, type RO, offset 0xFE0, reset 0x0000.0061</b>																
												PID0				
<b>GPIOPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000</b>																
												PID1				
<b>GPIOPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018</b>																
												PID2				
<b>GPIOPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001</b>																
												PID3				
<b>GPIOPCellID0, type RO, offset 0xFF0, reset 0x0000.000D</b>																
												CID0				
<b>GPIOPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0</b>																
												CID1				
<b>GPIOPCellID2, type RO, offset 0xFF8, reset 0x0000.0005</b>																
												CID2				
<b>GPIOPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1</b>																
												CID3				
<b>General-Purpose Timers</b>																
Timer0 base: 0x4003.0000																
Timer1 base: 0x4003.1000																
Timer2 base: 0x4003.2000																
Timer3 base: 0x4003.3000																
<b>GPTMCFG, type R/W, offset 0x000, reset 0x0000.0000</b>																
												GPTMCFG				
<b>GPTMTAMR, type R/W, offset 0x004, reset 0x0000.0000</b>																
												TAAMS	TACMR	TAMR		
<b>GPTMTBMR, type R/W, offset 0x008, reset 0x0000.0000</b>																
												TBAMS	TBCMR	TBMR		
<b>GPTMCTL, type R/W, offset 0x00C, reset 0x0000.0000</b>																
		TBPWML	TBOTE	TBEVENT		TBSTALL	TBEN			TAPWML	TAOTE	RTCEN	TAEVENT		TASTALL	TAEN
<b>GPTMIMR, type R/W, offset 0x018, reset 0x0000.0000</b>																
				CBEIM	CBMIM	TBTOIM						RTCIM	CAEIM	CAMIM	TATOIM	
<b>GPTMRIS, type RO, offset 0x01C, reset 0x0000.0000</b>																
				CBERIS	CBMRIS	TBTORIS						RTCIS	CAERIS	CAMRIS	TATORIS	
<b>GPTMMIS, type RO, offset 0x020, reset 0x0000.0000</b>																
				CBEMIS	CBMMIS	TBTOMIS						RTCMIS	CAEMIS	CAMMIS	TATOMIS	
<b>GPTMICR, type W1C, offset 0x024, reset 0x0000.0000</b>																
				CBECINT	CBMCINT	TBTCINT						RTCCINT	CAECINT	CAMCINT	TATOCINT	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPTMTAILR, type R/W, offset 0x028, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)															
TAILRH															
TAILRL															
GPTMTBILR, type R/W, offset 0x02C, reset 0x0000.FFFF															
TBILRL															
GPTMTAMATCHR, type R/W, offset 0x030, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)															
TAMRH															
TAMRL															
GPTMTBMATCHR, type R/W, offset 0x034, reset 0x0000.FFFF															
TBMRL															
GPTMTAPR, type R/W, offset 0x038, reset 0x0000.0000															
TAPSR															
GPTMTBPR, type R/W, offset 0x03C, reset 0x0000.0000															
TBPSR															
GPTMTAR, type RO, offset 0x048, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)															
TARH															
TARL															
GPTMTBR, type RO, offset 0x04C, reset 0x0000.FFFF															
TBRL															
<b>Watchdog Timer</b>															
Base 0x4000.0000															
WDTLOAD, type R/W, offset 0x000, reset 0xFFFF.FFFF															
WDTLoad															
WDTLoad															
WDTVALUE, type RO, offset 0x004, reset 0xFFFF.FFFF															
WDTValue															
WDTValue															
WDTCTL, type R/W, offset 0x008, reset 0x0000.0000															
														RESEN	INTEN
WDTICR, type WO, offset 0x00C, reset -															
WDTIntClr															
WDTIntClr															
WDTRIS, type RO, offset 0x010, reset 0x0000.0000															
WDTRIS															
WDTMIS, type RO, offset 0x014, reset 0x0000.0000															
WDTMIS															
WDTTEST, type R/W, offset 0x418, reset 0x0000.0000															
STALL															
WDTLOCK, type R/W, offset 0xC00, reset 0x0000.0000															
WDTLock															
WDTLock															
WDTPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000															
PID4															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>WDTPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000</b>															
												PID5			
<b>WDTPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000</b>															
												PID6			
<b>WDTPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000</b>															
												PID7			
<b>WDTPeriphID0, type RO, offset 0xFE0, reset 0x0000.0005</b>															
												PID0			
<b>WDTPeriphID1, type RO, offset 0xFE4, reset 0x0000.0018</b>															
												PID1			
<b>WDTPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018</b>															
												PID2			
<b>WDTPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001</b>															
												PID3			
<b>WDTPCellID0, type RO, offset 0xFF0, reset 0x0000.000D</b>															
												CID0			
<b>WDTPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0</b>															
												CID1			
<b>WDTPCellID2, type RO, offset 0xFF8, reset 0x0000.0005</b>															
												CID2			
<b>WDTPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1</b>															
												CID3			
<b>Analog-to-Digital Converter (ADC)</b> Base 0x4003.8000															
<b>ADCACTSS, type R/W, offset 0x000, reset 0x0000.0000</b>															
												ASEN3	ASEN2	ASEN1	ASEN0
<b>ADCRIS, type RO, offset 0x004, reset 0x0000.0000</b>															
												INR3	INR2	INR1	INR0
<b>ADCIM, type R/W, offset 0x008, reset 0x0000.0000</b>															
												MASK3	MASK2	MASK1	MASK0
<b>ADCISC, type R/W1C, offset 0x00C, reset 0x0000.0000</b>															
												IN3	IN2	IN1	IN0
<b>ADCOSTAT, type R/W1C, offset 0x010, reset 0x0000.0000</b>															
												OV3	OV2	OV1	OV0
<b>ADCEMUX, type R/W, offset 0x014, reset 0x0000.0000</b>															
EM3				EM2				EM1				EM0			

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>ADCUSTAT, type R/W1C, offset 0x018, reset 0x0000.0000</b>															
												UV3	UV2	UV1	UV0
<b>ADCSSPRI, type R/W, offset 0x020, reset 0x0000.3210</b>															
SS3				SS2				SS1				SS0			
<b>ADCPSSI, type WO, offset 0x028, reset -</b>															
												SS3	SS2	SS1	SS0
<b>ADCSSAC, type R/W, offset 0x030, reset 0x0000.0000</b>															
												AVG			
<b>ADCSSMUX0, type R/W, offset 0x040, reset 0x0000.0000</b>															
MUX7				MUX6				MUX5				MUX4			
MUX3				MUX2				MUX1				MUX0			
<b>ADCSSCTL0, type R/W, offset 0x044, reset 0x0000.0000</b>															
TS7	IE7	END7	D7	TS6	IE6	END6	D6	TS5	IE5	END5	D5	TS4	IE4	END4	D4
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
<b>ADCSSFIFO0, type RO, offset 0x048, reset 0x0000.0000</b>															
												DATA			
<b>ADCSSFIFO1, type RO, offset 0x068, reset 0x0000.0000</b>															
												DATA			
<b>ADCSSFIFO2, type RO, offset 0x088, reset 0x0000.0000</b>															
												DATA			
<b>ADCSSFIFO3, type RO, offset 0x0A8, reset 0x0000.0000</b>															
												DATA			
<b>ADCSSFSTAT0, type RO, offset 0x04C, reset 0x0000.0100</b>															
FULL				EMPTY				HPTR				TPTR			
<b>ADCSSFSTAT1, type RO, offset 0x06C, reset 0x0000.0100</b>															
FULL				EMPTY				HPTR				TPTR			
<b>ADCSSFSTAT2, type RO, offset 0x08C, reset 0x0000.0100</b>															
FULL				EMPTY				HPTR				TPTR			
<b>ADCSSFSTAT3, type RO, offset 0x0AC, reset 0x0000.0100</b>															
FULL				EMPTY				HPTR				TPTR			
<b>ADCSSMUX1, type R/W, offset 0x060, reset 0x0000.0000</b>															
MUX3				MUX2				MUX1				MUX0			
<b>ADCSSMUX2, type R/W, offset 0x080, reset 0x0000.0000</b>															
MUX3				MUX2				MUX1				MUX0			
<b>ADCSSCTL1, type R/W, offset 0x064, reset 0x0000.0000</b>															
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
<b>UARTICR, type W1C, offset 0x044, reset 0x0000.0000</b>																
				OEIC		BEIC		PEIC		FEIC		RTIC		TXIC RXIC		
<b>UARTDMACTL, type R/W, offset 0x048, reset 0x0000.0000</b>																
												DMAERR		TXDMAE	RXDMAE	
<b>UARTPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000</b>																
												PID4				
<b>UARTPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000</b>																
												PID5				
<b>UARTPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000</b>																
												PID6				
<b>UARTPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000</b>																
												PID7				
<b>UARTPeriphID0, type RO, offset 0xFE0, reset 0x0000.0011</b>																
												PID0				
<b>UARTPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000</b>																
												PID1				
<b>UARTPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018</b>																
												PID2				
<b>UARTPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001</b>																
												PID3				
<b>UARTPCellID0, type RO, offset 0xFF0, reset 0x0000.000D</b>																
												CID0				
<b>UARTPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0</b>																
												CID1				
<b>UARTPCellID2, type RO, offset 0xFF8, reset 0x0000.0005</b>																
												CID2				
<b>UARTPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1</b>																
												CID3				
<b>Synchronous Serial Interface (SSI)</b>																
SSI0 base: 0x4000.8000																
<b>SSICR0, type R/W, offset 0x000, reset 0x0000.0000</b>																
SCR				SPH		SPO		FRF		DSS						
<b>SSICR1, type R/W, offset 0x004, reset 0x0000.0000</b>																
												SOD	MS	SSE		LBM
<b>SSIDR, type R/W, offset 0x008, reset 0x0000.0000</b>																
DATA																

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
<b>SSISR, type RO, offset 0x00C, reset 0x0000.0003</b>																
												BSY	RFF	RNE	TNF	TFE
<b>SSICPSR, type R/W, offset 0x010, reset 0x0000.0000</b>																
CPSDVSR																
<b>SSIIM, type R/W, offset 0x014, reset 0x0000.0000</b>																
												TXIM	RXIM	RTIM	RORIM	
<b>SSIRIS, type RO, offset 0x018, reset 0x0000.0008</b>																
												TXRIS	RXRIS	RTRIS	RORRIS	
<b>SSIMIS, type RO, offset 0x01C, reset 0x0000.0000</b>																
												TXMIS	RXMIS	RTMIS	RORMIS	
<b>SSIICR, type W1C, offset 0x020, reset 0x0000.0000</b>																
														RTIC	RORIC	
<b>SSIDMACTL, type R/W, offset 0x024, reset 0x0000.0000</b>																
														TXDMAE	RXDMAE	
<b>SSIPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000</b>																
PID4																
<b>SSIPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000</b>																
PID5																
<b>SSIPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000</b>																
PID6																
<b>SSIPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000</b>																
PID7																
<b>SSIPeriphID0, type RO, offset 0xFE0, reset 0x0000.0022</b>																
PID0																
<b>SSIPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000</b>																
PID1																
<b>SSIPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018</b>																
PID2																
<b>SSIPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001</b>																
PID3																
<b>SSIPCellID0, type RO, offset 0xFF0, reset 0x0000.000D</b>																
CID0																
<b>SSIPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0</b>																
CID1																



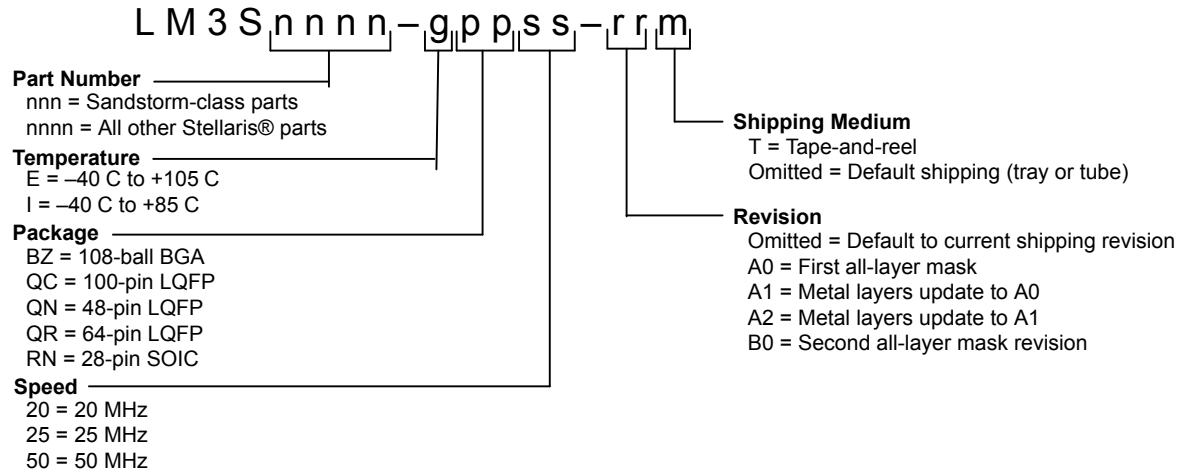
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
<b>SSIPCellID2, type RO, offset 0xFF8, reset 0x0000.0005</b>																		
												CID2						
<b>SSIPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1</b>																		
												CID3						
<b>Inter-Integrated Circuit (I<sup>2</sup>C) Interface</b>																		
<b>I<sup>2</sup>C Master</b>																		
I2C Master 0 base: 0x4002.0000																		
I2C Master 1 base: 0x4002.1000																		
<b>I2CMSA, type R/W, offset 0x000, reset 0x0000.0000</b>																		
												SA		R/S				
<b>I2CMCS, type RO, offset 0x004, reset 0x0000.0000</b>																		
												BUSBSY	IDLE	ARBLST	DATAACK	ADRACK	ERROR	BUSY
<b>I2CMCS, type WO, offset 0x004, reset 0x0000.0000</b>																		
												ACK	STOP	START	RUN			
<b>I2CMDR, type R/W, offset 0x008, reset 0x0000.0000</b>																		
												DATA						
<b>I2CMTPR, type R/W, offset 0x00C, reset 0x0000.0001</b>																		
												TPR						
<b>I2CMIMR, type R/W, offset 0x010, reset 0x0000.0000</b>																		
												IM						
<b>I2CMRIS, type RO, offset 0x014, reset 0x0000.0000</b>																		
												RIS						
<b>I2CMMIS, type RO, offset 0x018, reset 0x0000.0000</b>																		
												MIS						
<b>I2CMICR, type WO, offset 0x01C, reset 0x0000.0000</b>																		
												IC						
<b>I2CMCR, type R/W, offset 0x020, reset 0x0000.0000</b>																		
												SFE	MFE			LPBK		
<b>Inter-Integrated Circuit (I<sup>2</sup>C) Interface</b>																		
<b>I<sup>2</sup>C Slave</b>																		
I2C Slave 0 base: 0x4002.0800																		
I2C Slave 1 base: 0x4002.1800																		
<b>I2CSOAR, type R/W, offset 0x000, reset 0x0000.0000</b>																		
												OAR						
<b>I2CCSR, type RO, offset 0x004, reset 0x0000.0000</b>																		
												FBR	TREQ	RREQ				
<b>I2CCSR, type WO, offset 0x004, reset 0x0000.0000</b>																		
												DA						

**Register Quick Reference**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>I2CSDR, type R/W, offset 0x008, reset 0x0000.0000</b>															
												DATA			
<b>I2CSIMR, type R/W, offset 0x00C, reset 0x0000.0000</b>															
												STOPIM	STARTIM	DATAIM	
<b>I2CSRIS, type RO, offset 0x010, reset 0x0000.0000</b>															
												STOPRIS	STARTRIS	DATARIS	
<b>I2CSMIS, type RO, offset 0x014, reset 0x0000.0000</b>															
												STOPMIS	STARTMIS	DATAMIS	
<b>I2CSICR, type WO, offset 0x018, reset 0x0000.0000</b>															
												STOPIC	STARTIC	DATAIC	

## D Ordering and Contact Information

### D.1 Ordering Information



**Table D-1. Part Ordering Information**

Orderable Part Number	Description
LM3S1607-IQR50	Stellaris® LM3S1607 Microcontroller
LM3S1607-IQR50(T)	Stellaris® LM3S1607 Microcontroller

### D.2 Kits

The Luminary Micro Stellaris® Family provides the hardware and software tools that engineers need to begin development quickly.

- Reference Design Kits accelerate product development by providing ready-to-run hardware, and comprehensive documentation including hardware design files:  
[http://www.luminarymicro.com/products/reference\\_design\\_kits/](http://www.luminarymicro.com/products/reference_design_kits/)
- Evaluation Kits provide a low-cost and effective means of evaluating Stellaris® microcontrollers before purchase:  
<http://www.luminarymicro.com/products/kits.html>
- Development Kits provide you with all the tools you need to develop and prototype embedded applications right out of the box:  
[http://www.luminarymicro.com/products/development\\_kits.html](http://www.luminarymicro.com/products/development_kits.html)

See the Luminary Micro website for the latest tools available, or ask your Luminary Micro distributor.

### D.3 Company Information

Luminary Micro, Inc. designs, markets, and sells ARM Cortex-M3-based microcontrollers (MCUs). Austin, Texas-based Luminary Micro is the lead partner for the Cortex-M3 processor, delivering the world's first silicon implementation of the Cortex-M3 processor. Luminary Micro's introduction of the

Stellaris® family of products provides 32-bit performance for the same price as current 8- and 16-bit microcontroller designs. With entry-level pricing at \$1.00 for an ARM technology-based MCU, Luminary Micro's Stellaris product line allows for standardization that eliminates future architectural upgrades or software tool changes.

Luminary Micro, Inc.  
108 Wild Basin, Suite 350  
Austin, TX 78746  
Main: +1-512-279-8800  
Fax: +1-512-279-8879  
<http://www.luminarymicro.com>  
[sales@luminarymicro.com](mailto:sales@luminarymicro.com)

## **D.4 Support Information**

For support on Luminary Micro products, contact:  
[support@luminarymicro.com](mailto:support@luminarymicro.com) +1-512-279-8800, ext. 3