



# Stellaris<sup>®</sup> LM3S811 Microcontroller

DATA SHEET

---

# Copyright

Copyright © 2007-2014 Texas Instruments Incorporated All rights reserved. Stellaris and StellarisWarc® are registered trademarks of Texas Instruments Incorporated. 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.

PRODUCTION DATA information is current as of publication date. Products conform to specifications per the terms of Texas Instruments standard warranty. Production processing does not necessarily include testing of all parameters.

**⚠** Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

Texas Instruments Incorporated

108 Wild Basin, Suite 350

Austin, TX 78746

<http://www.ti.com/stellaris>

<http://www-k.ext.ti.com/sc/technical-support/product-information-centers.htm>



# Table of Contents

<b>Revision History .....</b>	<b>21</b>
<b>About This Document .....</b>	<b>25</b>
Audience .....	25
About This Manual .....	25
Related Documents .....	25
Documentation Conventions .....	26
<b>1      Architectural Overview .....</b>	<b>28</b>
1.1    Product Features .....	28
1.2    Target Applications .....	35
1.3    High-Level Block Diagram .....	35
1.4    Functional Overview .....	37
1.4.1    ARM Cortex™-M3 .....	37
1.4.2    Motor Control Peripherals .....	38
1.4.3    Analog Peripherals .....	38
1.4.4    Serial Communications Peripherals .....	39
1.4.5    System Peripherals .....	40
1.4.6    Memory Peripherals .....	41
1.4.7    Additional Features .....	41
1.4.8    Hardware Details .....	42
1.4.9    System Block Diagram .....	43
<b>2      The Cortex-M3 Processor .....</b>	<b>44</b>
2.1    Block Diagram .....	45
2.2    Overview .....	46
2.2.1    System-Level Interface .....	46
2.2.2    Integrated Configurable Debug .....	46
2.2.3    Trace Port Interface Unit (TPIU) .....	47
2.2.4    Cortex-M3 System Component Details .....	47
2.3    Programming Model .....	48
2.3.1    Processor Mode and Privilege Levels for Software Execution .....	48
2.3.2    Stacks .....	48
2.3.3    Register Map .....	49
2.3.4    Register Descriptions .....	50
2.3.5    Exceptions and Interrupts .....	63
2.3.6    Data Types .....	63
2.4    Memory Model .....	63
2.4.1    Memory Regions, Types and Attributes .....	64
2.4.2    Memory System Ordering of Memory Accesses .....	65
2.4.3    Behavior of Memory Accesses .....	65
2.4.4    Software Ordering of Memory Accesses .....	66
2.4.5    Bit-Banding .....	67
2.4.6    Data Storage .....	69
2.4.7    Synchronization Primitives .....	70
2.5    Exception Model .....	71
2.5.1    Exception States .....	72
2.5.2    Exception Types .....	72

2.5.3	Exception Handlers .....	75
2.5.4	Vector Table .....	75
2.5.5	Exception Priorities .....	76
2.5.6	Interrupt Priority Grouping .....	76
2.5.7	Exception Entry and Return .....	76
2.6	Fault Handling .....	78
2.6.1	Fault Types .....	79
2.6.2	Fault Escalation and Hard Faults .....	79
2.6.3	Fault Status Registers and Fault Address Registers .....	80
2.6.4	Lockup .....	80
2.7	Power Management .....	81
2.7.1	Entering Sleep Modes .....	81
2.7.2	Wake Up from Sleep Mode .....	81
2.8	Instruction Set Summary .....	82
<b>3</b>	<b>Cortex-M3 Peripherals .....</b>	<b>86</b>
3.1	Functional Description .....	86
3.1.1	System Timer (SysTick) .....	86
3.1.2	Nested Vectored Interrupt Controller (NVIC) .....	87
3.1.3	System Control Block (SCB) .....	89
3.1.4	Memory Protection Unit (MPU) .....	89
3.2	Register Map .....	94
3.3	System Timer (SysTick) Register Descriptions .....	95
3.4	NVIC Register Descriptions .....	99
3.5	System Control Block (SCB) Register Descriptions .....	107
3.6	Memory Protection Unit (MPU) Register Descriptions .....	134
<b>4</b>	<b>JTAG Interface .....</b>	<b>144</b>
4.1	Block Diagram .....	145
4.2	Signal Description .....	145
4.3	Functional Description .....	146
4.3.1	JTAG Interface Pins .....	146
4.3.2	JTAG TAP Controller .....	148
4.3.3	Shift Registers .....	149
4.3.4	Operational Considerations .....	149
4.4	Initialization and Configuration .....	151
4.5	Register Descriptions .....	151
4.5.1	Instruction Register (IR) .....	151
4.5.2	Data Registers .....	153
<b>5</b>	<b>System Control .....</b>	<b>155</b>
5.1	Signal Description .....	155
5.2	Functional Description .....	155
5.2.1	Device Identification .....	155
5.2.2	Reset Control .....	156
5.2.3	Power Control .....	160
5.2.4	Clock Control .....	160
5.2.5	System Control .....	164
5.3	Initialization and Configuration .....	165
5.4	Register Map .....	165
5.5	Register Descriptions .....	166

<b>6</b>	<b>Internal Memory .....</b>	<b>214</b>
6.1	Block Diagram .....	214
6.2	Functional Description .....	214
6.2.1	SRAM Memory .....	214
6.2.2	Flash Memory .....	215
6.3	Flash Memory Initialization and Configuration .....	218
6.3.1	Changing Flash Protection Bits .....	218
6.3.2	Flash Programming .....	219
6.4	Register Map .....	220
6.5	Flash Register Descriptions (Flash Control Offset) .....	220
6.6	Flash Register Descriptions (System Control Offset) .....	228
<b>7</b>	<b>General-Purpose Input/Outputs (GPIOs) .....</b>	<b>232</b>
7.1	Block Diagram .....	233
7.2	Signal Description .....	233
7.3	Functional Description .....	237
7.3.1	Data Control .....	238
7.3.2	Interrupt Control .....	239
7.3.3	Mode Control .....	240
7.3.4	Pad Control .....	240
7.3.5	Identification .....	240
7.4	Initialization and Configuration .....	240
7.5	Register Map .....	241
7.6	Register Descriptions .....	243
<b>8</b>	<b>General-Purpose Timers .....</b>	<b>275</b>
8.1	Block Diagram .....	276
8.2	Signal Description .....	276
8.3	Functional Description .....	277
8.3.1	GPTM Reset Conditions .....	277
8.3.2	32-Bit Timer Operating Modes .....	278
8.3.3	16-Bit Timer Operating Modes .....	279
8.4	Initialization and Configuration .....	283
8.4.1	32-Bit One-Shot/Periodic Timer Mode .....	283
8.4.2	32-Bit Real-Time Clock (RTC) Mode .....	284
8.4.3	16-Bit One-Shot/Periodic Timer Mode .....	284
8.4.4	16-Bit Input Edge Count Mode .....	285
8.4.5	16-Bit Input Edge Timing Mode .....	285
8.4.6	16-Bit PWM Mode .....	286
8.5	Register Map .....	286
8.6	Register Descriptions .....	287
<b>9</b>	<b>Watchdog Timer .....</b>	<b>312</b>
9.1	Block Diagram .....	313
9.2	Functional Description .....	313
9.3	Initialization and Configuration .....	314
9.4	Register Map .....	314
9.5	Register Descriptions .....	315
<b>10</b>	<b>Analog-to-Digital Converter (ADC) .....</b>	<b>336</b>
10.1	Block Diagram .....	336

10.2	Signal Description .....	337
10.3	Functional Description .....	338
10.3.1	Sample Sequencers .....	338
10.3.2	Module Control .....	338
10.3.3	Hardware Sample Averaging Circuit .....	339
10.3.4	Analog-to-Digital Converter .....	339
10.3.5	Differential Sampling .....	340
10.3.6	Test Modes .....	342
10.3.7	Internal Temperature Sensor .....	342
10.4	Initialization and Configuration .....	343
10.4.1	Module Initialization .....	343
10.4.2	Sample Sequencer Configuration .....	343
10.5	Register Map .....	344
10.6	Register Descriptions .....	345
<b>11</b>	<b>Universal Asynchronous Receivers/Transmitters (UARTs) .....</b>	<b>373</b>
11.1	Block Diagram .....	374
11.2	Signal Description .....	374
11.3	Functional Description .....	375
11.3.1	Transmit/Receive Logic .....	375
11.3.2	Baud-Rate Generation .....	375
11.3.3	Data Transmission .....	376
11.3.4	FIFO Operation .....	376
11.3.5	Interrupts .....	377
11.3.6	Loopback Operation .....	378
11.4	Initialization and Configuration .....	378
11.5	Register Map .....	379
11.6	Register Descriptions .....	380
<b>12</b>	<b>Synchronous Serial Interface (SSI) .....</b>	<b>413</b>
12.1	Block Diagram .....	413
12.2	Signal Description .....	413
12.3	Functional Description .....	414
12.3.1	Bit Rate Generation .....	414
12.3.2	FIFO Operation .....	415
12.3.3	Interrupts .....	415
12.3.4	Frame Formats .....	415
12.4	Initialization and Configuration .....	423
12.5	Register Map .....	424
12.6	Register Descriptions .....	425
<b>13</b>	<b>Inter-Integrated Circuit (I<sup>2</sup>C) Interface .....</b>	<b>451</b>
13.1	Block Diagram .....	452
13.2	Signal Description .....	452
13.3	Functional Description .....	452
13.3.1	I <sup>2</sup> C Bus Functional Overview .....	453
13.3.2	Available Speed Modes .....	455
13.3.3	Interrupts .....	456
13.3.4	Loopback Operation .....	456
13.3.5	Command Sequence Flow Charts .....	456
13.4	Initialization and Configuration .....	464

13.5	Register Map .....	465
13.6	Register Descriptions (I <sup>2</sup> C Master) .....	466
13.7	Register Descriptions (I <sup>2</sup> C Slave) .....	479
<b>14</b>	<b>Analog Comparator .....</b>	<b>488</b>
14.1	Block Diagram .....	488
14.2	Signal Description .....	488
14.3	Functional Description .....	489
14.3.1	Internal Reference Programming .....	490
14.4	Initialization and Configuration .....	491
14.5	Register Map .....	491
14.6	Register Descriptions .....	492
<b>15</b>	<b>Pulse Width Modulator (PWM) .....</b>	<b>500</b>
15.1	Block Diagram .....	501
15.2	Signal Description .....	502
15.3	Functional Description .....	503
15.3.1	PWM Timer .....	503
15.3.2	PWM Comparators .....	503
15.3.3	PWM Signal Generator .....	504
15.3.4	Dead-Band Generator .....	505
15.3.5	Interrupt/ADC-Trigger Selector .....	505
15.3.6	Synchronization Methods .....	506
15.3.7	Fault Conditions .....	506
15.3.8	Output Control Block .....	506
15.4	Initialization and Configuration .....	506
15.5	Register Map .....	507
15.6	Register Descriptions .....	509
<b>16</b>	<b>Pin Diagram .....</b>	<b>539</b>
<b>17</b>	<b>Signal Tables .....</b>	<b>541</b>
17.1	Signals by Pin Number .....	541
17.2	Signals by Signal Name .....	543
17.3	Signals by Function, Except for GPIO .....	545
17.4	GPIO Pins and Alternate Functions .....	547
17.5	Connections for Unused Signals .....	548
<b>18</b>	<b>Operating Characteristics .....</b>	<b>549</b>
<b>19</b>	<b>Electrical Characteristics .....</b>	<b>550</b>
19.1	DC Characteristics .....	550
19.1.1	Maximum Ratings .....	550
19.1.2	Recommended DC Operating Conditions .....	550
19.1.3	On-Chip Low Drop-Out (LDO) Regulator Characteristics .....	551
19.1.4	GPIO Module Characteristics .....	551
19.1.5	Power Specifications .....	551
19.1.6	Flash Memory Characteristics .....	552
19.2	AC Characteristics .....	552
19.2.1	Load Conditions .....	552
19.2.2	Clocks .....	553
19.2.3	JTAG and Boundary Scan .....	553
19.2.4	Reset .....	555

19.2.5 Sleep Modes .....	557
19.2.6 General-Purpose I/O (GPIO) .....	557
19.2.7 Analog-to-Digital Converter .....	558
19.2.8 Synchronous Serial Interface (SSI) .....	559
19.2.9 Inter-Integrated Circuit ( $I^2C$ ) Interface .....	561
19.2.10 Analog Comparator .....	562
<b>A      Serial Flash Loader .....</b>	<b>563</b>
A.1    Serial Flash Loader .....	563
A.2    Interfaces .....	563
A.2.1    UART .....	563
A.2.2    SSI .....	563
A.3    Packet Handling .....	564
A.3.1    Packet Format .....	564
A.3.2    Sending Packets .....	564
A.3.3    Receiving Packets .....	564
A.4    Commands .....	565
A.4.1    COMMAND_PING (0X20) .....	565
A.4.2    COMMAND_GET_STATUS (0x23) .....	565
A.4.3    COMMAND_DOWNLOAD (0x21) .....	565
A.4.4    COMMAND_SEND_DATA (0x24) .....	566
A.4.5    COMMAND_RUN (0x22) .....	566
A.4.6    COMMAND_RESET (0x25) .....	566
<b>B      Register Quick Reference .....</b>	<b>568</b>
<b>C      Ordering and Contact Information .....</b>	<b>588</b>
C.1    Ordering Information .....	588
C.2    Part Markings .....	588
C.3    Kits .....	588
C.4    Support Information .....	589
<b>D      Package Information .....</b>	<b>590</b>
D.1    48-Pin LQFP Package .....	590
D.1.1    Package Dimensions .....	590
D.1.2    Tray Dimensions .....	592
D.1.3    Tape and Reel Dimensions .....	594
D.2    48-Pin QFN Package .....	595
D.2.1    Package Dimensions .....	595

# List of Figures

Figure 1-1.	Stellaris LM3S811 Microcontroller High-Level Block Diagram .....	36
Figure 1-2.	LM3S811 Controller System-Level Block Diagram .....	43
Figure 2-1.	CPU Block Diagram .....	46
Figure 2-2.	TPIU Block Diagram .....	47
Figure 2-3.	Cortex-M3 Register Set .....	49
Figure 2-4.	Bit-Band Mapping .....	69
Figure 2-5.	Data Storage .....	70
Figure 2-6.	Vector Table .....	75
Figure 2-7.	Exception Stack Frame .....	77
Figure 3-1.	SRD Use Example .....	92
Figure 4-1.	JTAG Module Block Diagram .....	145
Figure 4-2.	Test Access Port State Machine .....	149
Figure 4-3.	IDCODE Register Format .....	153
Figure 4-4.	BYPASS Register Format .....	154
Figure 4-5.	Boundary Scan Register Format .....	154
Figure 5-1.	Basic $\overline{RST}$ Configuration .....	157
Figure 5-2.	External Circuitry to Extend Power-On Reset .....	158
Figure 5-3.	Reset Circuit Controlled by Switch .....	158
Figure 5-4.	Main Clock Tree .....	161
Figure 6-1.	Flash Block Diagram .....	214
Figure 7-1.	GPIO Module Block Diagram .....	233
Figure 7-2.	GPIO Port Block Diagram .....	238
Figure 7-3.	GPIO DATA Write Example .....	239
Figure 7-4.	GPIO DATA Read Example .....	239
Figure 8-1.	GPTM Module Block Diagram .....	276
Figure 8-2.	16-Bit Input Edge Count Mode Example .....	281
Figure 8-3.	16-Bit Input Edge Time Mode Example .....	282
Figure 8-4.	16-Bit PWM Mode Example .....	283
Figure 9-1.	WDT Module Block Diagram .....	313
Figure 10-1.	ADC Module Block Diagram .....	337
Figure 10-2.	Differential Sampling Range, $V_{IN\_ODD} = 1.5\text{ V}$ .....	341
Figure 10-3.	Differential Sampling Range, $V_{IN\_ODD} = 0.75\text{ V}$ .....	341
Figure 10-4.	Differential Sampling Range, $V_{IN\_ODD} = 2.25\text{ V}$ .....	342
Figure 10-5.	Internal Temperature Sensor Characteristic .....	343
Figure 11-1.	UART Module Block Diagram .....	374
Figure 11-2.	UART Character Frame .....	375
Figure 12-1.	SSI Module Block Diagram .....	413
Figure 12-2.	TI Synchronous Serial Frame Format (Single Transfer) .....	416
Figure 12-3.	TI Synchronous Serial Frame Format (Continuous Transfer) .....	417
Figure 12-4.	Freescale SPI Format (Single Transfer) with $SPO=0$ and $SPH=0$ .....	417
Figure 12-5.	Freescale SPI Format (Continuous Transfer) with $SPO=0$ and $SPH=0$ .....	418
Figure 12-6.	Freescale SPI Frame Format with $SPO=0$ and $SPH=1$ .....	419
Figure 12-7.	Freescale SPI Frame Format (Single Transfer) with $SPO=1$ and $SPH=0$ .....	419
Figure 12-8.	Freescale SPI Frame Format (Continuous Transfer) with $SPO=1$ and $SPH=0$ .....	420
Figure 12-9.	Freescale SPI Frame Format with $SPO=1$ and $SPH=1$ .....	421
Figure 12-10.	MICROWIRE Frame Format (Single Frame) .....	421

Figure 12-11.	MICROWIRE Frame Format (Continuous Transfer) .....	422
Figure 12-12.	MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements .....	423
Figure 13-1.	I <sup>2</sup> C Block Diagram .....	452
Figure 13-2.	I <sup>2</sup> C Bus Configuration .....	453
Figure 13-3.	START and STOP Conditions .....	453
Figure 13-4.	Complete Data Transfer with a 7-Bit Address .....	454
Figure 13-5.	R/S Bit in First Byte .....	454
Figure 13-6.	Data Validity During Bit Transfer on the I <sup>2</sup> C Bus .....	454
Figure 13-7.	Master Single SEND .....	458
Figure 13-8.	Master Single RECEIVE .....	459
Figure 13-9.	Master Burst SEND .....	460
Figure 13-10.	Master Burst RECEIVE .....	461
Figure 13-11.	Master Burst RECEIVE after Burst SEND .....	462
Figure 13-12.	Master Burst SEND after Burst RECEIVE .....	463
Figure 13-13.	Slave Command Sequence .....	464
Figure 14-1.	Analog Comparator Module Block Diagram .....	488
Figure 14-2.	Structure of Comparator Unit .....	489
Figure 14-3.	Comparator Internal Reference Structure .....	490
Figure 15-1.	PWM Unit Diagram .....	501
Figure 15-2.	PWM Module Block Diagram .....	502
Figure 15-3.	PWM Count-Down Mode .....	504
Figure 15-4.	PWM Count-Up/Down Mode .....	504
Figure 15-5.	PWM Generation Example In Count-Up/Down Mode .....	505
Figure 15-6.	PWM Dead-Band Generator .....	505
Figure 16-1.	48-Pin QFP Package Pin Diagram .....	539
Figure 16-2.	48-Pin QFN Package Pin Diagram .....	540
Figure 19-1.	Load Conditions .....	553
Figure 19-2.	JTAG Test Clock Input Timing .....	554
Figure 19-3.	JTAG Test Access Port (TAP) Timing .....	555
Figure 19-4.	JTAG TRST Timing .....	555
Figure 19-5.	External Reset Timing (RST) .....	556
Figure 19-6.	Power-On Reset Timing .....	556
Figure 19-7.	Brown-Out Reset Timing .....	556
Figure 19-8.	Software Reset Timing .....	557
Figure 19-9.	Watchdog Reset Timing .....	557
Figure 19-10.	LDO Reset Timing .....	557
Figure 19-11.	ADC Input Equivalency Diagram .....	559
Figure 19-12.	SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement .....	560
Figure 19-13.	SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer .....	560
Figure 19-14.	SSI Timing for SPI Frame Format (FRF=00), with SPH=1 .....	561
Figure 19-15.	I <sup>2</sup> C Timing .....	562
Figure D-1.	Stellaris LM3S811 48-Pin LQFP Package .....	590
Figure D-2.	48-Pin LQFP Tray Dimensions .....	592
Figure D-3.	48-Pin LQFP Tape and Reel Dimensions .....	594
Figure D-4.	Stellaris LM3S811 48-Pin QFN Package .....	595

# List of Tables

Table 1.	Revision History .....	21
Table 2.	Documentation Conventions .....	26
Table 2-1.	Summary of Processor Mode, Privilege Level, and Stack Use .....	49
Table 2-2.	Processor Register Map .....	50
Table 2-3.	PSR Register Combinations .....	55
Table 2-4.	Memory Map .....	63
Table 2-5.	Memory Access Behavior .....	65
Table 2-6.	SRAM Memory Bit-Banding Regions .....	67
Table 2-7.	Peripheral Memory Bit-Banding Regions .....	67
Table 2-8.	Exception Types .....	73
Table 2-9.	Interrupts .....	74
Table 2-10.	Exception Return Behavior .....	78
Table 2-11.	Faults .....	79
Table 2-12.	Fault Status and Fault Address Registers .....	80
Table 2-13.	Cortex-M3 Instruction Summary .....	82
Table 3-1.	Core Peripheral Register Regions .....	86
Table 3-2.	Memory Attributes Summary .....	89
Table 3-3.	TEX, S, C, and B Bit Field Encoding .....	92
Table 3-4.	Cache Policy for Memory Attribute Encoding .....	93
Table 3-5.	AP Bit Field Encoding .....	93
Table 3-6.	Memory Region Attributes for Stellaris Microcontrollers .....	93
Table 3-7.	Peripherals Register Map .....	94
Table 3-8.	Interrupt Priority Levels .....	113
Table 3-9.	Example SIZE Field Values .....	141
Table 4-1.	JTAG_SWD_SWO Signals (48QFP) .....	145
Table 4-2.	JTAG_SWD_SWO Signals (48QFN) .....	146
Table 4-3.	JTAG Port Pins Reset State .....	146
Table 4-4.	JTAG Instruction Register Commands .....	151
Table 5-1.	System Control & Clocks Signals (48QFP) .....	155
Table 5-2.	System Control & Clocks Signals (48QFN) .....	155
Table 5-3.	Reset Sources .....	156
Table 5-4.	Clock Source Options .....	161
Table 5-5.	Possible System Clock Frequencies Using the SYSDIV Field .....	162
Table 5-6.	System Control Register Map .....	165
Table 5-7.	PLL Mode Control .....	178
Table 6-1.	Flash Protection Policy Combinations .....	215
Table 6-2.	Flash Register Map .....	220
Table 7-1.	GPIO Pins With Non-Zero Reset Values .....	234
Table 7-2.	GPIO Pins and Alternate Functions (48QFP) .....	234
Table 7-3.	GPIO Pins and Alternate Functions (48QFN) .....	235
Table 7-4.	GPIO Signals (48QFP) .....	236
Table 7-5.	GPIO Signals (48QFN) .....	236
Table 7-6.	GPIO Pad Configuration Examples .....	241
Table 7-7.	GPIO Interrupt Configuration Example .....	241
Table 7-8.	GPIO Register Map .....	242
Table 8-1.	Available CCP Pins .....	276

Table 8-2.	General-Purpose Timers Signals (48QFP) .....	277
Table 8-3.	General-Purpose Timers Signals (48QFN) .....	277
Table 8-4.	16-Bit Timer With Prescaler Configurations .....	279
Table 8-5.	Timers Register Map .....	286
Table 9-1.	Watchdog Timer Register Map .....	314
Table 10-1.	ADC Signals (48QFP) .....	337
Table 10-2.	ADC Signals (48QFN) .....	337
Table 10-3.	Samples and FIFO Depth of Sequencers .....	338
Table 10-4.	Differential Sampling Pairs .....	340
Table 10-5.	ADC Register Map .....	344
Table 11-1.	UART Signals (48QFP) .....	374
Table 11-2.	UART Signals (48QFN) .....	375
Table 11-3.	UART Register Map .....	379
Table 12-1.	SSI Signals (48QFP) .....	414
Table 12-2.	SSI Signals (48QFN) .....	414
Table 12-3.	SSI Register Map .....	424
Table 13-1.	I <sup>2</sup> C Signals (48QFP) .....	452
Table 13-2.	I <sup>2</sup> C Signals (48QFN) .....	452
Table 13-3.	Examples of I <sup>2</sup> C Master Timer Period versus Speed Mode .....	455
Table 13-4.	Inter-Integrated Circuit (I <sup>2</sup> C) Interface Register Map .....	465
Table 13-5.	Write Field Decoding for I2CMCS[3:0] Field (Sheet 1 of 3) .....	470
Table 14-1.	Analog Comparators Signals (48QFP) .....	489
Table 14-2.	Analog Comparators Signals (48QFN) .....	489
Table 14-3.	Comparator 0 Operating Modes .....	490
Table 14-4.	Internal Reference Voltage and ACREFCTL Field Values .....	490
Table 14-5.	Analog Comparators Register Map .....	492
Table 15-1.	PWM Signals (48QFP) .....	502
Table 15-2.	PWM Signals (48QFN) .....	502
Table 15-3.	PWM Register Map .....	507
Table 17-1.	Signals by Pin Number .....	541
Table 17-2.	Signals by Signal Name .....	543
Table 17-3.	Signals by Function, Except for GPIO .....	545
Table 17-4.	GPIO Pins and Alternate Functions .....	547
Table 17-5.	Connections for Unused Signals .....	548
Table 18-1.	Temperature Characteristics .....	549
Table 18-2.	Thermal Characteristics .....	549
Table 18-3.	ESD Absolute Maximum Ratings .....	549
Table 19-1.	Maximum Ratings .....	550
Table 19-2.	Recommended DC Operating Conditions .....	550
Table 19-3.	LDO Regulator Characteristics .....	551
Table 19-4.	GPIO Module DC Characteristics .....	551
Table 19-5.	Detailed Power Specifications .....	552
Table 19-6.	Flash Memory Characteristics .....	552
Table 19-7.	Phase Locked Loop (PLL) Characteristics .....	553
Table 19-8.	Clock Characteristics .....	553
Table 19-9.	System Clock Characteristics with ADC Operation .....	553
Table 19-10.	JTAG Characteristics .....	553
Table 19-11.	Reset Characteristics .....	555

---

Table 19-12.	Sleep Modes AC Characteristics .....	557
Table 19-13.	GPIO Characteristics .....	558
Table 19-14.	ADC Characteristics .....	558
Table 19-15.	ADC Module Internal Reference Characteristics .....	559
Table 19-16.	SSI Characteristics .....	559
Table 19-17.	I <sup>2</sup> C Characteristics .....	561
Table 19-18.	Analog Comparator Characteristics .....	562
Table 19-19.	Analog Comparator Voltage Reference Characteristics .....	562

# List of Registers

<b>The Cortex-M3 Processor .....</b>	<b>44</b>
Register 1: Cortex General-Purpose Register 0 (R0) .....	51
Register 2: Cortex General-Purpose Register 1 (R1) .....	51
Register 3: Cortex General-Purpose Register 2 (R2) .....	51
Register 4: Cortex General-Purpose Register 3 (R3) .....	51
Register 5: Cortex General-Purpose Register 4 (R4) .....	51
Register 6: Cortex General-Purpose Register 5 (R5) .....	51
Register 7: Cortex General-Purpose Register 6 (R6) .....	51
Register 8: Cortex General-Purpose Register 7 (R7) .....	51
Register 9: Cortex General-Purpose Register 8 (R8) .....	51
Register 10: Cortex General-Purpose Register 9 (R9) .....	51
Register 11: Cortex General-Purpose Register 10 (R10) .....	51
Register 12: Cortex General-Purpose Register 11 (R11) .....	51
Register 13: Cortex General-Purpose Register 12 (R12) .....	51
Register 14: Stack Pointer (SP) .....	52
Register 15: Link Register (LR) .....	53
Register 16: Program Counter (PC) .....	54
Register 17: Program Status Register (PSR) .....	55
Register 18: Priority Mask Register (PRIMASK) .....	59
Register 19: Fault Mask Register (FAULTMASK) .....	60
Register 20: Base Priority Mask Register (BASEPRI) .....	61
Register 21: Control Register (CONTROL) .....	62
<b>Cortex-M3 Peripherals .....</b>	<b>86</b>
Register 1: SysTick Control and Status Register (STCTRL), offset 0x010 .....	96
Register 2: SysTick Reload Value Register (STRELOAD), offset 0x014 .....	98
Register 3: SysTick Current Value Register (STCURRENT), offset 0x018 .....	99
Register 4: Interrupt 0-29 Set Enable (EN0), offset 0x100 .....	100
Register 5: Interrupt 0-29 Clear Enable (DIS0), offset 0x180 .....	101
Register 6: Interrupt 0-29 Set Pending (PEND0), offset 0x200 .....	102
Register 7: Interrupt 0-29 Clear Pending (UNPEND0), offset 0x280 .....	103
Register 8: Interrupt 0-29 Active Bit (ACTIVE0), offset 0x300 .....	104
Register 9: Interrupt 0-3 Priority (PRI0), offset 0x400 .....	105
Register 10: Interrupt 4-7 Priority (PRI1), offset 0x404 .....	105
Register 11: Interrupt 8-11 Priority (PRI2), offset 0x408 .....	105
Register 12: Interrupt 12-15 Priority (PRI3), offset 0x40C .....	105
Register 13: Interrupt 16-19 Priority (PRI4), offset 0x410 .....	105
Register 14: Interrupt 20-23 Priority (PRI5), offset 0x414 .....	105
Register 15: Interrupt 24-27 Priority (PRI6), offset 0x418 .....	105
Register 16: Interrupt 28-29 Priority (PRI7), offset 0x41C .....	105
Register 17: Software Trigger Interrupt (SWTRIG), offset 0xF00 .....	107
Register 18: CPU ID Base (CPUID), offset 0xD00 .....	108
Register 19: Interrupt Control and State (INTCTRL), offset 0xD04 .....	109
Register 20: Vector Table Offset (VTABLE), offset 0xD08 .....	112
Register 21: Application Interrupt and Reset Control (APINT), offset 0xD0C .....	113
Register 22: System Control (SYSCTRL), offset 0xD10 .....	115

Register 23:	Configuration and Control (CFGCTRL), offset 0xD14 .....	117
Register 24:	System Handler Priority 1 (SYSPRI1), offset 0xD18 .....	119
Register 25:	System Handler Priority 2 (SYSPRI2), offset 0xD1C .....	120
Register 26:	System Handler Priority 3 (SYSPRI3), offset 0xD20 .....	121
Register 27:	System Handler Control and State (SYSHNDCTRL), offset 0xD24 .....	122
Register 28:	Configurable Fault Status (FAULTSTAT), offset 0xD28 .....	126
Register 29:	Hard Fault Status (HFAULTSTAT), offset 0xD2C .....	132
Register 30:	Memory Management Fault Address (MMADDR), offset 0xD34 .....	133
Register 31:	Bus Fault Address (FAULTADDR), offset 0xD38 .....	134
Register 32:	MPU Type (MPUTYPE), offset 0xD90 .....	135
Register 33:	MPU Control (MPUCTRL), offset 0xD94 .....	136
Register 34:	MPU Region Number (MPUNUMBER), offset 0xD98 .....	138
Register 35:	MPU Region Base Address (MPUBASE), offset 0xD9C .....	139
Register 36:	MPU Region Base Address Alias 1 (MPUBASE1), offset 0xDA4 .....	139
Register 37:	MPU Region Base Address Alias 2 (MPUBASE2), offset 0xDAC .....	139
Register 38:	MPU Region Base Address Alias 3 (MPUBASE3), offset 0xDB4 .....	139
Register 39:	MPU Region Attribute and Size (MPUATTR), offset 0xDA0 .....	141
Register 40:	MPU Region Attribute and Size Alias 1 (MPUATTR1), offset 0xDA8 .....	141
Register 41:	MPU Region Attribute and Size Alias 2 (MPUATTR2), offset 0xDB0 .....	141
Register 42:	MPU Region Attribute and Size Alias 3 (MPUATTR3), offset 0xDB8 .....	141
<b>System Control .....</b>	<b>155</b>	
Register 1:	Device Identification 0 (DID0), offset 0x000 .....	167
Register 2:	Power-On and Brown-Out Reset Control (PBORCTL), offset 0x030 .....	169
Register 3:	LDO Power Control (LDOPCTL), offset 0x034 .....	170
Register 4:	Raw Interrupt Status (RIS), offset 0x050 .....	171
Register 5:	Interrupt Mask Control (IMC), offset 0x054 .....	172
Register 6:	Masked Interrupt Status and Clear (MISC), offset 0x058 .....	173
Register 7:	Reset Cause (RESC), offset 0x05C .....	174
Register 8:	Run-Mode Clock Configuration (RCC), offset 0x060 .....	175
Register 9:	XTAL to PLL Translation (PLLCFG), offset 0x064 .....	179
Register 10:	Deep Sleep Clock Configuration (DSLPCLKCFG), offset 0x144 .....	180
Register 11:	Clock Verification Clear (CLKVCLR), offset 0x150 .....	181
Register 12:	Allow Unregulated LDO to Reset the Part (LDOARST), offset 0x160 .....	182
Register 13:	Device Identification 1 (DID1), offset 0x004 .....	183
Register 14:	Device Capabilities 0 (DC0), offset 0x008 .....	185
Register 15:	Device Capabilities 1 (DC1), offset 0x010 .....	186
Register 16:	Device Capabilities 2 (DC2), offset 0x014 .....	188
Register 17:	Device Capabilities 3 (DC3), offset 0x018 .....	190
Register 18:	Device Capabilities 4 (DC4), offset 0x01C .....	192
Register 19:	Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100 .....	193
Register 20:	Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110 .....	195
Register 21:	Deep Sleep Mode Clock Gating Control Register 0 (DCGC0), offset 0x120 .....	197
Register 22:	Run Mode Clock Gating Control Register 1 (RCGC1), offset 0x104 .....	199
Register 23:	Sleep Mode Clock Gating Control Register 1 (SCGC1), offset 0x114 .....	201
Register 24:	Deep Sleep Mode Clock Gating Control Register 1 (DCGC1), offset 0x124 .....	203
Register 25:	Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108 .....	205
Register 26:	Sleep Mode Clock Gating Control Register 2 (SCGC2), offset 0x118 .....	206
Register 27:	Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128 .....	208

Register 28:	Software Reset Control 0 (SRCR0), offset 0x040 .....	210
Register 29:	Software Reset Control 1 (SRCR1), offset 0x044 .....	211
Register 30:	Software Reset Control 2 (SRCR2), offset 0x048 .....	213
<b>Internal Memory .....</b>		<b>214</b>
Register 1:	Flash Memory Address (FMA), offset 0x000 .....	221
Register 2:	Flash Memory Data (FMD), offset 0x004 .....	222
Register 3:	Flash Memory Control (FMC), offset 0x008 .....	223
Register 4:	Flash Controller Raw Interrupt Status (FCRIS), offset 0x00C .....	225
Register 5:	Flash Controller Interrupt Mask (FCIM), offset 0x010 .....	226
Register 6:	Flash Controller Masked Interrupt Status and Clear (FCMISC), offset 0x014 .....	227
Register 7:	USec Reload (USECRL), offset 0x140 .....	229
Register 8:	Flash Memory Protection Read Enable (FMPRE), offset 0x130 .....	230
Register 9:	Flash Memory Protection Program Enable (FMPPE), offset 0x134 .....	231
<b>General-Purpose Input/Outputs (GPIOs) .....</b>		<b>232</b>
Register 1:	GPIO Data (GPIODATA), offset 0x000 .....	244
Register 2:	GPIO Direction (GPIODIR), offset 0x400 .....	245
Register 3:	GPIO Interrupt Sense (GPIOIS), offset 0x404 .....	246
Register 4:	GPIO Interrupt Both Edges (GPIOIBE), offset 0x408 .....	247
Register 5:	GPIO Interrupt Event (GPIOEV), offset 0x40C .....	248
Register 6:	GPIO Interrupt Mask (GPIOIM), offset 0x410 .....	249
Register 7:	GPIO Raw Interrupt Status (GPIORIS), offset 0x414 .....	250
Register 8:	GPIO Masked Interrupt Status (GPIOVIS), offset 0x418 .....	251
Register 9:	GPIO Interrupt Clear (GPIOICR), offset 0x41C .....	252
Register 10:	GPIO Alternate Function Select (GPIOAFSEL), offset 0x420 .....	253
Register 11:	GPIO 2-mA Drive Select (GPIODR2R), offset 0x500 .....	255
Register 12:	GPIO 4-mA Drive Select (GPIODR4R), offset 0x504 .....	256
Register 13:	GPIO 8-mA Drive Select (GPIODR8R), offset 0x508 .....	257
Register 14:	GPIO Open Drain Select (GPIOODR), offset 0x50C .....	258
Register 15:	GPIO Pull-Up Select (GPIOPUR), offset 0x510 .....	259
Register 16:	GPIO Pull-Down Select (GPIOPDR), offset 0x514 .....	260
Register 17:	GPIO Slew Rate Control Select (GPIOSLR), offset 0x518 .....	261
Register 18:	GPIO Digital Enable (GPIODEN), offset 0x51C .....	262
Register 19:	GPIO Peripheral Identification 4 (GPIOPeriphID4), offset 0xFD0 .....	263
Register 20:	GPIO Peripheral Identification 5 (GPIOPeriphID5), offset 0xFD4 .....	264
Register 21:	GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8 .....	265
Register 22:	GPIO Peripheral Identification 7 (GPIOPeriphID7), offset 0xFDC .....	266
Register 23:	GPIO Peripheral Identification 0 (GPIOPeriphID0), offset 0xFE0 .....	267
Register 24:	GPIO Peripheral Identification 1 (GPIOPeriphID1), offset 0xFE4 .....	268
Register 25:	GPIO Peripheral Identification 2 (GPIOPeriphID2), offset 0xFE8 .....	269
Register 26:	GPIO Peripheral Identification 3 (GPIOPeriphID3), offset 0xFEC .....	270
Register 27:	GPIO PrimeCell Identification 0 (GPIOCellID0), offset 0xFF0 .....	271
Register 28:	GPIO PrimeCell Identification 1 (GPIOCellID1), offset 0xFF4 .....	272
Register 29:	GPIO PrimeCell Identification 2 (GPIOCellID2), offset 0xFF8 .....	273
Register 30:	GPIO PrimeCell Identification 3 (GPIOCellID3), offset 0xFFC .....	274
<b>General-Purpose Timers .....</b>		<b>275</b>
Register 1:	GPTM Configuration (GPTMCFG), offset 0x000 .....	288
Register 2:	GPTM TimerA Mode (GPTMTAMR), offset 0x004 .....	289
Register 3:	GPTM TimerB Mode (GPTMTBMR), offset 0x008 .....	291

Register 4:	GPTM Control (GPTMCTL), offset 0x00C .....	293
Register 5:	GPTM Interrupt Mask (GPTMIMR), offset 0x018 .....	296
Register 6:	GPTM Raw Interrupt Status (GPTMRIS), offset 0x01C .....	298
Register 7:	GPTM Masked Interrupt Status (GPTMMIS), offset 0x020 .....	299
Register 8:	GPTM Interrupt Clear (GPTMICR), offset 0x024 .....	300
Register 9:	GPTM TimerA Interval Load (GPTMTAILR), offset 0x028 .....	302
Register 10:	GPTM TimerB Interval Load (GPTMTBILR), offset 0x02C .....	303
Register 11:	GPTM TimerA Match (GPTMTAMATCHR), offset 0x030 .....	304
Register 12:	GPTM TimerB Match (GPTMTBMATCHR), offset 0x034 .....	305
Register 13:	GPTM TimerA Prescale (GPTMTAPR), offset 0x038 .....	306
Register 14:	GPTM TimerB Prescale (GPTMTBPR), offset 0x03C .....	307
Register 15:	GPTM TimerA Prescale Match (GPTMTAPMR), offset 0x040 .....	308
Register 16:	GPTM TimerB Prescale Match (GPTMTBPMR), offset 0x044 .....	309
Register 17:	GPTM TimerA (GPTMTAR), offset 0x048 .....	310
Register 18:	GPTM TimerB (GPTMTBR), offset 0x04C .....	311
<b>Watchdog Timer .....</b>	<b>312</b>	
Register 1:	Watchdog Load (WDTLOAD), offset 0x000 .....	316
Register 2:	Watchdog Value (WDTVVALUE), offset 0x004 .....	317
Register 3:	Watchdog Control (WDTCTL), offset 0x008 .....	318
Register 4:	Watchdog Interrupt Clear (WDTICR), offset 0x00C .....	319
Register 5:	Watchdog Raw Interrupt Status (WDTRIS), offset 0x010 .....	320
Register 6:	Watchdog Masked Interrupt Status (WDTMIS), offset 0x014 .....	321
Register 7:	Watchdog Test (WDTTEST), offset 0x418 .....	322
Register 8:	Watchdog Lock (WDTLOCK), offset 0xC00 .....	323
Register 9:	Watchdog Peripheral Identification 4 (WDTPeriphID4), offset 0xFD0 .....	324
Register 10:	Watchdog Peripheral Identification 5 (WDTPeriphID5), offset 0xFD4 .....	325
Register 11:	Watchdog Peripheral Identification 6 (WDTPeriphID6), offset 0xFD8 .....	326
Register 12:	Watchdog Peripheral Identification 7 (WDTPeriphID7), offset 0xFDC .....	327
Register 13:	Watchdog Peripheral Identification 0 (WDTPeriphID0), offset 0xFE0 .....	328
Register 14:	Watchdog Peripheral Identification 1 (WDTPeriphID1), offset 0xFE4 .....	329
Register 15:	Watchdog Peripheral Identification 2 (WDTPeriphID2), offset 0xFE8 .....	330
Register 16:	Watchdog Peripheral Identification 3 (WDTPeriphID3), offset 0xFEC .....	331
Register 17:	Watchdog PrimeCell Identification 0 (WDTPCellID0), offset 0xFF0 .....	332
Register 18:	Watchdog PrimeCell Identification 1 (WDTPCellID1), offset 0xFF4 .....	333
Register 19:	Watchdog PrimeCell Identification 2 (WDTPCellID2), offset 0xFF8 .....	334
Register 20:	Watchdog PrimeCell Identification 3 (WDTPCellID3 ), offset 0xFFC .....	335
<b>Analog-to-Digital Converter (ADC) .....</b>	<b>336</b>	
Register 1:	ADC Active Sample Sequencer (ADCACTSS), offset 0x000 .....	346
Register 2:	ADC Raw Interrupt Status (ADCRIS), offset 0x004 .....	347
Register 3:	ADC Interrupt Mask (ADCIM), offset 0x008 .....	348
Register 4:	ADC Interrupt Status and Clear (ADCISC), offset 0x00C .....	349
Register 5:	ADC Overflow Status (ADCOSTAT), offset 0x010 .....	350
Register 6:	ADC Event Multiplexer Select (ADCEMUX), offset 0x014 .....	351
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 .....	358
Register 10:	ADC Sample Averaging Control (ADCSAC), offset 0x030 .....	359
Register 11:	ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040 .....	360

Register 12:	ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044 .....	362
Register 13:	ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048 .....	365
Register 14:	ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068 .....	365
Register 15:	ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088 .....	365
Register 16:	ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8 .....	365
Register 17:	ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C .....	366
Register 18:	ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C .....	366
Register 19:	ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C .....	366
Register 20:	ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC .....	366
Register 21:	ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060 .....	367
Register 22:	ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080 .....	367
Register 23:	ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064 .....	368
Register 24:	ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084 .....	368
Register 25:	ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0 .....	370
Register 26:	ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4 .....	371
Register 27:	ADC Test Mode Loopback (ADCTMLB), offset 0x100 .....	372
<b>Universal Asynchronous Receivers/Transmitters (UARTs) .....</b>	<b>373</b>	
Register 1:	UART Data (UARTDR), offset 0x000 .....	381
Register 2:	UART Receive Status/Error Clear (UARTRSR/UARTECR), offset 0x004 .....	383
Register 3:	UART Flag (UARTFR), offset 0x018 .....	385
Register 4:	UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024 .....	387
Register 5:	UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028 .....	388
Register 6:	UART Line Control (UARTLCRH), offset 0x02C .....	389
Register 7:	UART Control (UARTCTL), offset 0x030 .....	391
Register 8:	UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034 .....	393
Register 9:	UART Interrupt Mask (UARTIM), offset 0x038 .....	395
Register 10:	UART Raw Interrupt Status (UARTRIS), offset 0x03C .....	397
Register 11:	UART Masked Interrupt Status (UARTMIS), offset 0x040 .....	398
Register 12:	UART Interrupt Clear (UARTICR), offset 0x044 .....	399
Register 13:	UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0 .....	401
Register 14:	UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4 .....	402
Register 15:	UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8 .....	403
Register 16:	UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC .....	404
Register 17:	UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0 .....	405
Register 18:	UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4 .....	406
Register 19:	UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8 .....	407
Register 20:	UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC .....	408
Register 21:	UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0 .....	409
Register 22:	UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4 .....	410
Register 23:	UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8 .....	411
Register 24:	UART PrimeCell Identification 3 (UARTPCellID3), offset 0xFFC .....	412
<b>Synchronous Serial Interface (SSI) .....</b>	<b>413</b>	
Register 1:	SSI Control 0 (SSICR0), offset 0x000 .....	426
Register 2:	SSI Control 1 (SSICR1), offset 0x004 .....	428
Register 3:	SSI Data (SSIDR), offset 0x008 .....	430
Register 4:	SSI Status (SSISR), offset 0x00C .....	431
Register 5:	SSI Clock Prescale (SSICPSR), offset 0x010 .....	433
Register 6:	SSI Interrupt Mask (SSIIM), offset 0x014 .....	434

Register 7:	SSI Raw Interrupt Status (SSIRIS), offset 0x018 .....	436
Register 8:	SSI Masked Interrupt Status (SSIMIS), offset 0x01C .....	437
Register 9:	SSI Interrupt Clear (SSIICR), offset 0x020 .....	438
Register 10:	SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0 .....	439
Register 11:	SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4 .....	440
Register 12:	SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8 .....	441
Register 13:	SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC .....	442
Register 14:	SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0 .....	443
Register 15:	SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4 .....	444
Register 16:	SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8 .....	445
Register 17:	SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC .....	446
Register 18:	SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0 .....	447
Register 19:	SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4 .....	448
Register 20:	SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8 .....	449
Register 21:	SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC .....	450
<b>Inter-Integrated Circuit (I<sup>2</sup>C) Interface .....</b>	<b>451</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 0x800 .....	480
Register 11:	I <sup>2</sup> C Slave Control/Status (I2CSCSR), offset 0x804 .....	481
Register 12:	I <sup>2</sup> C Slave Data (I2CSDR), offset 0x808 .....	483
Register 13:	I <sup>2</sup> C Slave Interrupt Mask (I2CSIMR), offset 0x80C .....	484
Register 14:	I <sup>2</sup> C Slave Raw Interrupt Status (I2CSRIS), offset 0x810 .....	485
Register 15:	I <sup>2</sup> C Slave Masked Interrupt Status (I2CSMIS), offset 0x814 .....	486
Register 16:	I <sup>2</sup> C Slave Interrupt Clear (I2CSICR), offset 0x818 .....	487
<b>Analog Comparator .....</b>	<b>488</b>	
Register 1:	Analog Comparator Masked Interrupt Status (ACMIS), offset 0x000 .....	493
Register 2:	Analog Comparator Raw Interrupt Status (ACRIS), offset 0x004 .....	494
Register 3:	Analog Comparator Interrupt Enable (ACINTEN), offset 0x008 .....	495
Register 4:	Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x010 .....	496
Register 5:	Analog Comparator Status 0 (ACSTAT0), offset 0x020 .....	497
Register 6:	Analog Comparator Control 0 (ACCTL0), offset 0x024 .....	498
<b>Pulse Width Modulator (PWM) .....</b>	<b>500</b>	
Register 1:	PWM Master Control (PWMCTL), offset 0x000 .....	510
Register 2:	PWM Time Base Sync (PWMSYNC), offset 0x004 .....	511
Register 3:	PWM Output Enable (PWMenable), offset 0x008 .....	512
Register 4:	PWM Output Inversion (PwmInvert), offset 0x00C .....	513
Register 5:	PWM Output Fault (PwmFault), offset 0x010 .....	514
Register 6:	PWM Interrupt Enable (PwmInten), offset 0x014 .....	515
Register 7:	PWM Raw Interrupt Status (PwmRis), offset 0x018 .....	516

Register 8:	PWM Interrupt Status and Clear (PWMISC), offset 0x01C .....	517
Register 9:	PWM Status (PWMSTATUS), offset 0x020 .....	518
Register 10:	PWM0 Control (PWM0CTL), offset 0x040 .....	519
Register 11:	PWM1 Control (PWM1CTL), offset 0x080 .....	519
Register 12:	PWM2 Control (PWM2CTL), offset 0x0C0 .....	519
Register 13:	PWM0 Interrupt and Trigger Enable (PWM0INTEN), offset 0x044 .....	521
Register 14:	PWM1 Interrupt and Trigger Enable (PWM1INTEN), offset 0x084 .....	521
Register 15:	PWM2 Interrupt and Trigger Enable (PWM2INTEN), offset 0x0C4 .....	521
Register 16:	PWM0 Raw Interrupt Status (PWM0RIS), offset 0x048 .....	524
Register 17:	PWM1 Raw Interrupt Status (PWM1RIS), offset 0x088 .....	524
Register 18:	PWM2 Raw Interrupt Status (PWM2RIS), offset 0x0C8 .....	524
Register 19:	PWM0 Interrupt Status and Clear (PWM0ISC), offset 0x04C .....	525
Register 20:	PWM1 Interrupt Status and Clear (PWM1ISC), offset 0x08C .....	525
Register 21:	PWM2 Interrupt Status and Clear (PWM2ISC), offset 0x0CC .....	525
Register 22:	PWM0 Load (PWM0LOAD), offset 0x050 .....	526
Register 23:	PWM1 Load (PWM1LOAD), offset 0x090 .....	526
Register 24:	PWM2 Load (PWM2LOAD), offset 0x0D0 .....	526
Register 25:	PWM0 Counter (PWM0COUNT), offset 0x054 .....	527
Register 26:	PWM1 Counter (PWM1COUNT), offset 0x094 .....	527
Register 27:	PWM2 Counter (PWM2COUNT), offset 0x0D4 .....	527
Register 28:	PWM0 Compare A (PWM0CMPA), offset 0x058 .....	528
Register 29:	PWM1 Compare A (PWM1CMPA), offset 0x098 .....	528
Register 30:	PWM2 Compare A (PWM2CMPA), offset 0x0D8 .....	528
Register 31:	PWM0 Compare B (PWM0CMPB), offset 0x05C .....	529
Register 32:	PWM1 Compare B (PWM1CMPB), offset 0x09C .....	529
Register 33:	PWM2 Compare B (PWM2CMPB), offset 0x0DC .....	529
Register 34:	PWM0 Generator A Control (PWM0GENA), offset 0x060 .....	530
Register 35:	PWM1 Generator A Control (PWM1GENA), offset 0xA0 .....	530
Register 36:	PWM2 Generator A Control (PWM2GENA), offset 0xE0 .....	530
Register 37:	PWM0 Generator B Control (PWM0GENB), offset 0x064 .....	533
Register 38:	PWM1 Generator B Control (PWM1GENB), offset 0xA4 .....	533
Register 39:	PWM2 Generator B Control (PWM2GENB), offset 0xE4 .....	533
Register 40:	PWM0 Dead-Band Control (PWM0DBCTL), offset 0x068 .....	536
Register 41:	PWM1 Dead-Band Control (PWM1DBCTL), offset 0xA8 .....	536
Register 42:	PWM2 Dead-Band Control (PWM2DBCTL), offset 0xE8 .....	536
Register 43:	PWM0 Dead-Band Rising-Edge Delay (PWM0DBRISE), offset 0x06C .....	537
Register 44:	PWM1 Dead-Band Rising-Edge Delay (PWM1DBRISE), offset 0xAC .....	537
Register 45:	PWM2 Dead-Band Rising-Edge Delay (PWM2DBRISE), offset 0xEC .....	537
Register 46:	PWM0 Dead-Band Falling-Edge-Delay (PWM0DBFALL), offset 0x070 .....	538
Register 47:	PWM1 Dead-Band Falling-Edge-Delay (PWM1DBFALL), offset 0xB0 .....	538
Register 48:	PWM2 Dead-Band Falling-Edge-Delay (PWM2DBFALL), offset 0xF0 .....	538

# Revision History

The revision history table notes changes made between the indicated revisions of the LM3S811 data sheet.

**Table 1. Revision History**

Date	Revision	Description
July 2014	15852.2743	<ul style="list-style-type: none"> <li>■ In Internal Memory chapter, added sections on Execute-Only Protection, Read-Only Protection, and Permanently Disabling Debug.</li> <li>■ In UART chapter: <ul style="list-style-type: none"> <li>– Clarified that the transmit interrupt is based on a transition through level.</li> <li>– Corrected reset for <b>UART Raw Interrupt Status (UARTRIS)</b> register.</li> </ul> </li> <li>■ In Ordering and Contact Information appendix, moved orderable part numbers table to addendum.</li> <li>■ Additional minor data sheet clarifications and corrections.</li> </ul>
July 2012	12972.2532	<ul style="list-style-type: none"> <li>■ Marked 48-pin QFN package as OBSOLETE.</li> </ul>
June 2012	12739.2515	<ul style="list-style-type: none"> <li>■ In Reset Characteristics table, changed values and units for Internal reset timeout after hardware reset (R7).</li> <li>■ Removed extended temperature package.</li> <li>■ Minor data sheet clarifications and corrections.</li> </ul>
November 2011	11107	<ul style="list-style-type: none"> <li>■ Added module-specific pin tables to each chapter in the new Signal Description sections.</li> <li>■ In Timer chapter, clarified that in 16-Bit Input Edge Time Mode, the timer is capable of capturing three types of events: rising edge, falling edge, or both.</li> <li>■ In UART chapter, clarified interrupt behavior.</li> <li>■ In SSI chapter, corrected SSIClk in the figure "Synchronous Serial Frame Format (Single Transfer)".</li> <li>■ In Signal Tables chapter: <ul style="list-style-type: none"> <li>– Corrected pin numbers in table "Connections for Unused Signals" (other pin tables were correct).</li> <li>– Corrected buffer type for PWMn signals in pin tables.</li> </ul> </li> <li>■ In Electrical Characteristics chapter: <ul style="list-style-type: none"> <li>– Added parameter "Input voltage for a GPIO configured as an analog input" to the "Maximum Ratings" table.</li> <li>– Corrected Nom values for parameters "TCK clock Low time" and "TCK clock High time" in "JTAG Characteristics" table.</li> </ul> </li> <li>■ Additional minor data sheet clarifications and corrections.</li> </ul>

**Table 1. Revision History (continued)**

Date	Revision	Description
January 2011	9102	<ul style="list-style-type: none"> <li>■ In Application Interrupt and Reset Control (APINT) register, changed bit name from SYSRESETREQ to SYSRESREQ.</li> <li>■ Added DEBUG (Debug Priority) bit field to System Handler Priority 3 (SYSPRI3) register.</li> <li>■ Added "Reset Sources" table to System Control chapter.</li> <li>■ Removed mention of false-start bit detection in the UART chapter. This feature is not supported.</li> <li>■ Added note that specific module clocks must be enabled before that module's registers can be programmed. There must be a delay of 3 system clocks after the module clock is enabled before any of that module's registers are accessed.</li> <li>■ Changed I<sup>2</sup>C slave register base addresses and offsets to be relative to the I<sup>2</sup>C module base address of 0x4002.0000 , so register bases and offsets were changed for all I<sup>2</sup>C slave registers. Note that the hw_i2c.h file in the StellarisWare® Driver Library uses a base address of 0x4002.0800 for the I<sup>2</sup>C slave registers. Be aware when using registers with offsets between 0x800 and 0x818 that StellarisWare uses the old slave base address for these offsets.</li> <li>■ Corrected nonlinearity and offset error parameters (<math>E_L</math>, <math>E_D</math>, and <math>E_O</math>) in ADC Characteristics table.</li> <li>■ Added specification for maximum input voltage on a non-power pin when the microcontroller is unpowered (<math>V_{NON}</math> parameter in Maximum Ratings table).</li> <li>■ Additional minor data sheet clarifications and corrections.</li> </ul>
September 2010	7783	<ul style="list-style-type: none"> <li>■ Reorganized ARM Cortex-M3 Processor Core, Memory Map and Interrupts chapters, creating two new chapters, The Cortex-M3 Processor and Cortex-M3 Peripherals. Much additional content was added, including all the Cortex-M3 registers.</li> <li>■ Changed register names to be consistent with StellarisWare names: the Cortex-M3 <b>Interrupt Control and Status (ICSR)</b> register to the <b>Interrupt Control and State (INTCTRL)</b> register, and the Cortex-M3 <b>Interrupt Set Enable (SETNA)</b> register to the <b>Interrupt 0-31 Set Enable (ENO)</b> register.</li> <li>■ Added clarification of instruction execution during Flash operations.</li> <li>■ Modified Figure 7-2 on page 238 to clarify operation of the GPIO inputs when used as an alternate function.</li> <li>■ Added caution not to apply a Low value to PB7 when debugging; a Low value on the pin causes the JTAG controller to be reset, resulting in a loss of JTAG communication.</li> <li>■ In General-Purpose Timers chapter, clarified operation of the 32-bit RTC mode.</li> <li>■ Added missing table "Connections for Unused Signals" (Table 17-5 on page 548).</li> <li>■ In Electrical Characteristics chapter: <ul style="list-style-type: none"> <li>– Added <math>I_{LKG}</math> parameter (GPIO input leakage current) to Table 19-4 on page 551.</li> <li>– Corrected values for <math>t_{CLKRF}</math> parameter (SSIClk rise/fall time) in Table 19-16 on page 559.</li> </ul> </li> <li>■ Added dimensions for Tray and Tape and Reel shipping mediums.</li> </ul>
June 2010	7393	<ul style="list-style-type: none"> <li>■ Corrected base address for SRAM in architectural overview chapter.</li> <li>■ Clarified system clock operation, adding content to "Clock Control" on page 160.</li> <li>■ In Signal Tables chapter, added table "Connections for Unused Signals."</li> <li>■ In "Reset Characteristics" table, corrected value for supply voltage (VDD) rise time.</li> <li>■ Additional minor data sheet clarifications and corrections.</li> </ul>

**Table 1. Revision History (continued)**

Date	Revision	Description
April 2010	7004	<ul style="list-style-type: none"> <li>■ Added caution note to the <b>I<sup>2</sup>C Master Timer Period (I2CMTPR)</b> register description and changed field width to 7 bits.</li> <li>■ Added note about <b>RST</b> signal routing.</li> <li>■ Clarified the function of the <b>TnSTALL</b> bit in the <b>GPTMCTL</b> register.</li> <li>■ Additional minor data sheet clarifications and corrections.</li> </ul>
January 2010	6712	<ul style="list-style-type: none"> <li>■ In "System Control" section, clarified Debug Access Port operation after Sleep modes.</li> <li>■ Clarified wording on Flash memory access errors.</li> <li>■ Added section on Flash interrupts.</li> <li>■ Changed the reset value of the <b>ADC Sample Sequence Result FIFO n (ADCSSFIFO<sub>n</sub>)</b> registers to be indeterminate.</li> <li>■ Clarified operation of SSI transmit FIFO.</li> <li>■ Made these changes to the Operating Characteristics chapter: <ul style="list-style-type: none"> <li>– Added storage temperature ratings to "Temperature Characteristics" table</li> <li>– Added "ESD Absolute Maximum Ratings" table</li> </ul> </li> <li>■ Made these changes to the Electrical Characteristics chapter: <ul style="list-style-type: none"> <li>– In "Flash Memory Characteristics" table, corrected Mass erase time</li> <li>– Added sleep and deep-sleep wake-up times ("Sleep Modes AC Characteristics" table)</li> <li>– In "Reset Characteristics" table, corrected supply voltage (VDD) rise time</li> </ul> </li> </ul>
October 2009	6438	<ul style="list-style-type: none"> <li>■ The reset value for the <b>DID1</b> register may change, depending on the package.</li> <li>■ Deleted <b>MAXADCSPD</b> bit field from <b>DCCG0</b> register as it is not applicable in Deep-Sleep mode.</li> <li>■ Deleted reset value for 16-bit mode from <b>GPTMTAILR</b>, <b>GPTMTAMATCHR</b>, and <b>GPTMTAR</b> registers because the module resets in 32-bit mode.</li> <li>■ Made these changes to the Electrical Characteristics chapter: <ul style="list-style-type: none"> <li>– Removed <b>VSIH</b> and <b>VSIL</b> parameters from Operating Conditions table.</li> <li>– Changed SSI set up and hold times to be expressed in system clocks, not ns.</li> <li>– Revised ADC electrical specifications to clarify, including reorganizing and adding new data.</li> </ul> </li> <li>■ Added 48QFN package.</li> <li>■ Additional minor data sheet clarifications and corrections.</li> </ul>
July 2009	5953	<ul style="list-style-type: none"> <li>■ Clarified Power-on reset and <b>RST</b> pin operation; added new diagrams.</li> <li>■ Added <b>DBG</b> bits missing from <b>FMPRE</b> register. This changes register reset value.</li> <li>■ In ADC characteristics table, changed Max value for <b>GAIN</b> parameter from <math>\pm 1</math> to <math>\pm 3</math> and added <b>E<sub>IR</sub></b> (Internal voltage reference error) parameter.</li> <li>■ Corrected ordering numbers.</li> <li>■ Additional minor data sheet clarifications and corrections.</li> </ul>

**Table 1. Revision History (continued)**

Date	Revision	Description
April 2009	5369	<ul style="list-style-type: none"> <li>■ Added JTAG/SWD clarification (see "Communication with JTAG/SWD" on page 150).</li> <li>■ Added "GPIO Module DC Characteristics" table (see Table 19-4 on page 551).</li> <li>■ Additional minor data sheet clarifications and corrections.</li> </ul>
January 2009	4644	<ul style="list-style-type: none"> <li>■ Incorrect bit type for RELOAD bit field in SysTick Reload Value register; changed to R/W.</li> <li>■ Clarification added as to what happens when the SSI in slave mode is required to transmit but there is no data in the TX FIFO.</li> <li>■ Minor corrections to comparator operating mode tables.</li> <li>■ Additional minor data sheet clarifications and corrections.</li> </ul>
November 2008	4283	<ul style="list-style-type: none"> <li>■ Revised High-Level Block Diagram.</li> <li>■ Corrected descriptions for UART1 signals.</li> <li>■ Additional minor data sheet clarifications and corrections were made.</li> </ul>
October 2008	4149	<ul style="list-style-type: none"> <li>■ Added note on clearing interrupts to the Interrupts chapter:   <b>Note:</b> It may take several processor cycles after a write to clear an interrupt source in order for NVIC to see the interrupt source de-assert. This means if the interrupt clear is done as the last action in an interrupt handler, it is possible for the interrupt handler to complete while NVIC sees the interrupt as still asserted, causing the interrupt handler to be re-entered errantly. This can be avoided by either clearing the interrupt source at the beginning of the interrupt handler or by performing a read or write after the write to clear the interrupt source (and flush the write buffer)           </li> <li>■ Step 1 of the Initialization and Configuration procedure in the ADC chapter states the wrong register to use to enable the ADC clock. Sentence changed to:             1. Enable the ADC clock by writing a value of 0x0001.0000 to the <b>RCGC0</b> register.         </li> <li>■ Additional minor data sheet clarifications and corrections were made.</li> </ul>
June 2008	2972	Started tracking revision history.

# About This Document

This data sheet provides reference information for the LM3S811 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 related documents are available on the Stellaris® web site at [www.ti.com/stellaris](http://www.ti.com/stellaris):

- *Stellaris® Errata*
- *ARM® Cortex™-M3 Errata*
- *Cortex™-M3/M4 Instruction Set Technical User's Manual*
- *Stellaris® Graphics Library User's Guide*
- *Stellaris® Peripheral Driver Library User's Guide*

The following related documents are also referenced:

- *ARM® Debug Interface V5 Architecture Specification*
- *ARM® Embedded Trace Macrocell Architecture Specification*
- *IEEE Standard 1149.1-Test Access Port and Boundary-Scan Architecture*

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

## Documentation Conventions

This document uses the conventions shown in Table 2 on page 26.

**Table 2. 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 0xnnnn	A hexadecimal increment to a register's address, relative to that module's base address as specified in Table 2-4 on page 63.
Register N	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/WC	Software can read or write this field. Writing to it with any value clears the register.
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.

**Table 2. Documentation Conventions (continued)**

Notation	Meaning
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 SIGNAL and <u>SIGNAL</u> below).
deassert a signal	Change the value of the signal from the logically True state to the logically False state.
<u>SIGNAL</u>	Signal names are in uppercase and in the Courier font. An overbar on a signal name indicates that it is active Low. To assert <u>SIGNAL</u> is to drive it Low; to deassert <u>SIGNAL</u> is to drive it High.
SIGNAL	Signal names are in uppercase and in the Courier font. An active High signal has no overbar. To assert SIGNAL is to drive it High; to deassert SIGNAL 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.
0x	Hexadecimal numbers have a prefix of 0x. For example, 0x0FF 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 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 LM3S811 microcontroller is targeted for industrial applications, including test and measurement equipment, factory automation, HVAC and building control, motion control, medical instrumentation, fire and security, and power/energy.

In addition, the LM3S811 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 LM3S811 microcontroller is code-compatible to all members of the extensive Stellaris family; providing flexibility to fit our customers' precise needs.

Texas Instruments 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 588 for ordering information for Stellaris family devices.

## 1.1 Product Features

The LM3S811 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
  - 26 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
- ARM® Cortex™-M3 Processor Core

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

- Internal Memory
  - 64 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
  - 8 KB single-cycle SRAM
- GPIOs
  - 1-32 GPIOs, depending on configuration
  - 5-V-tolerant in input configuration
  - Fast toggle capable of a change every two clock cycles
  - Programmable control for GPIO interrupts
    - Interrupt generation masking
    - Edge-triggered on rising, falling, or both
    - Level-sensitive on High or Low values
  - 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
    - Slew rate control for the 8-mA drive
    - Open drain enables
    - Digital input enables
- General-Purpose Timers
  - Three General-Purpose Timer Modules (GPTM), each of which provides two 16-bit timers/counters. 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 when the controller asserts CPU Halt flag during debug
  - ADC event trigger
- 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
  - 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
- ADC
  - Four analog input channels
  - Single-ended and differential-input configurations

- On-chip internal temperature sensor
  - 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
  - Flexible trigger control
    - Controller (software)
    - Timers
    - Analog Comparators
    - PWM
    - GPIO
  - Hardware averaging of up to 64 samples for improved accuracy
  - Converter uses an internal 3-V reference
- **UART**
    - Two fully programmable 16C550-type UARTs
    - Separate 16x8 transmit (TX) and 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
    - 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
  - **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
- I<sup>2</sup>C
  - Devices on the I<sup>2</sup>C bus can be designated as either a master or a slave
    - Supports both sending and receiving data as either a master or a slave
    - Supports simultaneous master and slave operation
  - Four I<sup>2</sup>C modes
    - Master transmit
    - Master receive
    - Slave transmit
    - Slave receive
  - Two transmission speeds: Standard (100 Kbps) and Fast (400 Kbps)
  - Master and slave interrupt generation
    - Master generates interrupts when a transmit or receive operation completes (or aborts due to an error)
    - Slave generates interrupts when data has been sent or requested by a master
  - Master with arbitration and clock synchronization, multimaster support, and 7-bit addressing mode
- Analog Comparators
  - One integrated analog comparator
  - Configurable for output to drive an output pin, generate an interrupt, or initiate an ADC sample sequence
  - Compare external pin input to external pin input or to internal programmable voltage reference
  - Compare a test voltage against any one of these voltages
    - An individual external reference voltage
    - A shared single external reference voltage
    - A shared internal reference voltage
- PWM

- Three PWM generator blocks, each with one 16-bit counter, two PWM comparators, a PWM signal generator, a dead-band generator, and an interrupt/ADC-trigger selector
  - One fault input in hardware to promote low-latency shutdown
  - One 16-bit counter
    - Runs in Down or Up/Down mode
    - Output frequency controlled by a 16-bit load value
    - Load value updates can be synchronized
    - Produces output signals at zero and load value
  - Two PWM comparators
    - Comparator value updates can be synchronized
    - Produces output signals on match
  - PWM generator
    - Output PWM signal is constructed based on actions taken as a result of the counter and PWM comparator output signals
    - Produces two independent PWM signals
  - Dead-band generator
    - Produces two PWM signals with programmable dead-band delays suitable for driving a half-H bridge
    - Can be bypassed, leaving input PWM signals unmodified
  - Flexible output control block with PWM output enable of each PWM signal
    - PWM output enable of each PWM signal
    - Optional output inversion of each PWM signal (polarity control)
    - Optional fault handling for each PWM signal
    - Synchronization of timers in the PWM generator blocks
    - Interrupt status summary of the PWM generator blocks
  - Can initiate an ADC sample sequence
- Power
- On-chip Low Drop-Out (LDO) voltage regulator, with programmable output user-adjustable from 2.25 V to 2.75 V
  - Low-power options on controller: Sleep and Deep-sleep modes
  - Low-power options for peripherals: software controls shutdown of individual peripherals

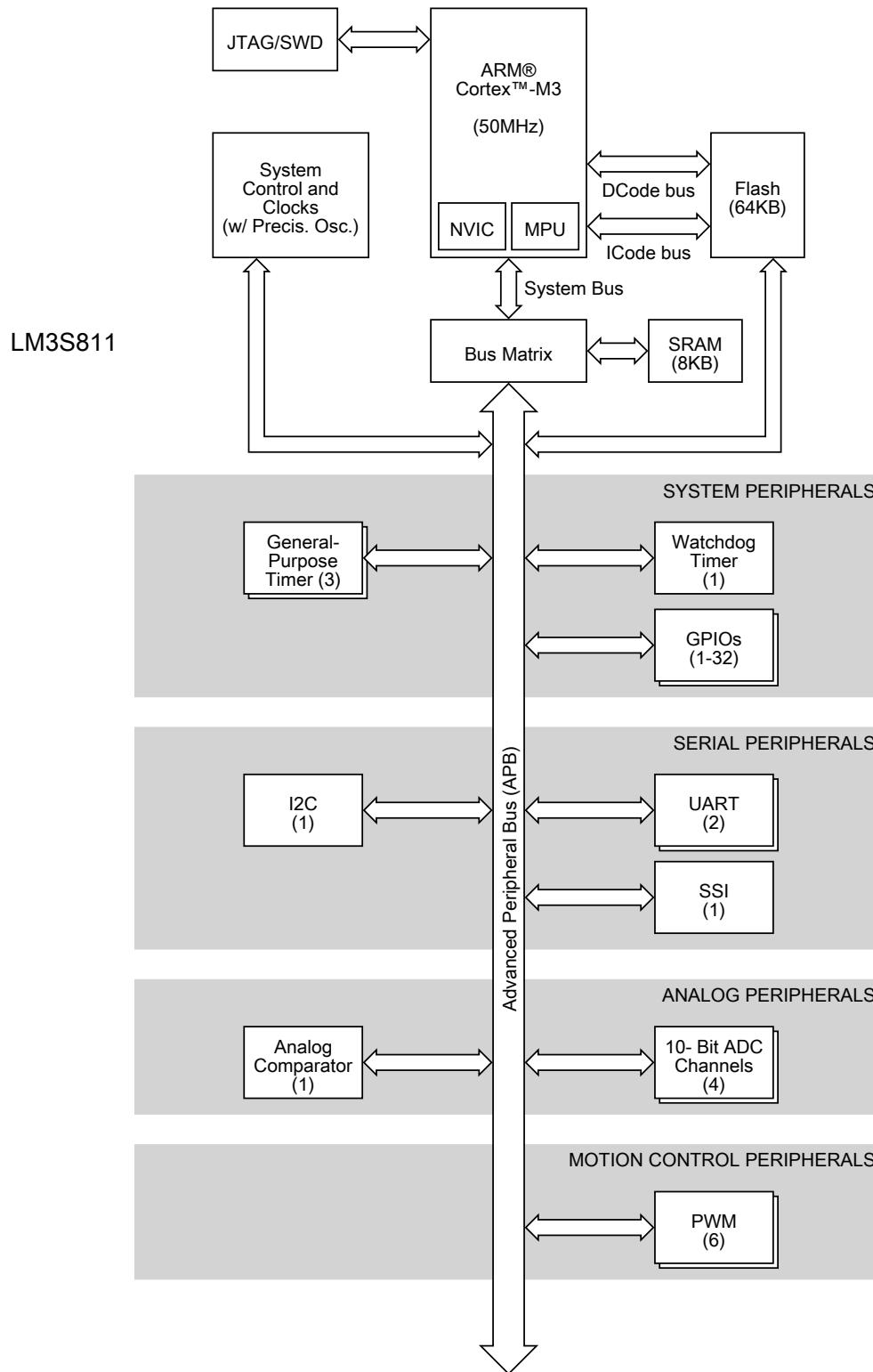
- User-enabled LDO unregulated voltage detection and automatic reset
- 3.3-V supply brown-out detection and reporting via interrupt or reset
- Flexible Reset Sources
  - Power-on reset (POR)
  - Reset pin assertion
  - Brown-out (BOR) detector alerts to system power drops
  - Software reset
  - Watchdog timer reset
  - Internal low drop-out (LDO) regulator output goes unregulated
- Industrial temperature 48-pin RoHS-compliant LQFP package
- Industrial temperature 48-pin RoHS-compliant QFN package

## 1.2 Target Applications

- Factory automation and control
- Industrial control power devices
- Building and home automation
- Stepper motors
- Brushless DC motors
- AC induction motors

## 1.3 High-Level Block Diagram

Figure 1-1 on page 36 depicts the features on the Stellaris LM3S811 microcontroller.

**Figure 1-1. Stellaris LM3S811 Microcontroller High-Level Block Diagram**

## 1.4 Functional Overview

The following sections provide an overview of the features of the LM3S811 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 588.

### 1.4.1 ARM Cortex™-M3

#### 1.4.1.1 Processor Core (see page 44)

All members of the Stellaris product family, including the LM3S811 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.

#### 1.4.1.2 Memory Map (see page 63)

A memory map lists the location of instructions and data in memory. The memory map for the LM3S811 controller can be found in Table 2-4 on page 63. Register addresses are given as a hexadecimal increment, relative to the module's base address as shown in the memory map.

#### 1.4.1.3 System Timer (SysTick) (see page 86)

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.

#### 1.4.1.4 Nested Vectored Interrupt Controller (NVIC) (see page 87)

The LM3S811 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 26 interrupts.

#### 1.4.1.5 System Control Block (SCB) (see page 89)

The SCB provides system implementation information and system control, including configuration, control, and reporting of system exceptions.

#### 1.4.1.6 Memory Protection Unit (MPU) (see page 89)

The MPU 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.

### 1.4.2 Motor Control Peripherals

To enhance motor control, the LM3S811 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 LM3S811, PWM motion control functionality can be achieved through:

- Dedicated, flexible motion control hardware using the PWM pins
- The motion control features of the general-purpose timers using the CCP pins

##### **PWM Pins (see page 500)**

The LM3S811 PWM module consists of three PWM generator blocks and a control block. Each PWM generator block contains one timer (16-bit down or up/down counter), two comparators, a PWM signal generator, a dead-band generator, and an interrupt/ADC-trigger selector. The control block determines the polarity of the PWM signals, and which signals are passed through to the pins.

Each PWM generator block produces two PWM signals that can either be independent signals or a single pair of complementary signals with dead-band delays inserted. The output of the PWM generation blocks are managed by the output control block before being passed to the device pins.

##### **CCP Pins (see page 282)**

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.

##### **Fault Pin (see page 506)**

The LM3S811 PWM module includes one fault-condition handling input to quickly provide low-latency shutdown and prevent damage to the motor being controlled.

### 1.4.3 Analog Peripherals

To handle analog signals, the LM3S811 microcontroller offers an Analog-to-Digital Converter (ADC).

For support of analog signals, the LM3S811 microcontroller offers one analog comparator.

#### 1.4.3.1 ADC (see page 336)

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

The LM3S811 ADC module features 10-bit conversion resolution and supports four 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.

#### **1.4.3.2 Analog Comparators (see page 488)**

An analog comparator is a peripheral that compares two analog voltages, and provides a logical output that signals the comparison result.

The LM3S811 microcontroller provides one analog comparator that can be configured to drive an output or generate an interrupt or ADC event.

A comparator can compare a test voltage against any one of these voltages:

- An individual external reference voltage
- A shared single external reference voltage
- A shared internal reference voltage

The comparator can provide its output to a device pin, acting as a replacement for an analog comparator on the board, or it can be used to signal the application via interrupts or triggers to the ADC to cause it to start capturing a sample sequence. The interrupt generation and ADC triggering logic is separate. This means, for example, that an interrupt can be generated on a rising edge and the ADC triggered on a falling edge.

### **1.4.4 Serial Communications Peripherals**

The LM3S811 controller supports both asynchronous and synchronous serial communications with:

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

#### **1.4.4.1 UART (see page 373)**

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 LM3S811 controller includes two 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.)

Separate 16x8 transmit (TX) and 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 413)**

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

The LM3S811 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 451)

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 LM3S811 controller includes one I<sup>2</sup>C module that provides 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. The 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 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 232)

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

The Stellaris 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 1-32 programmable input/output pins. The number of GPIOs available depends on the peripherals being used (see "Signal Tables" on page 541 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 Three Programmable Timers (see page 275)

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

The Stellaris General-Purpose Timer Module (GPTM) contains three 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 312)

A watchdog timer can generate an interrupt 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 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 LM3S811 controller offers both single-cycle SRAM and single-cycle Flash memory.

### 1.4.6.1 SRAM (see page 214)

The LM3S811 static random access memory (SRAM) controller supports 8 KB SRAM. The internal SRAM of the Stellaris devices starts at base 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 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 215)

The LM3S811 Flash controller supports 64 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.

## 1.4.7 Additional Features

### 1.4.7.1 JTAG TAP Controller (see page 144)

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 five pins: `TRST`, `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 Stellaris 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 Stellaris JTAG instructions select the Stellaris `TDO` outputs. The multiplexer is controlled by the Stellaris JTAG controller, which has comprehensive programming for the ARM, Stellaris, and unimplemented JTAG instructions.

#### 1.4.7.2 System Control and Clocks (see page 155)

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.

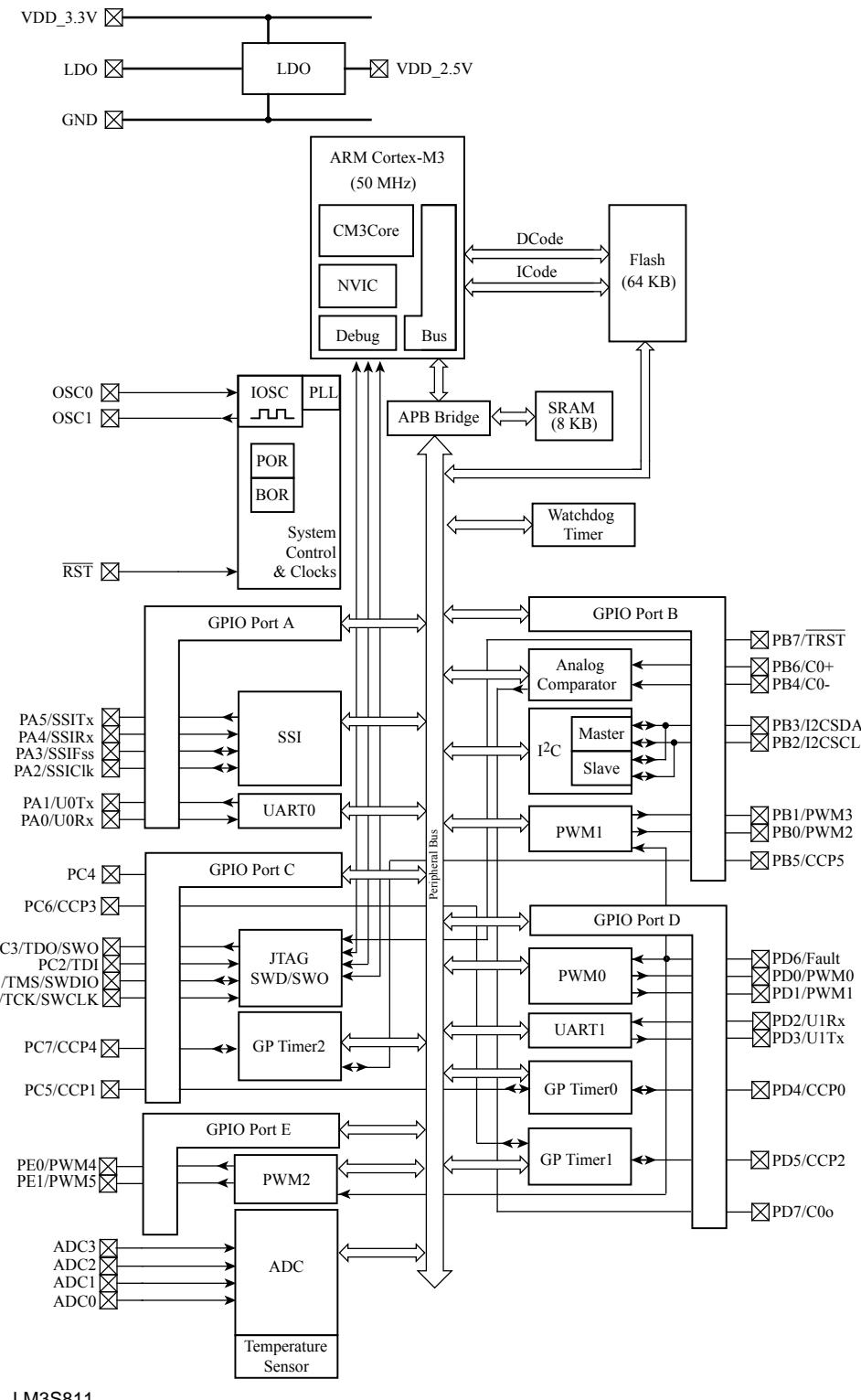
#### 1.4.8 Hardware Details

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

- “Pin Diagram” on page 539
- “Signal Tables” on page 541
- “Operating Characteristics” on page 549
- “Electrical Characteristics” on page 550
- “Package Information” on page 590

### 1.4.9 System Block Diagram

**Figure 1-2. LM3S811 Controller System-Level Block Diagram**



LM3S811

## 2 The Cortex-M3 Processor

The ARM® Cortex™-M3 processor provides a high-performance, low-cost platform that meets the system requirements 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
- 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
  - 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® 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 motor control.

This chapter provides information on the Stellaris implementation of the Cortex-M3 processor, including the programming model, the memory model, the exception model, fault handling, and power management.

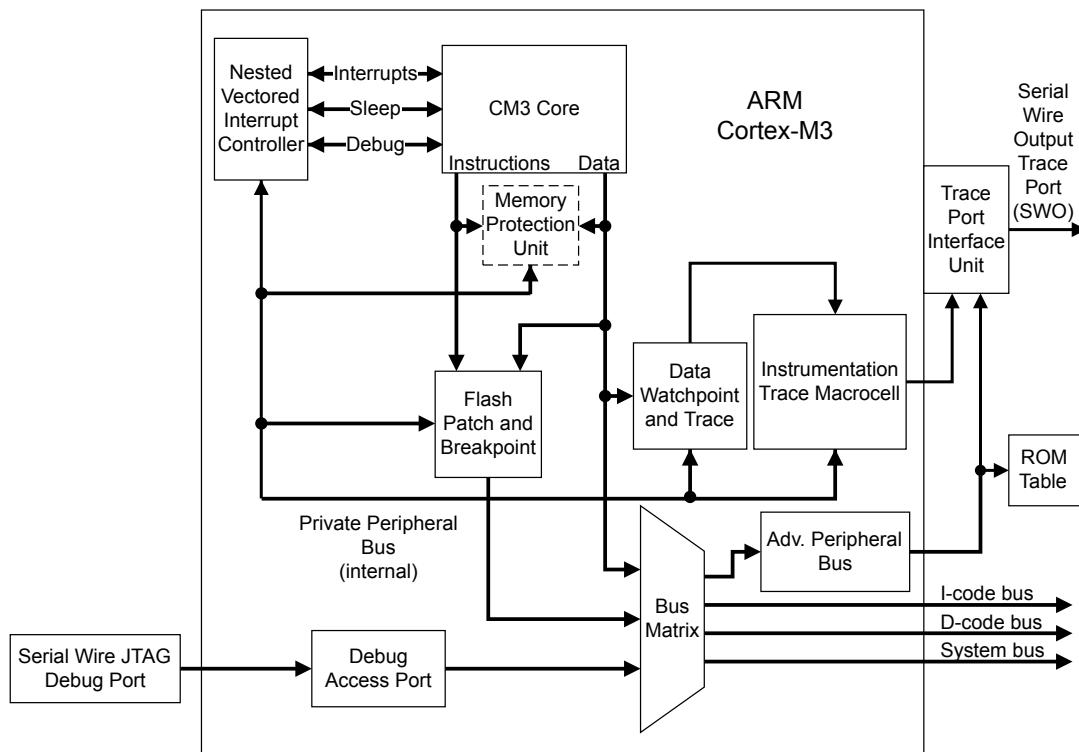
For technical details on the instruction set, see the *Cortex™-M3/M4 Instruction Set Technical User's Manual*.

## 2.1 Block Diagram

The Cortex-M3 processor is built on a high-performance processor core, with a 3-stage pipeline Harvard architecture, making it ideal for demanding embedded applications. The processor delivers exceptional power efficiency through an efficient instruction set and extensively optimized design, providing high-end processing hardware including a range of single-cycle and SIMD multiplication and multiply-with-accumulate capabilities, saturating arithmetic and dedicated hardware division.

To facilitate the design of cost-sensitive devices, the Cortex-M3 processor implements tightly coupled system components that reduce processor area while significantly improving interrupt handling and system debug capabilities. The Cortex-M3 processor implements a version of the Thumb® instruction set based on Thumb-2 technology, ensuring high code density and reduced program memory requirements. The Cortex-M3 instruction set provides the exceptional performance expected of a modern 32-bit architecture, with the high code density of 8-bit and 16-bit microcontrollers.

The Cortex-M3 processor closely integrates a nested interrupt controller (NVIC), to deliver industry-leading interrupt performance. The Stellaris NVIC includes a non-maskable interrupt (NMI) and provides eight interrupt priority levels. The tight integration of the processor core and NVIC provides fast execution of interrupt service routines (ISRs), dramatically reducing interrupt latency. The hardware stacking of registers and the ability to suspend load-multiple and store-multiple operations further reduce interrupt latency. Interrupt handlers do not require any assembler stubs which removes code overhead from the ISRs. Tail-chaining optimization also significantly reduces the overhead when switching from one ISR to another. To optimize low-power designs, the NVIC integrates with the sleep modes, including Deep-sleep mode, which enables the entire device to be rapidly powered down.

**Figure 2-1. CPU Block Diagram**

## 2.2 Overview

### 2.2.1 System-Level Interface

The Cortex-M3 processor provides multiple interfaces using AMBA® technology to provide high-speed, low-latency memory accesses. The core supports unaligned data accesses and implements atomic bit manipulation that enables faster peripheral controls, system spinlocks, and thread-safe Boolean data handling.

The Cortex-M3 processor has a memory protection unit (MPU) that provides fine-grain memory control, enabling applications to implement security privilege levels and separate code, data and stack on a task-by-task basis.

### 2.2.2 Integrated Configurable Debug

The Cortex-M3 processor implements a complete hardware debug solution, providing high system visibility of the processor and memory through either a traditional JTAG port or a 2-pin Serial Wire Debug (SWD) port that is ideal for microcontrollers and other small package devices. The Stellaris implementation replaces the ARM SW-DP and JTAG-DP with the ARM CoreSight™-compliant Serial Wire JTAG Debug Port (SWJ-DP) interface. The SWJ-DP interface combines the SWD and JTAG debug ports into one module. See the *ARM® Debug Interface V5 Architecture Specification* for details on SWJ-DP.

For system trace, the processor integrates an Instrumentation Trace Macrocell (ITM) alongside data watchpoints and a profiling unit. To enable simple and cost-effective profiling of the system trace events, a Serial Wire Viewer (SWV) can export a stream of software-generated messages, data trace, and profiling information through a single pin.

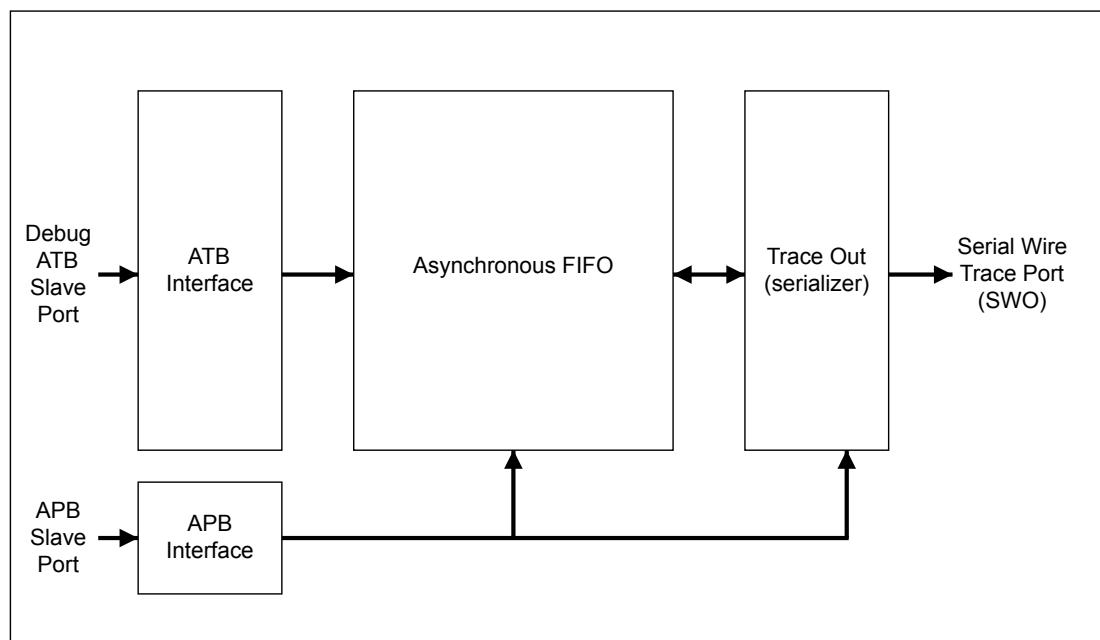
The Flash Patch and Breakpoint Unit (FPB) provides up to eight hardware breakpoint comparators that debuggers can use. The comparators in the FPB also provide remap functions of up to eight words in the program code in the CODE memory region. This enables applications stored in a read-only area of Flash memory to be patched in another area of on-chip SRAM or Flash memory. If a patch is required, the application programs the FPB to remap a number of addresses. When those addresses are accessed, the accesses are redirected to a remap table specified in the FPB configuration.

For more information on the Cortex-M3 debug capabilities, see the *ARM® Debug Interface V5 Architecture Specification*.

### 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, as shown in Figure 2-2 on page 47.

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



### 2.2.4 Cortex-M3 System Component Details

The Cortex-M3 includes the following system components:

- **SysTick**  
A 24-bit count-down timer that can be used as a Real-Time Operating System (RTOS) tick timer or as a simple counter (see “System Timer (SysTick)” on page 86).
- **Nested Vectored Interrupt Controller (NVIC)**  
An embedded interrupt controller that supports low latency interrupt processing (see “Nested Vectored Interrupt Controller (NVIC)” on page 87).
- **System Control Block (SCB)**

The programming model interface to the processor. The SCB provides system implementation information and system control, including configuration, control, and reporting of system exceptions (see “System Control Block (SCB)” on page 89).

- Memory Protection Unit (MPU)

Improves system reliability by defining the memory attributes for different memory regions. The MPU provides up to eight different regions and an optional predefined background region (see “Memory Protection Unit (MPU)” on page 89).

## 2.3 Programming Model

This section describes the Cortex-M3 programming model. In addition to the individual core register descriptions, information about the processor modes and privilege levels for software execution and stacks is included.

### 2.3.1 Processor Mode and Privilege Levels for Software Execution

The Cortex-M3 has two modes of operation:

- Thread mode

Used to execute application software. The processor enters Thread mode when it comes out of reset.

- Handler mode

Used to handle exceptions. When the processor has finished exception processing, it returns to Thread mode.

In addition, the Cortex-M3 has two privilege levels:

- Unprivileged

In this mode, software has the following restrictions:

- Limited access to the `MSR` and `MRS` instructions and no use of the `CPS` instruction
- No access to the system timer, NVIC, or system control block
- Possibly restricted access to memory or peripherals

- Privileged

In this mode, software can use all the instructions and has access to all resources.

In Thread mode, the **CONTROL** register (see page 62) controls whether software execution is privileged or unprivileged. In Handler mode, software execution is always privileged.

Only privileged software can write to the **CONTROL** register to change the privilege level for software execution in Thread mode. Unprivileged software can use the `SVC` instruction to make a supervisor call to transfer control to privileged software.

### 2.3.2 Stacks

The processor uses a full descending stack, meaning that the stack pointer indicates the last stacked item on the memory. When the processor pushes a new item onto the stack, it decrements the stack pointer and then writes the item to the new memory location. The processor implements two stacks:

the main stack and the process stack, with a pointer for each held in independent registers (see the **SP** register on page 52).

In Thread mode, the **CONTROL** register (see page 62) controls whether the processor uses the main stack or the process stack. In Handler mode, the processor always uses the main stack. The options for processor operations are shown in Table 2-1 on page 49.

**Table 2-1. Summary of Processor Mode, Privilege Level, and Stack Use**

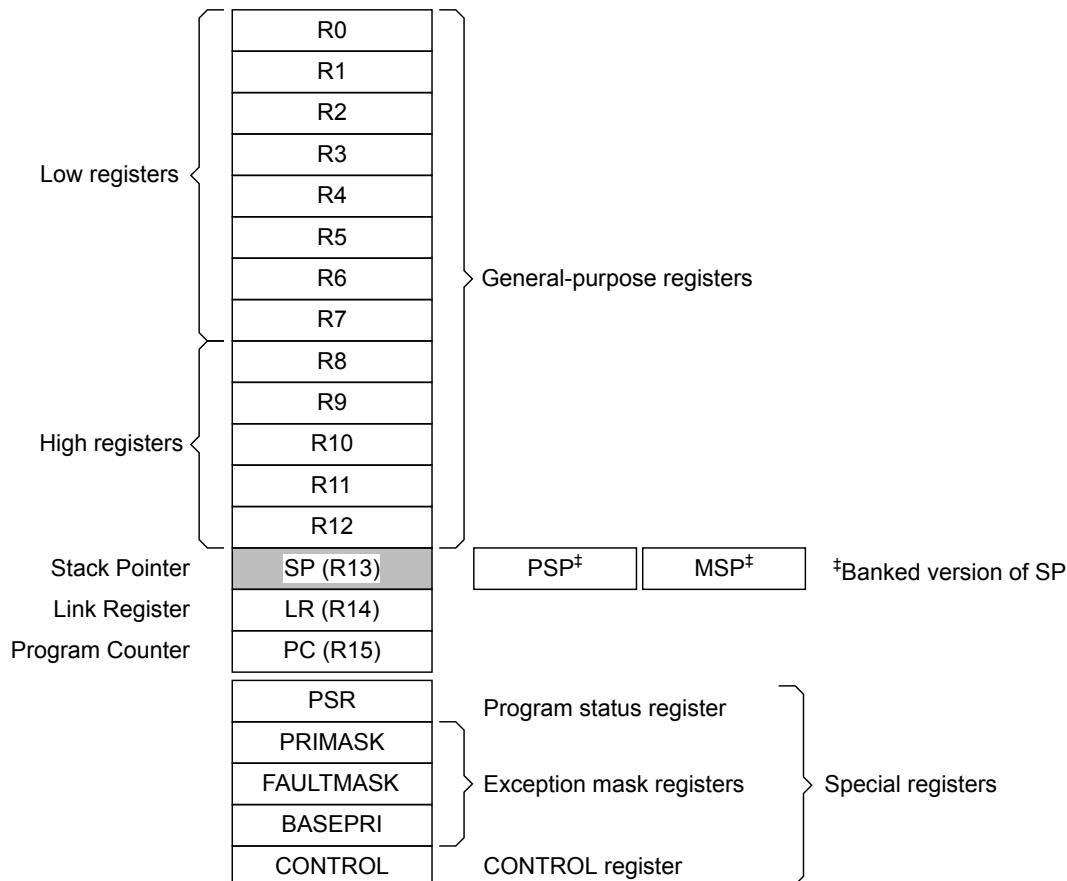
Processor Mode	Use	Privilege Level	Stack Used
Thread	Applications	Privileged or unprivileged <sup>a</sup>	Main stack or process stack <sup>a</sup>
Handler	Exception handlers	Always privileged	Main stack

a. See **CONTROL** (page 62).

### 2.3.3 Register Map

Figure 2-3 on page 49 shows the Cortex-M3 register set. Table 2-2 on page 50 lists the Core registers. The core registers are not memory mapped and are accessed by register name, so the base address is n/a (not applicable) and there is no offset.

**Figure 2-3. Cortex-M3 Register Set**



**Table 2-2. Processor Register Map**

Offset	Name	Type	Reset	Description	See page
-	R0	R/W	-	Cortex General-Purpose Register 0	51
-	R1	R/W	-	Cortex General-Purpose Register 1	51
-	R2	R/W	-	Cortex General-Purpose Register 2	51
-	R3	R/W	-	Cortex General-Purpose Register 3	51
-	R4	R/W	-	Cortex General-Purpose Register 4	51
-	R5	R/W	-	Cortex General-Purpose Register 5	51
-	R6	R/W	-	Cortex General-Purpose Register 6	51
-	R7	R/W	-	Cortex General-Purpose Register 7	51
-	R8	R/W	-	Cortex General-Purpose Register 8	51
-	R9	R/W	-	Cortex General-Purpose Register 9	51
-	R10	R/W	-	Cortex General-Purpose Register 10	51
-	R11	R/W	-	Cortex General-Purpose Register 11	51
-	R12	R/W	-	Cortex General-Purpose Register 12	51
-	SP	R/W	-	Stack Pointer	52
-	LR	R/W	0xFFFF.FFFF	Link Register	53
-	PC	R/W	-	Program Counter	54
-	PSR	R/W	0x0100.0000	Program Status Register	55
-	PRIMASK	R/W	0x0000.0000	Priority Mask Register	59
-	FAULTMASK	R/W	0x0000.0000	Fault Mask Register	60
-	BASEPRI	R/W	0x0000.0000	Base Priority Mask Register	61
-	CONTROL	R/W	0x0000.0000	Control Register	62

### 2.3.4 Register Descriptions

This section lists and describes the Cortex-M3 registers, in the order shown in Figure 2-3 on page 49. The core registers are not memory mapped and are accessed by register name rather than offset.

**Note:** The register type shown in the register descriptions refers to type during program execution in Thread mode and Handler mode. Debug access can differ.

- Register 1: Cortex General-Purpose Register 0 (R0)**
- Register 2: Cortex General-Purpose Register 1 (R1)**
- Register 3: Cortex General-Purpose Register 2 (R2)**
- Register 4: Cortex General-Purpose Register 3 (R3)**
- Register 5: Cortex General-Purpose Register 4 (R4)**
- Register 6: Cortex General-Purpose Register 5 (R5)**
- Register 7: Cortex General-Purpose Register 6 (R6)**
- Register 8: Cortex General-Purpose Register 7 (R7)**
- Register 9: Cortex General-Purpose Register 8 (R8)**
- Register 10: Cortex General-Purpose Register 9 (R9)**
- Register 11: Cortex General-Purpose Register 10 (R10)**
- Register 12: Cortex General-Purpose Register 11 (R11)**
- Register 13: Cortex General-Purpose Register 12 (R12)**

The **Rn** registers are 32-bit general-purpose registers for data operations and can be accessed from either privileged or unprivileged mode.

#### Cortex General-Purpose Register 0 (R0)

Type R/W, reset -

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

Bit/Field	Name	Type	Reset	Description
31:0	DATA	R/W	-	Register data.

## Register 14: Stack Pointer (SP)

The **Stack Pointer (SP)** is register R13. In Thread mode, the function of this register changes depending on the **ASP** bit in the **Control Register (CONTROL)** register. When the **ASP** bit is clear, this register is the **Main Stack Pointer (MSP)**. When the **ASP** bit is set, this register is the **Process Stack Pointer (PSP)**. On reset, the **ASP** bit is clear, and the processor loads the **MSP** with the value from address 0x0000.0000. The **MSP** can only be accessed in privileged mode; the **PSP** can be accessed in either privileged or unprivileged mode.

### Stack Pointer (SP)

Type R/W, reset -

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

Bit/Field	Name	Type	Reset	Description
31:0	SP	R/W	-	This field is the address of the stack pointer.

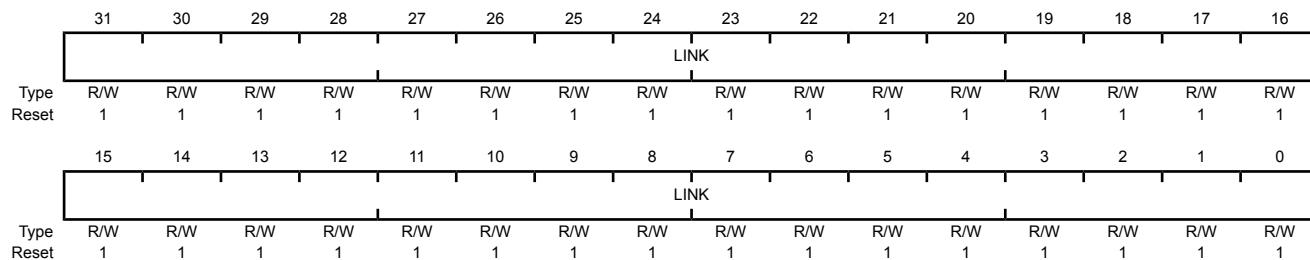
## Register 15: Link Register (LR)

The **Link Register (LR)** is register R14, and it stores the return information for subroutines, function calls, and exceptions. **LR** can be accessed from either privileged or unprivileged mode.

`EXC_RETURN` is loaded into **LR** on exception entry. See Table 2-10 on page 78 for the values and description.

### Link Register (LR)

Type R/W, reset 0xFFFF.FFFF



Bit/Field                  Name                  Type                  Reset                  Description

31:0                  LINK                  R/W                  0xFFFF.FFFF          This field is the return address.

## Register 16: Program Counter (PC)

The **Program Counter (PC)** is register R15, and it contains the current program address. On reset, the processor loads the **PC** with the value of the reset vector, which is at address 0x0000.0004. Bit 0 of the reset vector is loaded into the **THUMB** bit of the **EPSR** at reset and must be 1. The **PC** register can be accessed in either privileged or unprivileged mode.

### Program Counter (PC)

Type R/W, reset -

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

Bit/Field	Name	Type	Reset	Description
-----------	------	------	-------	-------------

31:0	PC	R/W	-	This field is the current program address.
------	----	-----	---	--

## Register 17: Program Status Register (PSR)

**Note:** This register is also referred to as **xPSR**.

The **Program Status Register (PSR)** has three functions, and the register bits are assigned to the different functions:

- **Application Program Status Register (APSR)**, bits 31:27,
- **Execution Program Status Register (EPSR)**, bits 26:24, 15:10
- **Interrupt Program Status Register (IPSR)**, bits 5:0

The **PSR**, **IPSR**, and **EPSR** registers can only be accessed in privileged mode; the **APSR** register can be accessed in either privileged or unprivileged mode.

**APSR** contains the current state of the condition flags from previous instruction executions.

**EPSR** contains the Thumb state bit and the execution state bits for the If-Then (**IT**) instruction or the Interruptible-Continuable Instruction (**ICI**) field for an interrupted load multiple or store multiple instruction. Attempts to read the **EPSR** directly through application software using the **MSR** instruction always return zero. Attempts to write the **EPSR** using the **MSR** instruction in application software are always ignored. Fault handlers can examine the **EPSR** value in the stacked **PSR** to determine the operation that faulted (see “Exception Entry and Return” on page 76).

**IPSR** contains the exception type number of the current Interrupt Service Routine (ISR).

These registers can be accessed individually or as a combination of any two or all three registers, using the register name as an argument to the **MRS** or **MSR** instructions. For example, all of the registers can be read using **PSR** with the **MRS** instruction, or **APSR** only can be written to using **APSR** with the **MSR** instruction. page 55 shows the possible register combinations for the **PSR**. See the **MRS** and **MSR** instruction descriptions in the *Cortex™-M3/M4 Instruction Set Technical User’s Manual* for more information about how to access the program status registers.

**Table 2-3. PSR Register Combinations**

Register	Type	Combination
<b>PSR</b>	R/W <sup>a, b</sup>	<b>APSR</b> , <b>EPSR</b> , and <b>IPSR</b>
<b>IEPSR</b>	RO	<b>EPSR</b> and <b>IPSR</b>
<b>IAPSR</b>	R/W <sup>a</sup>	<b>APSR</b> and <b>IPSR</b>
<b>EAPSR</b>	R/W <sup>b</sup>	<b>APSR</b> and <b>EPSR</b>

a. The processor ignores writes to the **IPSR** bits.

b. Reads of the **EPSR** bits return zero, and the processor ignores writes to these bits.

### Program Status Register (PSR)

Type R/W, reset 0x0100.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
N	Z	C	V	Q	ICI / IT	THUMB	reserved								
Type	R/W	R/W	R/W	R/W	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					ICI / IT	reserved									ISRNUM
Type	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

Bit/Field	Name	Type	Reset	Description
31	N	R/W	0	<p><b>APSR</b> Negative or Less Flag</p> <p>Value Description</p> <p>1 The previous operation result was negative or less than.</p> <p>0 The previous operation result was positive, zero, greater than, or equal.</p> <p>The value of this bit is only meaningful when accessing <b>PSR</b> or <b>APSR</b>.</p>
30	Z	R/W	0	<p><b>APSR</b> Zero Flag</p> <p>Value Description</p> <p>1 The previous operation result was zero.</p> <p>0 The previous operation result was non-zero.</p> <p>The value of this bit is only meaningful when accessing <b>PSR</b> or <b>APSR</b>.</p>
29	C	R/W	0	<p><b>APSR</b> Carry or Borrow Flag</p> <p>Value Description</p> <p>1 The previous add operation resulted in a carry bit or the previous subtract operation did not result in a borrow bit.</p> <p>0 The previous add operation did not result in a carry bit or the previous subtract operation resulted in a borrow bit.</p> <p>The value of this bit is only meaningful when accessing <b>PSR</b> or <b>APSR</b>.</p>
28	V	R/W	0	<p><b>APSR</b> Overflow Flag</p> <p>Value Description</p> <p>1 The previous operation resulted in an overflow.</p> <p>0 The previous operation did not result in an overflow.</p> <p>The value of this bit is only meaningful when accessing <b>PSR</b> or <b>APSR</b>.</p>
27	Q	R/W	0	<p><b>APSR</b> DSP Overflow and Saturation Flag</p> <p>Value Description</p> <p>1 DSP Overflow or saturation has occurred.</p> <p>0 DSP overflow or saturation has not occurred since reset or since the bit was last cleared.</p> <p>The value of this bit is only meaningful when accessing <b>PSR</b> or <b>APSR</b>. This bit is cleared by software using an <b>MRS</b> instruction.</p>

Bit/Field	Name	Type	Reset	Description
26:25	ICI / IT	RO	0x0	<p><b>EPSR ICI / IT status</b></p> <p>These bits, along with bits 15:10, contain the Interruptible-Continuable Instruction (<b>ICI</b>) field for an interrupted load multiple or store multiple instruction or the execution state bits of the <b>IT</b> instruction.</p> <p>When <b>EPSR</b> holds the <b>ICI</b> execution state, bits 26:25 are zero.</p> <p>The If-Then block contains up to four instructions following an <b>IT</b> instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See the <i>Cortex™-M3/M4 Instruction Set Technical User's Manual</i> for more information.</p> <p>The value of this field is only meaningful when accessing <b>PSR</b> or <b>EPSR</b>.</p>
24	THUMB	RO	1	<p><b>EPSR Thumb State</b></p> <p>This bit indicates the Thumb state and should always be set.</p> <p>The following can clear the <b>THUMB</b> bit:</p> <ul style="list-style-type: none"> <li>■ The <b>BLX</b>, <b>BX</b> and <b>POP{PC}</b> instructions</li> <li>■ Restoration from the stacked <b>xPSR</b> value on an exception return</li> <li>■ Bit 0 of the vector value on an exception entry or reset</li> </ul> <p>Attempting to execute instructions when this bit is clear results in a fault or lockup. See "Lockup" on page 80 for more information.</p> <p>The value of this bit is only meaningful when accessing <b>PSR</b> or <b>EPSR</b>.</p>
23:16	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.
15:10	ICI / IT	RO	0x0	<p><b>EPSR ICI / IT status</b></p> <p>These bits, along with bits 26:25, contain the Interruptible-Continuable Instruction (<b>ICI</b>) field for an interrupted load multiple or store multiple instruction or the execution state bits of the <b>IT</b> instruction.</p> <p>When an interrupt occurs during the execution of an <b>LDM</b>, <b>STM</b>, <b>PUSH</b> or <b>POP</b> instruction, the processor stops the load multiple or store multiple instruction operation temporarily and stores the next register operand in the multiple operation to bits 15:12. After servicing the interrupt, the processor returns to the register pointed to by bits 15:12 and resumes execution of the multiple load or store instruction. When <b>EPSR</b> holds the <b>ICI</b> execution state, bits 11:10 are zero.</p> <p>The If-Then block contains up to four instructions following a 16-bit <b>IT</b> instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See the <i>Cortex™-M3/M4 Instruction Set Technical User's Manual</i> for more information.</p> <p>The value of this field is only meaningful when accessing <b>PSR</b> or <b>EPSR</b>.</p>
9:6	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.

Bit/Field	Name	Type	Reset	Description
5:0	ISRNUM	RO	0x00	<b>IPSR</b> ISR Number
				This field contains the exception type number of the current Interrupt Service Routine (ISR).
				Value      Description
				0x00      Thread mode
				0x01      Reserved
				0x02      NMI
				0x03      Hard fault
				0x04      Memory management fault
				0x05      Bus fault
				0x06      Usage fault
				0x07-0x0A Reserved
				0x0B      SVCall
				0x0C      Reserved for Debug
				0x0D      Reserved
				0x0E      PendSV
				0x0F      SysTick
				0x10      Interrupt Vector 0
				0x11      Interrupt Vector 1
				...      ...
				0x2D      Interrupt Vector 29
				0x2E-0x3F Reserved

See “Exception Types” on page 72 for more information.

The value of this field is only meaningful when accessing **PSR** or **IPSR**.

## Register 18: Priority Mask Register (PRIMASK)

The **PRIMASK** register prevents activation of all exceptions with programmable priority. Reset, non-maskable interrupt (NMI), and hard fault are the only exceptions with fixed priority. Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. The MSR and MRS instructions are used to access the **PRIMASK** register, and the CPS instruction may be used to change the value of the **PRIMASK** register. See the *Cortex™-M3/M4 Instruction Set Technical User's Manual* for more information on these instructions. For more information on exception priority levels, see “Exception Types” on page 72.

### Priority Mask Register (PRIMASK)

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															PRIMASK
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

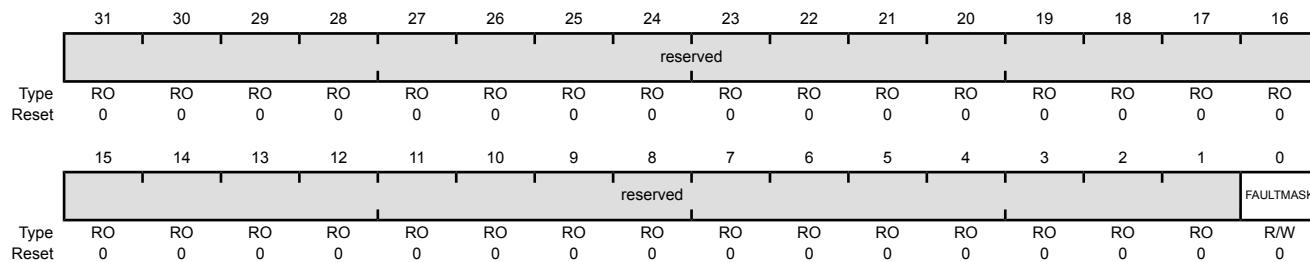
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	PRIMASK	R/W	0	Priority Mask
		Value	Description	
		1	Prevents the activation of all exceptions with configurable priority.	
		0	No effect.	

## Register 19: Fault Mask Register (FAULTMASK)

The **FAULTMASK** register prevents activation of all exceptions except for the Non-Maskable Interrupt (NMI). Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. The **MSR** and **MRS** instructions are used to access the **FAULTMASK** register, and the **CPS** instruction may be used to change the value of the **FAULTMASK** register. See the *Cortex™-M3/M4 Instruction Set Technical User's Manual* for more information on these instructions. For more information on exception priority levels, see "Exception Types" on page 72.

### Fault Mask Register (FAULTMASK)

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.0000	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.
0	FAULTMASK	R/W	0	Fault Mask
		Value	Description	
		1	Prevents the activation of all exceptions except for NMI.	
		0	No effect.	

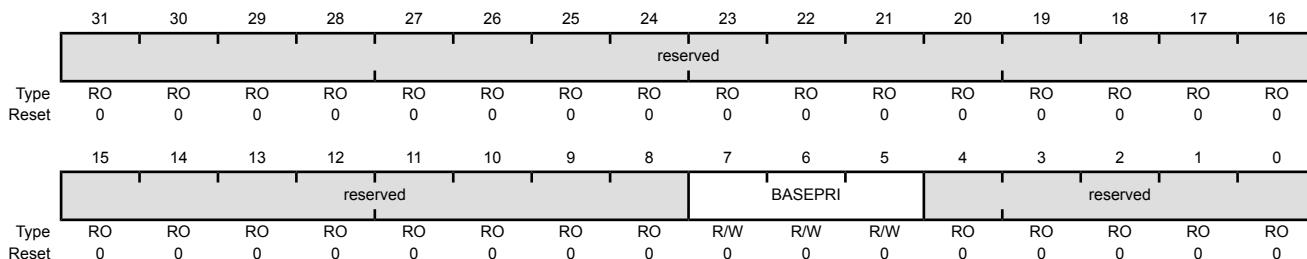
The processor clears the **FAULTMASK** bit on exit from any exception handler except the NMI handler.

## Register 20: Base Priority Mask Register (BASEPRI)

The **BASEPRI** register defines the minimum priority for exception processing. When **BASEPRI** is set to a nonzero value, it prevents the activation of all exceptions with the same or lower priority level as the **BASEPRI** value. Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. For more information on exception priority levels, see “Exception Types” on page 72.

### Base Priority Mask Register (BASEPRI)

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description																		
31:8	reserved	RO	0x0000.00	Software should not rely on the value of 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	BASEPRI	R/W	0x0	<p>Base Priority</p> <p>Any exception that has a programmable priority level with the same or lower priority as the value of this field is masked. The <b>PRIMASK</b> register can be used to mask all exceptions with programmable priority levels. Higher priority exceptions have lower priority levels.</p> <table border="0"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>All exceptions are unmasked.</td> </tr> <tr> <td>0x1</td> <td>All exceptions with priority level 1-7 are masked.</td> </tr> <tr> <td>0x2</td> <td>All exceptions with priority level 2-7 are masked.</td> </tr> <tr> <td>0x3</td> <td>All exceptions with priority level 3-7 are masked.</td> </tr> <tr> <td>0x4</td> <td>All exceptions with priority level 4-7 are masked.</td> </tr> <tr> <td>0x5</td> <td>All exceptions with priority level 5-7 are masked.</td> </tr> <tr> <td>0x6</td> <td>All exceptions with priority level 6-7 are masked.</td> </tr> <tr> <td>0x7</td> <td>All exceptions with priority level 7 are masked.</td> </tr> </tbody> </table>	Value	Description	0x0	All exceptions are unmasked.	0x1	All exceptions with priority level 1-7 are masked.	0x2	All exceptions with priority level 2-7 are masked.	0x3	All exceptions with priority level 3-7 are masked.	0x4	All exceptions with priority level 4-7 are masked.	0x5	All exceptions with priority level 5-7 are masked.	0x6	All exceptions with priority level 6-7 are masked.	0x7	All exceptions with priority level 7 are masked.
Value	Description																					
0x0	All exceptions are unmasked.																					
0x1	All exceptions with priority level 1-7 are masked.																					
0x2	All exceptions with priority level 2-7 are masked.																					
0x3	All exceptions with priority level 3-7 are masked.																					
0x4	All exceptions with priority level 4-7 are masked.																					
0x5	All exceptions with priority level 5-7 are masked.																					
0x6	All exceptions with priority level 6-7 are masked.																					
0x7	All exceptions with priority level 7 are masked.																					
4:0	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 21: Control Register (CONTROL)

The **CONTROL** register controls the stack used and the privilege level for software execution when the processor is in Thread mode. This register is only accessible in privileged mode.

Handler mode always uses **MSP**, so the processor ignores explicit writes to the **ASP** bit of the **CONTROL** register when in Handler mode. The exception entry and return mechanisms automatically update the **CONTROL** register based on the **EXC\_RETURN** value (see Table 2-10 on page 78). In an OS environment, threads running in Thread mode should use the process stack and the kernel and exception handlers should use the main stack. By default, Thread mode uses **MSP**. To switch the stack pointer used in Thread mode to **PSP**, either use the **MSR** instruction to set the **ASP** bit, as detailed in the *Cortex™-M3/M4 Instruction Set Technical User's Manual*, or perform an exception return to Thread mode with the appropriate **EXC\_RETURN** value, as shown in Table 2-10 on page 78.

**Note:** When changing the stack pointer, software must use an **ISB** instruction immediately after the **MSR** instruction, ensuring that instructions after the **ISB** execute use the new stack pointer. See the *Cortex™-M3/M4 Instruction Set Technical User's Manual*.

### Control Register (CONTROL)

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														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	RO	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
31:2	reserved	RO	0x0000.0000	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.
1	ASP	R/W	0	<p>Active Stack Pointer</p> <p>Value Description</p> <p>1     <b>PSP</b> is the current stack pointer.</p> <p>0     <b>MSP</b> is the current stack pointer</p> <p>In Handler mode, this bit reads as zero and ignores writes. The Cortex-M3 updates this bit automatically on exception return.</p>
0	TMPL	R/W	0	<p>Thread Mode Privilege Level</p> <p>Value Description</p> <p>1     Unprivileged software can be executed in Thread mode.</p> <p>0     Only privileged software can be executed in Thread mode.</p>

### 2.3.5 Exceptions and Interrupts

The Cortex-M3 processor supports interrupts and system exceptions. The processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions. An exception changes the normal flow of software control. The processor uses Handler mode to handle all exceptions except for reset. See “Exception Entry and Return” on page 76 for more information.

The NVIC registers control interrupt handling. See “Nested Vectored Interrupt Controller (NVIC)” on page 87 for more information.

### 2.3.6 Data Types

The Cortex-M3 supports 32-bit words, 16-bit halfwords, and 8-bit bytes. The processor also supports 64-bit data transfer instructions. All instruction and data memory accesses are little endian. See “Memory Regions, Types and Attributes” on page 64 for more information.

## 2.4 Memory Model

This section describes the processor memory map, the behavior of memory accesses, and the bit-banding features. The processor has a fixed memory map that provides up to 4 GB of addressable memory.

The memory map for the LM3S811 controller is provided in Table 2-4 on page 63. In this manual, register addresses are given as a hexadecimal increment, relative to the module’s base address as shown in the memory map.

The regions for SRAM and peripherals include bit-band regions. Bit-banding provides atomic operations to bit data (see “Bit-Banding” on page 67).

The processor reserves regions of the Private peripheral bus (PPB) address range for core peripheral registers (see “Cortex-M3 Peripherals” on page 86).

**Note:** Within the memory map, all reserved space returns a bus fault when read or written.

**Table 2-4. Memory Map**

Start	End	Description	For details, see page ...
<b>Memory</b>			
0x0000.0000	0x0000.FFFF	On-chip Flash	220
0x0001.0000	0x1FFF.FFFF	Reserved	-
0x2000.0000	0x2000.1FFF	Bit-banded on-chip SRAM	214
0x2000.2000	0x21FF.FFFF	Reserved	-
0x2200.0000	0x2203.FFFF	Bit-band alias of bit-banded on-chip SRAM starting at 0x2000.0000	214
0x2204.0000	0x3FFF.FFFF	Reserved	-
<b>FiRM Peripherals</b>			
0x4000.0000	0x4000.0FFF	Watchdog timer 0	315
0x4000.1000	0x4000.3FFF	Reserved	-
0x4000.4000	0x4000.4FFF	GPIO Port A	243
0x4000.5000	0x4000.5FFF	GPIO Port B	243
0x4000.6000	0x4000.6FFF	GPIO Port C	243
0x4000.7000	0x4000.7FFF	GPIO Port D	243
0x4000.8000	0x4000.8FFF	SSI0	425

**Table 2-4. Memory Map (continued)**

<b>Start</b>	<b>End</b>	<b>Description</b>	<b>For details, see page ...</b>
0x4000.9000	0x4000.BFFF	Reserved	-
0x4000.C000	0x4000.CFFF	UART0	380
0x4000.D000	0x4000.DFFF	UART1	380
0x4000.E000	0x4001.FFFF	Reserved	-
<b>Peripherals</b>			
0x4002.0000	0x4002.0FFF	I <sup>2</sup> C 0	466
0x4002.1000	0x4002.3FFF	Reserved	-
0x4002.4000	0x4002.4FFF	GPIO Port E	243
0x4002.5000	0x4002.7FFF	Reserved	-
0x4002.8000	0x4002.8FFF	PWM	509
0x4002.9000	0x4002.FFFF	Reserved	-
0x4003.0000	0x4003.0FFF	Timer 0	287
0x4003.1000	0x4003.1FFF	Timer 1	287
0x4003.2000	0x4003.2FFF	Timer 2	287
0x4003.3000	0x4003.7FFF	Reserved	-
0x4003.8000	0x4003.8FFF	ADC0	345
0x4003.9000	0x4003.BFFF	Reserved	-
0x4003.C000	0x4003.CFFF	Analog Comparators	488
0x4003.D000	0x400F.CFFF	Reserved	-
0x400F.D000	0x400F.DFFF	Flash memory control	220
0x400F.E000	0x400F.EFFF	System control	166
0x400F.F000	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)	46
0xE000.1000	0xE000.1FFF	Data Watchpoint and Trace (DWT)	46
0xE000.2000	0xE000.2FFF	Flash Patch and Breakpoint (FPB)	46
0xE000.3000	0xE000.DFFF	Reserved	-
0xE000.E000	0xE000.EFFF	Cortex-M3 Peripherals (SysTick, NVIC, MPU and SCB)	94
0xE000.F000	0xE003.FFFF	Reserved	-
0xE004.0000	0xE004.0FFF	Trace Port Interface Unit (TPIU)	47
0xE004.1000	0xFFFF.FFFF	Reserved	-

## 2.4.1 Memory Regions, Types and Attributes

The memory map and the programming of the MPU split the memory map into regions. Each region has a defined memory type, and some regions have additional memory attributes. The memory type and attributes determine the behavior of accesses to the region.

The memory types are:

- Normal: The processor can re-order transactions for efficiency and perform speculative reads.

- Device: The processor preserves transaction order relative to other transactions to Device or Strongly Ordered memory.
- Strongly Ordered: The processor preserves transaction order relative to all other transactions.

The different ordering requirements for Device and Strongly Ordered memory mean that the memory system can buffer a write to Device memory but must not buffer a write to Strongly Ordered memory.

An additional memory attribute is Execute Never (XN), which means the processor prevents instruction accesses. A fault exception is generated only on execution of an instruction executed from an XN region.

#### 2.4.2 Memory System Ordering of Memory Accesses

For most memory accesses caused by explicit memory access instructions, the memory system does not guarantee that the order in which the accesses complete matches the program order of the instructions, providing the order does not affect the behavior of the instruction sequence. Normally, if correct program execution depends on two memory accesses completing in program order, software must insert a memory barrier instruction between the memory access instructions (see “Software Ordering of Memory Accesses” on page 66).

However, the memory system does guarantee ordering of accesses to Device and Strongly Ordered memory. For two memory access instructions A1 and A2, if both A1 and A2 are accesses to either Device or Strongly Ordered memory, and if A1 occurs before A2 in program order, A1 is always observed before A2.

#### 2.4.3 Behavior of Memory Accesses

Table 2-5 on page 65 shows the behavior of accesses to each region in the memory map. See “Memory Regions, Types and Attributes” on page 64 for more information on memory types and the XN attribute. Stellaris devices may have reserved memory areas within the address ranges shown below (refer to Table 2-4 on page 63 for more information).

**Table 2-5. Memory Access Behavior**

Address Range	Memory Region	Memory Type	Execute Never (XN)	Description
0x0000.0000 - 0x1FFF.FFFF	Code	Normal	-	This executable region is for program code. Data can also be stored here.
0x2000.0000 - 0x3FFF.FFFF	SRAM	Normal	-	This executable region is for data. Code can also be stored here. This region includes bit band and bit band alias areas (see Table 2-6 on page 67).
0x4000.0000 - 0x5FFF.FFFF	Peripheral	Device	XN	This region includes bit band and bit band alias areas (see Table 2-7 on page 67).
0x6000.0000 - 0x9FFF.FFFF	External RAM	Normal	-	This executable region is for data.
0xA000.0000 - 0xDFFF.FFFF	External device	Device	XN	This region is for external device memory.
0xE000.0000- 0xE00F.FFFF	Private peripheral bus	Strongly Ordered	XN	This region includes the NVIC, system timer, and system control block.
0xE010.0000- 0xFFFF.FFFF	Reserved	-	-	-

The Code, SRAM, and external RAM regions can hold programs. However, it is recommended that programs always use the Code region because the Cortex-M3 has separate buses that can perform instruction fetches and data accesses simultaneously.

The MPU can override the default memory access behavior described in this section. For more information, see “Memory Protection Unit (MPU)” on page 89.

The Cortex-M3 prefetches instructions ahead of execution and speculatively prefetches from branch target addresses.

#### 2.4.4 Software Ordering of Memory Accesses

The order of instructions in the program flow does not always guarantee the order of the corresponding memory transactions for the following reasons:

- The processor can reorder some memory accesses to improve efficiency, providing this does not affect the behavior of the instruction sequence.
- The processor has multiple bus interfaces.
- Memory or devices in the memory map have different wait states.
- Some memory accesses are buffered or speculative.

“Memory System Ordering of Memory Accesses” on page 65 describes the cases where the memory system guarantees the order of memory accesses. Otherwise, if the order of memory accesses is critical, software must include memory barrier instructions to force that ordering. The Cortex-M3 has the following memory barrier instructions:

- The Data Memory Barrier (`DMB`) instruction ensures that outstanding memory transactions complete before subsequent memory transactions.
- The Data Synchronization Barrier (`DSB`) instruction ensures that outstanding memory transactions complete before subsequent instructions execute.
- The Instruction Synchronization Barrier (`ISB`) instruction ensures that the effect of all completed memory transactions is recognizable by subsequent instructions.

Memory barrier instructions can be used in the following situations:

- MPU programming
  - If the MPU settings are changed and the change must be effective on the very next instruction, use a `DSB` instruction to ensure the effect of the MPU takes place immediately at the end of context switching.
  - Use an `ISB` instruction to ensure the new MPU setting takes effect immediately after programming the MPU region or regions, if the MPU configuration code was accessed using a branch or call. If the MPU configuration code is entered using exception mechanisms, then an `ISB` instruction is not required.
- Vector table
  - If the program changes an entry in the vector table and then enables the corresponding exception, use a `DMB` instruction between the operations. The `DMB` instruction ensures that if the exception is taken immediately after being enabled, the processor uses the new exception vector.
- Self-modifying code

If a program contains self-modifying code, use an `ISB` instruction immediately after the code modification in the program. The `ISB` instruction ensures subsequent instruction execution uses the updated program.

- **Memory map switching**

If the system contains a memory map switching mechanism, use a `DSB` instruction after switching the memory map in the program. The `DSB` instruction ensures subsequent instruction execution uses the updated memory map.

- **Dynamic exception priority change**

When an exception priority has to change when the exception is pending or active, use `DSB` instructions after the change. The change then takes effect on completion of the `DSB` instruction.

Memory accesses to Strongly Ordered memory, such as the System Control Block, do not require the use of `DMB` instructions.

For more information on the memory barrier instructions, see the *Cortex™-M3/M4 Instruction Set Technical User's Manual*.

## 2.4.5 Bit-Banding

A bit-band region maps each word in a bit-band alias region to a single bit in the bit-band region. The bit-band regions occupy the lowest 1 MB of the SRAM and peripheral memory regions. Accesses to the 32-MB SRAM alias region map to the 1-MB SRAM bit-band region, as shown in Table 2-6 on page 67. Accesses to the 32-MB peripheral alias region map to the 1-MB peripheral bit-band region, as shown in Table 2-7 on page 67. For the specific address range of the bit-band regions, see Table 2-4 on page 63.

**Note:** A word access to the SRAM or the peripheral bit-band alias region maps to a single bit in the SRAM or peripheral bit-band region.

A word access to a bit band address results in a word access to the underlying memory, and similarly for halfword and byte accesses. This allows bit band accesses to match the access requirements of the underlying peripheral.

**Table 2-6. SRAM Memory Bit-Banding Regions**

Address Range		Memory Region	Instruction and Data Accesses
Start	End		
0x2000.0000	0x2000.1FFF	SRAM bit-band region	Direct accesses to this memory range behave as SRAM memory accesses, but this region is also bit addressable through bit-band alias.
0x2200.0000	0x2203.FFFF	SRAM bit-band alias	Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not remapped.

**Table 2-7. Peripheral Memory Bit-Banding Regions**

Address Range		Memory Region	Instruction and Data Accesses
Start	End		
0x4000.0000	0x400F.FFFF	Peripheral bit-band region	Direct accesses to this memory range behave as peripheral memory accesses, but this region is also bit addressable through bit-band alias.

**Table 2-7. Peripheral Memory Bit-Banding Regions (continued)**

Address Range		Memory Region	Instruction and Data Accesses
Start	End		
0x4200.0000	0x43FF.FFFF	Peripheral bit-band alias	Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not permitted.

The following formula shows how the alias region maps onto the bit-band region:

$$\text{bit\_word\_offset} = (\text{byte\_offset} \times 32) + (\text{bit\_number} \times 4)$$

$$\text{bit\_word\_addr} = \text{bit\_band\_base} + \text{bit\_word\_offset}$$

where:

**bit\_word\_offset**

The position of the target bit in the bit-band memory region.

**bit\_word\_addr**

The address of the word in the alias memory region that maps to the targeted bit.

**bit\_band\_base**

The starting address of the alias region.

**byte\_offset**

The number of the byte in the bit-band region that contains the targeted bit.

**bit\_number**

The bit position, 0-7, of the targeted bit.

Figure 2-4 on page 69 shows examples of bit-band mapping between the SRAM bit-band alias region and the SRAM bit-band region:

- The alias word at 0x23FF.FFE0 maps to bit 0 of the bit-band byte at 0x200F.FFFF:

$$0x23FF.FFE0 = 0x2200.0000 + (0x000F.FFFF * 32) + (0 * 4)$$

- The alias word at 0x23FF.FFFC maps to bit 7 of the bit-band byte at 0x200F.FFFF:

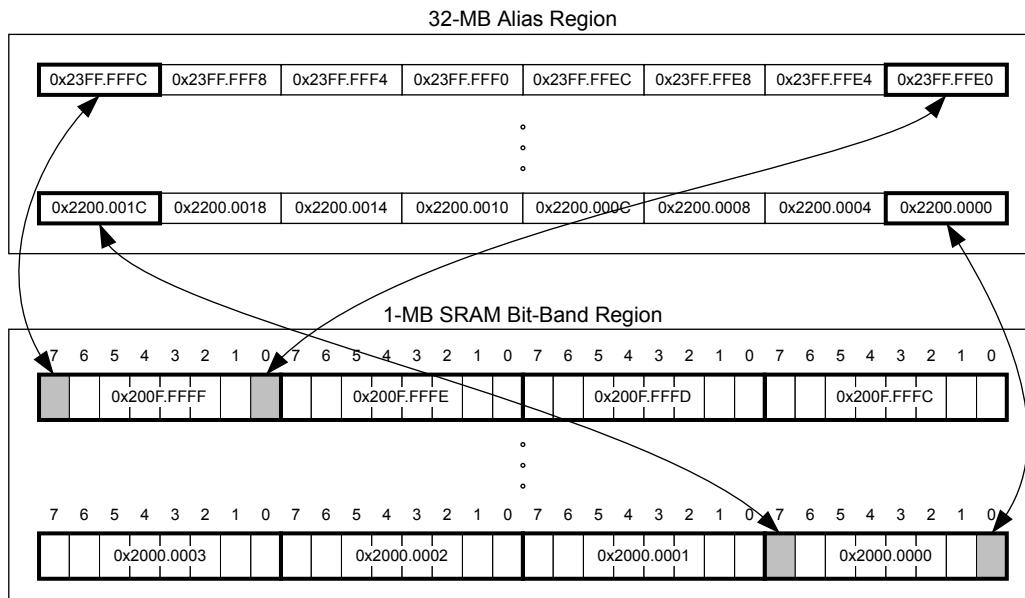
$$0x23FF.FFFC = 0x2200.0000 + (0x000F.FFFF * 32) + (7 * 4)$$

- The alias word at 0x2200.0000 maps to bit 0 of the bit-band byte at 0x2000.0000:

$$0x2200.0000 = 0x2200.0000 + (0 * 32) + (0 * 4)$$

- The alias word at 0x2200.001C maps to bit 7 of the bit-band byte at 0x2000.0000:

$$0x2200.001C = 0x2200.0000 + (0 * 32) + (7 * 4)$$

**Figure 2-4. Bit-Band Mapping**

#### 2.4.5.1 Directly Accessing an Alias Region

Writing to a word in the alias region updates a single bit in the bit-band region.

Bit 0 of the value written to a word in the alias region determines the value written to the targeted bit in the bit-band region. Writing a value with bit 0 set writes a 1 to the bit-band bit, and writing a value with bit 0 clear writes a 0 to the bit-band bit.

Bits 31:1 of the alias word have no effect on the bit-band bit. Writing 0x01 has the same effect as writing 0xFF. Writing 0x00 has the same effect as writing 0x0E.

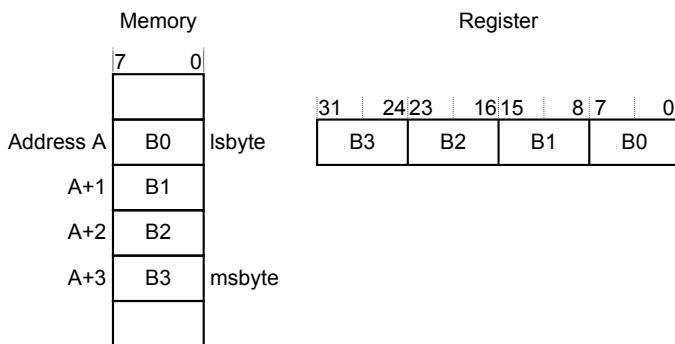
When reading a word in the alias region, 0x0000.0000 indicates that the targeted bit in the bit-band region is clear and 0x0000.0001 indicates that the targeted bit in the bit-band region is set.

#### 2.4.5.2 Directly Accessing a Bit-Band Region

“Behavior of Memory Accesses” on page 65 describes the behavior of direct byte, halfword, or word accesses to the bit-band regions.

### 2.4.6 Data Storage

The processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0-3 hold the first stored word, and bytes 4-7 hold the second stored word. Data is stored in little-endian format, with the least-significant byte (lsbyte) of a word stored at the lowest-numbered byte, and the most-significant byte (msbyte) stored at the highest-numbered byte. Figure 2-5 on page 70 illustrates how data is stored.

**Figure 2-5. Data Storage**

## 2.4.7 Synchronization Primitives

The Cortex-M3 instruction set includes pairs of synchronization primitives which provide a non-blocking mechanism that a thread or process can use to obtain exclusive access to a memory location. Software can use these primitives to perform a guaranteed read-modify-write memory update sequence or for a semaphore mechanism.

A pair of synchronization primitives consists of:

- A Load-Exclusive instruction, which is used to read the value of a memory location and requests exclusive access to that location.
- A Store-Exclusive instruction, which is used to attempt to write to the same memory location and returns a status bit to a register. If this status bit is clear, it indicates that the thread or process gained exclusive access to the memory and the write succeeds; if this status bit is set, it indicates that the thread or process did not gain exclusive access to the memory and no write was performed.

The pairs of Load-Exclusive and Store-Exclusive instructions are:

- The word instructions `LDREX` and `STREX`
- The halfword instructions `LDREXH` and `STREXH`
- The byte instructions `LDREXB` and `STREXB`

Software must use a Load-Exclusive instruction with the corresponding Store-Exclusive instruction.

To perform an exclusive read-modify-write of a memory location, software must:

1. Use a Load-Exclusive instruction to read the value of the location.
2. Modify the value, as required.
3. Use a Store-Exclusive instruction to attempt to write the new value back to the memory location.
4. Test the returned status bit.

If the status bit is clear, the read-modify-write completed successfully. If the status bit is set, no write was performed, which indicates that the value returned at step 1 might be out of date. The software must retry the entire read-modify-write sequence.

Software can use the synchronization primitives to implement a semaphore as follows:

1. Use a Load-Exclusive instruction to read from the semaphore address to check whether the semaphore is free.
2. If the semaphore is free, use a Store-Exclusive to write the claim value to the semaphore address.
3. If the returned status bit from step 2 indicates that the Store-Exclusive succeeded, then the software has claimed the semaphore. However, if the Store-Exclusive failed, another process might have claimed the semaphore after the software performed step 1.

The Cortex-M3 includes an exclusive access monitor that tags the fact that the processor has executed a Load-Exclusive instruction. The processor removes its exclusive access tag if:

- It executes a CLREX instruction.
- It executes a Store-Exclusive instruction, regardless of whether the write succeeds.
- An exception occurs, which means the processor can resolve semaphore conflicts between different threads.

For more information about the synchronization primitive instructions, see the *Cortex™-M3/M4 Instruction Set Technical User's Manual*.

## 2.5 Exception Model

The ARM Cortex-M3 processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions 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, enabling 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 2-8 on page 73 lists all exception types. Software can set eight priority levels on seven of these exceptions (system handlers) as well as on 26 interrupts (listed in Table 2-9 on page 74).

Priorities on the system handlers are set with the NVIC **System Handler Priority n (SYSPRI<sub>n</sub>)** registers. Interrupts are enabled through the NVIC **Interrupt Set Enable n (EN<sub>n</sub>)** register and prioritized with the NVIC **Interrupt Priority n (PRI<sub>n</sub>)** registers. Priorities can be grouped by splitting priority levels into preemption priorities and subpriorities. All the interrupt registers are described in “Nested Vectored Interrupt Controller (NVIC)” on page 87.

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

**Important:** After a write to clear an interrupt source, it may take several processor cycles for the NVIC to see the interrupt source de-assert. Thus if the interrupt clear is done as the last action in an interrupt handler, it is possible for the interrupt handler to complete while the NVIC sees the interrupt as still asserted, causing the interrupt handler to be re-entered errantly. This situation can be avoided by either clearing the interrupt source at the beginning of the interrupt handler or by performing a read or write after the write to clear the interrupt source (and flush the write buffer).

See “Nested Vectored Interrupt Controller (NVIC)” on page 87 for more information on exceptions and interrupts.

### 2.5.1 Exception States

Each exception is in one of the following states:

- **Inactive.** The exception is not active and not pending.
- **Pending.** The exception is waiting to be serviced by the processor. An interrupt request from a peripheral or from software can change the state of the corresponding interrupt to pending.
- **Active.** An exception that is being serviced by the processor but has not completed.  
**Note:** An exception handler can interrupt the execution of another exception handler. In this case, both exceptions are in the active state.
- **Active and Pending.** The exception is being serviced by the processor, and there is a pending exception from the same source.

### 2.5.2 Exception Types

The exception types are:

- **Reset.** Reset is invoked on power up or a warm reset. The exception model treats reset as a special form of exception. When reset is asserted, the operation of the processor stops, potentially at any point in an instruction. When reset is deasserted, execution restarts from the address provided by the reset entry in the vector table. Execution restarts as privileged execution in Thread mode.
- **NMI.** A non-maskable Interrupt (NMI) can be signaled using the NMI signal or triggered by software using the **Interrupt Control and State (INTCTRL)** register. This exception has the highest priority other than reset. NMI is permanently enabled and has a fixed priority of -2. NMIs cannot be masked or prevented from activation by any other exception or preempted by any exception other than reset.
- **Hard Fault.** A hard fault is an exception that occurs because of an error during exception processing, or because an exception cannot be managed by any other exception mechanism. Hard faults have a fixed priority of -1, meaning they have higher priority than any exception with configurable priority.
- **Memory Management Fault.** A memory management fault is an exception that occurs because of a memory protection related fault, including access violation and no match. The MPU or the fixed memory protection constraints determine this fault, for both instruction and data memory transactions. This fault is used to abort instruction accesses to Execute Never (XN) memory regions, even if the MPU is disabled.
- **Bus Fault.** A bus fault is an exception that occurs because of a memory-related fault for an instruction or data memory transaction such as a prefetch fault or a memory access fault. This fault can be enabled or disabled.
- **Usage Fault.** A usage fault is an exception that occurs because of a fault related to instruction execution, such as:
  - An undefined instruction
  - An illegal unaligned access
  - Invalid state on instruction execution

- An error on exception return
- An unaligned address on a word or halfword memory access or division by zero can cause a usage fault when the core is properly configured.
- **SVCall.** A supervisor call (SVC) is an exception that is triggered by the SVC instruction. In an OS environment, applications can use SVC instructions to access OS kernel functions and device drivers.
  - **Debug Monitor.** This exception is caused by the debug monitor (when not halting). This exception is only active when enabled. This exception does not activate if it is a lower priority than the current activation.
  - **PendSV.** PendSV is a pendable, interrupt-driven request for system-level service. In an OS environment, use PendSV for context switching when no other exception is active. PendSV is triggered using the **Interrupt Control and State (INTCTRL)** register.
  - **SysTick.** A SysTick exception is an exception that the system timer generates when it reaches zero when it is enabled to generate an interrupt. Software can also generate a SysTick exception using the **Interrupt Control and State (INTCTRL)** register. In an OS environment, the processor can use this exception as system tick.
  - **Interrupt (IRQ).** An interrupt, or IRQ, is an exception signaled by a peripheral or generated by a software request and fed through the NVIC (prioritized). All interrupts are asynchronous to instruction execution. In the system, peripherals use interrupts to communicate with the processor. Table 2-9 on page 74 lists the interrupts on the LM3S811 controller.

For an asynchronous exception, other than reset, the processor can execute another instruction between when the exception is triggered and when the processor enters the exception handler.

Privileged software can disable the exceptions that Table 2-8 on page 73 shows as having configurable priority (see the **SYSHNDCTRL** register on page 122 and the **DIS0** register on page 101).

For more information about hard faults, memory management faults, bus faults, and usage faults, see “Fault Handling” on page 78.

**Table 2-8. Exception Types**

Exception Type	Vector Number	Priority <sup>a</sup>	Vector Address or Offset <sup>b</sup>	Activation
-	0	-	0x0000.0000	Stack top is loaded from the first entry of the vector table on reset.
Reset	1	-3 (highest)	0x0000.0004	Asynchronous
Non-Maskable Interrupt (NMI)	2	-2	0x0000.0008	Asynchronous
Hard Fault	3	-1	0x0000.000C	-
Memory Management	4	programmable <sup>c</sup>	0x0000.0010	Synchronous
Bus Fault	5	programmable <sup>c</sup>	0x0000.0014	Synchronous when precise and asynchronous when imprecise
Usage Fault	6	programmable <sup>c</sup>	0x0000.0018	Synchronous
-	7-10	-	-	Reserved
SVCall	11	programmable <sup>c</sup>	0x0000.002C	Synchronous
Debug Monitor	12	programmable <sup>c</sup>	0x0000.0030	Synchronous
-	13	-	-	Reserved

**Table 2-8. Exception Types (continued)**

Exception Type	Vector Number	Priority <sup>a</sup>	Vector Address or Offset <sup>b</sup>	Activation
PendSV	14	programmable <sup>c</sup>	0x0000.0038	Asynchronous
SysTick	15	programmable <sup>c</sup>	0x0000.003C	Asynchronous
Interrupts	16 and above	programmable <sup>d</sup>	0x0000.0040 and above	Asynchronous

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

b. See “Vector Table” on page 75.

c. See **SYSPRI1** on page 119.

d. See **PRIn** registers on page 105.

**Table 2-9. Interrupts**

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Vector Address or Offset	Description
0-15	-	0x0000.0000 - 0x0000.003C	Processor exceptions
16	0	0x0000.0040	GPIO Port A
17	1	0x0000.0044	GPIO Port B
18	2	0x0000.0048	GPIO Port C
19	3	0x0000.004C	GPIO Port D
20	4	0x0000.0050	GPIO Port E
21	5	0x0000.0054	UART0
22	6	0x0000.0058	UART1
23	7	0x0000.005C	SSI0
24	8	0x0000.0060	I <sup>2</sup> C0
25	9	-	Reserved
26	10	0x0000.0068	PWM Generator 0
27	11	0x0000.006C	PWM Generator 1
28	12	0x0000.0070	PWM Generator 2
29	13	-	Reserved
30	14	0x0000.0078	ADC0 Sequence 0
31	15	0x0000.007C	ADC0 Sequence 1
32	16	0x0000.0080	ADC0 Sequence 2
33	17	0x0000.0084	ADC0 Sequence 3
34	18	0x0000.0088	Watchdog Timer 0
35	19	0x0000.008C	Timer 0A
36	20	0x0000.0090	Timer 0B
37	21	0x0000.0094	Timer 1A
38	22	0x0000.0098	Timer 1B
39	23	0x0000.009C	Timer 2A
40	24	0x0000.00A0	Timer 2B
41	25	0x0000.00A4	Analog Comparator 0
42-43	26-27	-	Reserved
44	28	0x0000.00B0	System Control
45	29	0x0000.00B4	Flash Memory Control

### 2.5.3 Exception Handlers

The processor handles exceptions using:

- **Interrupt Service Routines (ISRs).** Interrupts (IRQx) are the exceptions handled by ISRs.
- **Fault Handlers.** Hard fault, memory management fault, usage fault, and bus fault are fault exceptions handled by the fault handlers.
- **System Handlers.** NMI, PendSV, SVCALL, SysTick, and the fault exceptions are all system exceptions that are handled by system handlers.

### 2.5.4 Vector Table

The vector table contains the reset value of the stack pointer and the start addresses, also called exception vectors, for all exception handlers. The vector table is constructed using the vector address or offset shown in Table 2-8 on page 73. Figure 2-6 on page 75 shows the order of the exception vectors in the vector table. The least-significant bit of each vector must be 1, indicating that the exception handler is Thumb code

**Figure 2-6. Vector Table**

Exception number	IRQ number	Offset	Vector
45	29	0x00B4	IRQ29
.	.	.	.
.	.	0x004C	IRQ2
18	2	0x0048	IRQ1
17	1	0x0044	IRQ0
16	0	0x0040	Systick
15	-1	0x003C	PendSV
14	-2	0x0038	Reserved
13			Reserved for Debug
12			SVCALL
11	-5	0x002C	
10			
9			Reserved
8			
7			
6	-10	0x0018	Usage fault
5	-11	0x0014	Bus fault
4	-12	0x0010	Memory management fault
3	-13	0x000C	Hard fault
2	-14	0x0008	NMI
1		0x0004	Reset
		0x0000	Initial SP value

On system reset, the vector table is fixed at address 0x0000.0000. Privileged software can write to the **Vector Table Offset (VTABLE)** register to relocate the vector table start address to a different

memory location, in the range 0x0000.0100 to 0x3FFF.FF00 (see “Vector Table” on page 75). Note that when configuring the **VTABLE** register, the offset must be aligned on a 256-byte boundary.

### 2.5.5 Exception Priorities

As Table 2-8 on page 73 shows, all exceptions have an associated priority, with a lower priority value indicating a higher priority and configurable priorities for all exceptions except Reset, Hard fault, and NMI. If software does not configure any priorities, then all exceptions with a configurable priority have a priority of 0. For information about configuring exception priorities, see page 119 and page 105.

**Note:** Configurable priority values for the Stellaris implementation are in the range 0-7. This means that the Reset, Hard fault, and NMI exceptions, with fixed negative priority values, always have higher priority than any other exception.

For example, assigning a higher priority value to IRQ[0] and a lower priority value to IRQ[1] means that IRQ[1] has higher priority than IRQ[0]. If both IRQ[1] and IRQ[0] are asserted, IRQ[1] is processed before IRQ[0].

If multiple pending exceptions have the same priority, the pending exception with the lowest exception number takes precedence. For example, if both IRQ[0] and IRQ[1] are pending and have the same priority, then IRQ[0] is processed before IRQ[1].

When the processor is executing an exception handler, the exception handler is preempted if a higher priority exception occurs. If an exception occurs with the same priority as the exception being handled, the handler is not preempted, irrespective of the exception number. However, the status of the new interrupt changes to pending.

### 2.5.6 Interrupt Priority Grouping

To increase priority control in systems with interrupts, the NVIC supports priority grouping. This grouping divides each interrupt priority register entry into two fields:

- An upper field that defines the group priority
- A lower field that defines a subpriority within the group

Only the group priority determines preemption of interrupt exceptions. When the processor is executing an interrupt exception handler, another interrupt with the same group priority as the interrupt being handled does not preempt the handler.

If multiple pending interrupts have the same group priority, the subpriority field determines the order in which they are processed. If multiple pending interrupts have the same group priority and subpriority, the interrupt with the lowest IRQ number is processed first.

For information about splitting the interrupt priority fields into group priority and subpriority, see page 113.

### 2.5.7 Exception Entry and Return

Descriptions of exception handling use the following terms:

- **Preemption.** When the processor is executing an exception handler, an exception can preempt the exception handler if its priority is higher than the priority of the exception being handled. See “Interrupt Priority Grouping” on page 76 for more information about preemption by an interrupt. When one exception preempts another, the exceptions are called nested exceptions. See “Exception Entry” on page 77 for more information.

- **Return.** Return occurs when the exception handler is completed, and there is no pending exception with sufficient priority to be serviced and the completed exception handler was not handling a late-arriving exception. The processor pops the stack and restores the processor state to the state it had before the interrupt occurred. See “Exception Return” on page 78 for more information.
- **Tail-Chaining.** This mechanism speeds up exception servicing. On completion of an exception handler, if there is a pending exception that meets the requirements for exception entry, the stack pop is skipped and control transfers to the new exception handler.
- **Late-Arriving.** This mechanism speeds up preemption. If a higher priority exception occurs during state saving for a previous exception, the processor switches to handle the higher priority exception and initiates the vector fetch for that exception. State saving is not affected by late arrival because the state saved is the same for both exceptions. Therefore, the state saving continues uninterrupted. The processor can accept a late arriving exception until the first instruction of the exception handler of the original exception enters the execute stage of the processor. On return from the exception handler of the late-arriving exception, the normal tail-chaining rules apply.

#### 2.5.7.1 Exception Entry

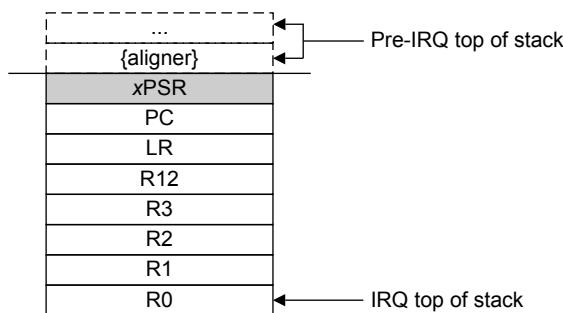
Exception entry occurs when there is a pending exception with sufficient priority and either the processor is in Thread mode or the new exception is of higher priority than the exception being handled, in which case the new exception preempts the original exception.

When one exception preempts another, the exceptions are nested.

Sufficient priority means the exception has more priority than any limits set by the mask registers (see **PRIMASK** on page 59, **FAULTMASK** on page 60, and **BASEPRI** on page 61). An exception with less priority than this is pending but is not handled by the processor.

When the processor takes an exception, unless the exception is a tail-chained or a late-arriving exception, the processor pushes information onto the current stack. This operation is referred to as *stacking* and the structure of eight data words is referred to as *stack frame*.

**Figure 2-7. Exception Stack Frame**



Immediately after stacking, the stack pointer indicates the lowest address in the stack frame. Unless stack alignment is disabled, the stack frame is aligned to a double-word address. If the **STKALIGN** bit of the **Configuration Control (CCR)** register is set, stack align adjustment is performed during stacking.

The stack frame includes the return address, which is the address of the next instruction in the interrupted program. This value is restored to the **PC** at exception return so that the interrupted program resumes.

In parallel to the stacking operation, the processor performs a vector fetch that reads the exception handler start address from the vector table. When stacking is complete, the processor starts executing the exception handler. At the same time, the processor writes an EXC\_RETURN value to the **LR**, indicating which stack pointer corresponds to the stack frame and what operation mode the processor was in before the entry occurred.

If no higher-priority exception occurs during exception entry, the processor starts executing the exception handler and automatically changes the status of the corresponding pending interrupt to active.

If another higher-priority exception occurs during exception entry, known as late arrival, the processor starts executing the exception handler for this exception and does not change the pending status of the earlier exception.

### 2.5.7.2 Exception Return

Exception return occurs when the processor is in Handler mode and executes one of the following instructions to load the EXC\_RETURN value into the **PC**:

- An **LDM** or **POP** instruction that loads the **PC**
- A **BX** instruction using any register
- An **LDR** instruction with the **PC** as the destination

EXC\_RETURN is the value loaded into the **LR** on exception entry. The exception mechanism relies on this value to detect when the processor has completed an exception handler. The lowest four bits of this value provide information on the return stack and processor mode. Table 2-10 on page 78 shows the EXC\_RETURN values with a description of the exception return behavior.

EXC\_RETURN bits 31:4 are all set. When this value is loaded into the **PC**, it indicates to the processor that the exception is complete, and the processor initiates the appropriate exception return sequence.

**Table 2-10. Exception Return Behavior**

EXC_RETURN[31:0]	Description
0xFFFF.FFF0	Reserved
0xFFFF.FFF1	Return to Handler mode. Exception return uses state from <b>MSP</b> . Execution uses <b>MSP</b> after return.
0xFFFF.FFF2 - 0xFFFF.FFF8	Reserved
0xFFFF.FFF9	Return to Thread mode. Exception return uses state from <b>MSP</b> . Execution uses <b>MSP</b> after return.
0xFFFF.FFFA - 0xFFFF.FFFC	Reserved
0xFFFF.FFFD	Return to Thread mode. Exception return uses state from <b>PSP</b> . Execution uses <b>PSP</b> after return.
0xFFFF.FFFE - 0xFFFF.FFFF	Reserved

## 2.6 Fault Handling

Faults are a subset of the exceptions (see “Exception Model” on page 71). The following conditions generate a fault:

- A bus error on an instruction fetch or vector table load or a data access.
- An internally detected error such as an undefined instruction or an attempt to change state with a BX instruction.
- Attempting to execute an instruction from a memory region marked as Non-Executable (XN).
- An MPU fault because of a privilege violation or an attempt to access an unmanaged region.

## 2.6.1 Fault Types

Table 2-11 on page 79 shows the types of fault, the handler used for the fault, the corresponding fault status register, and the register bit that indicates the fault has occurred. See page 126 for more information about the fault status registers.

**Table 2-11. Faults**

Fault	Handler	Fault Status Register	Bit Name
Bus error on a vector read	Hard fault	Hard Fault Status (HFAULTSTAT)	VECT
Fault escalated to a hard fault	Hard fault	Hard Fault Status (HFAULTSTAT)	FORCED
MPU or default memory mismatch on instruction access	Memory management fault	Memory Management Fault Status (MFAULTSTAT)	IERR <sup>a</sup>
MPU or default memory mismatch on data access	Memory management fault	Memory Management Fault Status (MFAULTSTAT)	DERR
MPU or default memory mismatch on exception stacking	Memory management fault	Memory Management Fault Status (MFAULTSTAT)	MSTKE
MPU or default memory mismatch on exception unstacking	Memory management fault	Memory Management Fault Status (MFAULTSTAT)	MUSTKE
Bus error during exception stacking	Bus fault	Bus Fault Status (BFAULTSTAT)	BSTKE
Bus error during exception unstacking	Bus fault	Bus Fault Status (BFAULTSTAT)	BUSTKE
Bus error during instruction prefetch	Bus fault	Bus Fault Status (BFAULTSTAT)	IBUS
Precise data bus error	Bus fault	Bus Fault Status (BFAULTSTAT)	PRECISE
Imprecise data bus error	Bus fault	Bus Fault Status (BFAULTSTAT)	IMPRE
Attempt to access a coprocessor	Usage fault	Usage Fault Status (UFAULTSTAT)	NOCP
Undefined instruction	Usage fault	Usage Fault Status (UFAULTSTAT)	UNDEF
Attempt to enter an invalid instruction set state <sup>b</sup>	Usage fault	Usage Fault Status (UFAULTSTAT)	INVSTAT
Invalid EXC_RETURN value	Usage fault	Usage Fault Status (UFAULTSTAT)	INVPC
Illegal unaligned load or store	Usage fault	Usage Fault Status (UFAULTSTAT)	UNALIGN
Divide by 0	Usage fault	Usage Fault Status (UFAULTSTAT)	DIV0

a. Occurs on an access to an XN region even if the MPU is disabled.

b. Attempting to use an instruction set other than the Thumb instruction set, or returning to a non load-store-multiple instruction with ICI continuation.

## 2.6.2 Fault Escalation and Hard Faults

All fault exceptions except for hard fault have configurable exception priority (see **SYSPRI1** on page 119). Software can disable execution of the handlers for these faults (see **SYSHNDCTRL** on page 122).

Usually, the exception priority, together with the values of the exception mask registers, determines whether the processor enters the fault handler, and whether a fault handler can preempt another fault handler as described in “Exception Model” on page 71.

In some situations, a fault with configurable priority is treated as a hard fault. This process is called priority escalation, and the fault is described as *escalated to hard fault*. Escalation to hard fault occurs when:

- A fault handler causes the same kind of fault as the one it is servicing. This escalation to hard fault occurs because a fault handler cannot preempt itself because it must have the same priority as the current priority level.
- A fault handler causes a fault with the same or lower priority as the fault it is servicing. This situation happens because the handler for the new fault cannot preempt the currently executing fault handler.
- An exception handler causes a fault for which the priority is the same as or lower than the currently executing exception.
- A fault occurs and the handler for that fault is not enabled.

If a bus fault occurs during a stack push when entering a bus fault handler, the bus fault does not escalate to a hard fault. Thus if a corrupted stack causes a fault, the fault handler executes even though the stack push for the handler failed. The fault handler operates but the stack contents are corrupted.

**Note:** Only Reset and NMI can preempt the fixed priority hard fault. A hard fault can preempt any exception other than Reset, NMI, or another hard fault.

### 2.6.3 Fault Status Registers and Fault Address Registers

The fault status registers indicate the cause of a fault. For bus faults and memory management faults, the fault address register indicates the address accessed by the operation that caused the fault, as shown in Table 2-12 on page 80.

**Table 2-12. Fault Status and Fault Address Registers**

Handler	Status Register Name	Address Register Name	Register Description
Hard fault	Hard Fault Status (HFAULTSTAT)	-	page 132
Memory management fault	Memory Management Fault Status (MFAULTSTAT)	Memory Management Fault Address (MMADDR)	page 126 page 133
Bus fault	Bus Fault Status (BFAULTSTAT)	Bus Fault Address (FAULTADDR)	page 126 page 134
Usage fault	Usage Fault Status (UFAULTSTAT)	-	page 126

### 2.6.4 Lockup

The processor enters a lockup state if a hard fault occurs when executing the NMI or hard fault handlers. When the processor is in the lockup state, it does not execute any instructions. The processor remains in lockup state until it is reset, an NMI occurs, or it is halted by a debugger.

**Note:** If the lockup state occurs from the NMI handler, a subsequent NMI does not cause the processor to leave the lockup state.

## 2.7 Power Management

The Cortex-M3 processor sleep modes reduce power consumption:

- Sleep mode stops the processor clock.
- Deep-sleep mode stops the system clock and switches off the PLL and Flash memory.

The SLEEPDEEP bit of the **System Control (SYSCTRL)** register selects which sleep mode is used (see page 115). For more information about the behavior of the sleep modes, see “System Control” on page 164.

This section describes the mechanisms for entering sleep mode and the conditions for waking up from sleep mode, both of which apply to Sleep mode and Deep-sleep mode.

### 2.7.1 Entering Sleep Modes

This section describes the mechanisms software can use to put the processor into one of the sleep modes.

The system can generate spurious wake-up events, for example a debug operation wakes up the processor. Therefore, software must be able to put the processor back into sleep mode after such an event. A program might have an idle loop to put the processor back to sleep mode.

#### 2.7.1.1 Wait for Interrupt

The wait for interrupt instruction, `WFI`, causes immediate entry to sleep mode unless the wake-up condition is true (see “Wake Up from WFI or Sleep-on-Exit” on page 82). When the processor executes a `WFI` instruction, it stops executing instructions and enters sleep mode. See the *Cortex™-M3/M4 Instruction Set Technical User’s Manual* for more information.

#### 2.7.1.2 Wait for Event

The wait for event instruction, `WFE`, causes entry to sleep mode conditional on the value of a one-bit event register. When the processor executes a `WFE` instruction, it checks the event register. If the register is 0, the processor stops executing instructions and enters sleep mode. If the register is 1, the processor clears the register and continues executing instructions without entering sleep mode.

If the event register is 1, the processor must not enter sleep mode on execution of a `WFE` instruction. Typically, this situation occurs if an `SEV` instruction has been executed. Software cannot access this register directly.

See the *Cortex™-M3/M4 Instruction Set Technical User’s Manual* for more information.

#### 2.7.1.3 Sleep-on-Exit

If the SLEEP EXIT bit of the **SYSCTRL** register is set, when the processor completes the execution of all exception handlers, it returns to Thread mode and immediately enters sleep mode. This mechanism can be used in applications that only require the processor to run when an exception occurs.

### 2.7.2 Wake Up from Sleep Mode

The conditions for the processor to wake up depend on the mechanism that cause it to enter sleep mode.

### 2.7.2.1 Wake Up from WFI or Sleep-on-Exit

Normally, the processor wakes up only when the NVIC detects an exception with sufficient priority to cause exception entry. Some embedded systems might have to execute system restore tasks after the processor wakes up and before executing an interrupt handler. Entry to the interrupt handler can be delayed by setting the PRIMASK bit and clearing the FAULTMASK bit. If an interrupt arrives that is enabled and has a higher priority than current exception priority, the processor wakes up but does not execute the interrupt handler until the processor clears PRIMASK. For more information about **PRIMASK** and **FAULTMASK**, see page 59 and page 60.

### 2.7.2.2 Wake Up from WFE

The processor wakes up if it detects an exception with sufficient priority to cause exception entry.

In addition, if the SEVONPEND bit in the **SYSCTRL** register is set, any new pending interrupt triggers an event and wakes up the processor, even if the interrupt is disabled or has insufficient priority to cause exception entry. For more information about **SYSCTRL**, see page 115.

## 2.8 Instruction Set Summary

The processor implements a version of the Thumb instruction set. Table 2-13 on page 82 lists the supported instructions.

**Note:** In Table 2-13 on page 82:

- Angle brackets, <>, enclose alternative forms of the operand
- Braces, {}, enclose optional operands
- The Operands column is not exhaustive
- Op2 is a flexible second operand that can be either a register or a constant
- Most instructions can use an optional condition code suffix

For more information on the instructions and operands, see the instruction descriptions in the *Cortex™-M3/M4 Instruction Set Technical User's Manual*.

**Table 2-13. Cortex-M3 Instruction Summary**

Mnemonic	Operands	Brief Description	Flags
ADC, ADCS	{Rd, } Rn, Op2	Add with carry	N,Z,C,V
ADD, ADDS	{Rd, } Rn, Op2	Add	N,Z,C,V
ADD, ADDW	{Rd, } Rn , #imm12	Add	N,Z,C,V
ADR	Rd, label	Load PC-relative address	-
AND, ANDS	{Rd, } Rn, Op2	Logical AND	N,Z,C
ASR, ASRS	Rd, Rm, <Rs #n>	Arithmetic shift right	N,Z,C
B	label	Branch	-
BFC	Rd, #lsb, #width	Bit field clear	-
BFI	Rd, Rn, #lsb, #width	Bit field insert	-
BIC, BICS	{Rd, } Rn, Op2	Bit clear	N,Z,C
BKPT	#imm	Breakpoint	-
BL	label	Branch with link	-
BLX	Rm	Branch indirect with link	-
BX	Rm	Branch indirect	-
CBNZ	Rn, label	Compare and branch if non-zero	-

**Table 2-13. Cortex-M3 Instruction Summary (continued)**

Mnemonic	Operands	Brief Description	Flags
CBZ	Rn, label	Compare and branch if zero	-
CLREX	-	Clear exclusive	-
CLZ	Rd, Rm	Count leading zeros	-
CMN	Rn, Op2	Compare negative	N, Z, C, V
CMP	Rn, Op2	Compare	N, Z, C, V
CPSID	i	Change processor state, disable interrupts	-
CPSIE	i	Change processor state, enable interrupts	-
DMB	-	Data memory barrier	-
DSB	-	Data synchronization barrier	-
EOR, EORS	{Rd, } Rn, Op2	Exclusive OR	N, Z, C
ISB	-	Instruction synchronization barrier	-
IT	-	If-Then condition block	-
LDM	Rn{ ! }, reglist	Load multiple registers, increment after	-
LDMDB, LDMEA	Rn{ ! }, reglist	Load multiple registers, decrement before	-
LDMFD, LDMIA	Rn{ ! }, reglist	Load multiple registers, increment after	-
LDR	Rt, [Rn, #offset]	Load register with word	-
LDRB, LDRBT	Rt, [Rn, #offset]	Load register with byte	-
LDRD	Rt, Rt2, [Rn, #offset]	Load register with two bytes	-
LDREX	Rt, [Rn, #offset]	Load register exclusive	-
LDREXB	Rt, [Rn]	Load register exclusive with byte	-
LDREXH	Rt, [Rn]	Load register exclusive with halfword	-
LDRH, LDRHT	Rt, [Rn, #offset]	Load register with halfword	-
LDRSB, LDRSBT	Rt, [Rn, #offset]	Load register with signed byte	-
LDRSH, LDRSHT	Rt, [Rn, #offset]	Load register with signed halfword	-
LDRT	Rt, [Rn, #offset]	Load register with word	-
LSL, LSLS	Rd, Rm, <Rs #n>	Logical shift left	N, Z, C
LSR, LSRS	Rd, Rm, <Rs #n>	Logical shift right	N, Z, C
MLA	Rd, Rn, Rm, Ra	Multiply with accumulate, 32-bit result	-
MLS	Rd, Rn, Rm, Ra	Multiply and subtract, 32-bit result	-
MOV, MOVS	Rd, Op2	Move	N, Z, C
MOV, MOVW	Rd, #imm16	Move 16-bit constant	N, Z, C
MOVT	Rd, #imm16	Move top	-
MRS	Rd, spec_reg	Move from special register to general register	-
MSR	spec_reg, Rm	Move from general register to special register	N, Z, C, V
MUL, MULS	{Rd, } Rn, Rm	Multiply, 32-bit result	N, Z
MVN, MVNS	Rd, Op2	Move NOT	N, Z, C
NOP	-	No operation	-
ORN, ORNS	{Rd, } Rn, Op2	Logical OR NOT	N, Z, C

**Table 2-13. Cortex-M3 Instruction Summary (continued)**

Mnemonic	Operands	Brief Description	Flags
ORR, ORRS	{Rd, } Rn, Op2	Logical OR	N,Z,C
POP	reglist	Pop registers from stack	-
PUSH	reglist	Push registers onto stack	-
RBIT	Rd, Rn	Reverse bits	-
REV	Rd, Rn	Reverse byte order in a word	-
REV16	Rd, Rn	Reverse byte order in each halfword	-
REVSH	Rd, Rn	Reverse byte order in bottom halfword and sign extend	-
ROR, RORS	Rd, Rm, <Rs #n>	Rotate right	N,Z,C
RRX, RRXS	Rd, Rm	Rotate right with extend	N,Z,C
RSB, RSBS	{Rd, } Rn, Op2	Reverse subtract	N,Z,C,V
SBC, SBCS	{Rd, } Rn, Op2	Subtract with carry	N,Z,C,V
SBFX	Rd, Rn, #lsb, #width	Signed bit field extract	-
SDIV	{Rd, } Rn, Rm	Signed divide	-
SEV	-	Send event	-
SMLAL	RdLo, RdHi, Rn, Rm	Signed multiply with accumulate (32x32+64), 64-bit result	-
SMULL	RdLo, RdHi, Rn, Rm	Signed multiply (32x32), 64-bit result	-
SSAT	Rd, #n, Rm {,shift #s}	Signed saturate	Q
STM	Rn{!}, reglist	Store multiple registers, increment after	-
STMDB, STMEA	Rn{!}, reglist	Store multiple registers, decrement before	-
STMFD, STMIA	Rn{!}, reglist	Store multiple registers, increment after	-
STR	Rt, [Rn {, #offset}]	Store register word	-
STRB, STRBT	Rt, [Rn {, #offset}]	Store register byte	-
STRD	Rt, Rt2, [Rn {, #offset}]	Store register two words	-
STREX	Rt, Rt, [Rn {, #offset}]	Store register exclusive	-
STREXB	Rd, Rt, [Rn]	Store register exclusive byte	-
STREXH	Rd, Rt, [Rn]	Store register exclusive halfword	-
STRH, STRHT	Rt, [Rn {, #offset}]	Store register halfword	-
STRSB, STRSBT	Rt, [Rn {, #offset}]	Store register signed byte	-
STRSH, STRSHT	Rt, [Rn {, #offset}]	Store register signed halfword	-
STRT	Rt, [Rn {, #offset}]	Store register word	-
SUB, SUBS	{Rd, } Rn, Op2	Subtract	N,Z,C,V
SUB, SUBW	{Rd, } Rn, #imm12	Subtract 12-bit constant	N,Z,C,V
SVC	#imm	Supervisor call	-
SXTB	{Rd, } Rm {,ROR #n}	Sign extend a byte	-
SXTH	{Rd, } Rm {,ROR #n}	Sign extend a halfword	-
TBB	[Rn, Rm]	Table branch byte	-
TBH	[Rn, Rm, LSL #1]	Table branch halfword	-
TEQ	Rn, Op2	Test equivalence	N,Z,C
TST	Rn, Op2	Test	N,Z,C

**Table 2-13. Cortex-M3 Instruction Summary (continued)**

Mnemonic	Operands	Brief Description	Flags
UBFX	Rd, Rn, #lsb, #width	Unsigned bit field extract	-
UDIV	{Rd, } Rn, Rm	Unsigned divide	-
UMLAL	RdLo, RdHi, Rn, Rm	Unsigned multiply with accumulate (32x32+32+32), 64-bit result	-
UMULL	RdLo, RdHi, Rn, Rm	Unsigned multiply (32x 2), 64-bit result	-
USAT	Rd, #n, Rm {,shift #s}	Unsigned Saturate	Q
UXTB	{Rd, } Rm, {,ROR #n}	Zero extend a Byte	-
UXTH	{Rd, } Rm, {,ROR #n}	Zero extend a Halfword	-
WFE	-	Wait for event	-
WFI	-	Wait for interrupt	-

## 3 Cortex-M3 Peripherals

This chapter provides information on the Stellaris® implementation of the Cortex-M3 processor peripherals, including:

- SysTick (see page 86)
 

Provides a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism.
- Nested Vectored Interrupt Controller (NVIC) (see page 87)
  - Facilitates low-latency exception and interrupt handling
  - Controls power management
  - Implements system control registers
- System Control Block (SCB) (see page 89)
 

Provides system implementation information and system control, including configuration, control, and reporting of system exceptions.
- Memory Protection Unit (MPU) (see page 89)
 

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.

Table 3-1 on page 86 shows the address map of the Private Peripheral Bus (PPB). Some peripheral register regions are split into two address regions, as indicated by two addresses listed.

**Table 3-1. Core Peripheral Register Regions**

Address	Core Peripheral	Description (see page ...)
0xE000.E010-0xE000.E01F	System Timer	86
0xE000.E100-0xE000.E4EF	Nested Vectored Interrupt Controller	87
0xE000.EF00-0xE000.EF03		
0xE000.ED00-0xE000.ED3F	System Control Block	89
0xE000.ED90-0xE000.EDB8	Memory Protection Unit	89

### 3.1 Functional Description

This chapter provides information on the Stellaris implementation of the Cortex-M3 processor peripherals: SysTick, NVIC, SCB and MPU.

#### 3.1.1 System Timer (SysTick)

Cortex-M3 includes an integrated system timer, SysTick, which 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 as:

- An RTOS tick timer that 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 used to measure time to completion and time used.
- An internal clock source control based on missing/meeting durations. The COUNT bit in the **STCTRL** 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.

The timer consists of three registers:

- **SysTick Control and Status (STCTRL)**: A control and status counter to configure its clock, enable the counter, enable the SysTick interrupt, and determine counter status.
- **SysTick Reload Value (STRELOAD)**: The reload value for the counter, used to provide the counter's wrap value.
- **SysTick Current Value (STCURRENT)**: The current value of the counter.

When enabled, the timer counts down on each clock from the reload value to zero, reloads (wraps) to the value in the **STRELOAD** register on the next clock edge, then decrements on subsequent clocks. Clearing the **STRELOAD** register disables the counter on the next wrap. When the counter reaches zero, the COUNT status bit is set. The COUNT bit clears on reads.

Writing to the **STCURRENT** register clears the register and the COUNT 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.

The SysTick counter runs on the system clock. If this clock signal is stopped for low power mode, the SysTick counter stops. Ensure software uses aligned word accesses to access the SysTick registers.

**Note:** When the processor is halted for debugging, the counter does not decrement.

### 3.1.2 Nested Vectored Interrupt Controller (NVIC)

This section describes the Nested Vectored Interrupt Controller (NVIC) and the registers it uses. The NVIC supports:

- 26 interrupts.
- A programmable priority level of 0-7 for each interrupt. A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority.
- Low-latency exception and interrupt handling.
- Level and pulse detection of interrupt signals.
- Dynamic reprioritization of interrupts.
- Grouping of priority values into group priority and subpriority fields.
- Interrupt tail-chaining.
- An external Non-maskable interrupt (NMI).

The processor automatically stacks its state on exception entry and unstacks this state on exception exit, with no instruction overhead, providing low latency exception handling.

### 3.1.2.1 Level-Sensitive and Pulse Interrupts

The processor supports both level-sensitive and pulse interrupts. Pulse interrupts are also described as edge-triggered interrupts.

A level-sensitive interrupt is held asserted until the peripheral deasserts the interrupt signal. Typically this happens because the ISR accesses the peripheral, causing it to clear the interrupt request. A pulse interrupt is an interrupt signal sampled synchronously on the rising edge of the processor clock. To ensure the NVIC detects the interrupt, the peripheral must assert the interrupt signal for at least one clock cycle, during which the NVIC detects the pulse and latches the interrupt.

When the processor enters the ISR, it automatically removes the pending state from the interrupt (see “Hardware and Software Control of Interrupts” on page 88 for more information). For a level-sensitive interrupt, if the signal is not deasserted before the processor returns from the ISR, the interrupt becomes pending again, and the processor must execute its ISR again. As a result, the peripheral can hold the interrupt signal asserted until it no longer needs servicing.

### 3.1.2.2 Hardware and Software Control of Interrupts

The Cortex-M3 latches all interrupts. A peripheral interrupt becomes pending for one of the following reasons:

- The NVIC detects that the interrupt signal is High and the interrupt is not active.
- The NVIC detects a rising edge on the interrupt signal.
- Software writes to the corresponding interrupt set-pending register bit, or to the **Software Trigger Interrupt (SWTRIG)** register to make a Software-Generated Interrupt pending. See the `INT` bit in the **PEND0** register on page 102 or **SWTRIG** on page 107.

A pending interrupt remains pending until one of the following:

- The processor enters the ISR for the interrupt, changing the state of the interrupt from pending to active. Then:
  - For a level-sensitive interrupt, when the processor returns from the ISR, the NVIC samples the interrupt signal. If the signal is asserted, the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR. Otherwise, the state of the interrupt changes to inactive.
  - For a pulse interrupt, the NVIC continues to monitor the interrupt signal, and if this is pulsed the state of the interrupt changes to pending and active. In this case, when the processor returns from the ISR the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR.

If the interrupt signal is not pulsed while the processor is in the ISR, when the processor returns from the ISR the state of the interrupt changes to inactive.

- Software writes to the corresponding interrupt clear-pending register bit
  - For a level-sensitive interrupt, if the interrupt signal is still asserted, the state of the interrupt does not change. Otherwise, the state of the interrupt changes to inactive.

- For a pulse interrupt, the state of the interrupt changes to inactive, if the state was pending or to active, if the state was active and pending.

### 3.1.3 System Control Block (SCB)

The System Control Block (SCB) provides system implementation information and system control, including configuration, control, and reporting of the system exceptions.

### 3.1.4 Memory Protection Unit (MPU)

This section describes the Memory protection unit (MPU). The MPU divides the memory map into a number of regions and defines the location, size, access permissions, and memory attributes of each region. The MPU supports independent attribute settings for each region, overlapping regions, and export of memory attributes to the system.

The memory attributes affect the behavior of memory accesses to the region. The Cortex-M3 MPU defines eight separate memory regions, 0-7, and a background region.

When memory regions overlap, a memory access is affected by the attributes of the region with the highest number. For example, the attributes for region 7 take precedence over the attributes of any region that overlaps region 7.

The background region has the same memory access attributes as the default memory map, but is accessible from privileged software only.

The Cortex-M3 MPU memory map is unified, meaning that instruction accesses and data accesses have the same region settings.

If a program accesses a memory location that is prohibited by the MPU, the processor generates a memory management fault, causing a fault exception and possibly causing termination of the process in an OS environment. In an OS environment, the kernel can update the MPU region setting dynamically based on the process to be executed. Typically, an embedded OS uses the MPU for memory protection.

Configuration of MPU regions is based on memory types (see “Memory Regions, Types and Attributes” on page 64 for more information).

Table 3-2 on page 89 shows the possible MPU region attributes. See the section called “MPU Configuration for a Stellaris Microcontroller” on page 93 for guidelines for programming a microcontroller implementation.

**Table 3-2. Memory Attributes Summary**

Memory Type	Description
Strongly Ordered	All accesses to Strongly Ordered memory occur in program order.
Device	Memory-mapped peripherals
Normal	Normal memory

To avoid unexpected behavior, disable the interrupts before updating the attributes of a region that the interrupt handlers might access.

Ensure software uses aligned accesses of the correct size to access MPU registers:

- Except for the **MPU Region Attribute and Size (MPUATTR)** register, all MPU registers must be accessed with aligned word accesses.
- The **MPUATTR** register can be accessed with byte or aligned halfword or word accesses.

The processor does not support unaligned accesses to MPU registers.

When setting up the MPU, and if the MPU has previously been programmed, disable unused regions to prevent any previous region settings from affecting the new MPU setup.

### 3.1.4.1 Updating an MPU Region

To update the attributes for an MPU region, the **MPU Region Number (MPUNUMBER)**, **MPU Region Base Address (MPUBASE)** and **MPUATTR** registers must be updated. Each register can be programmed separately or with a multiple-word write to program all of these registers. You can use the **MPUBASEEx** and **MPUATTRx** aliases to program up to four regions simultaneously using an STM instruction.

#### *Updating an MPU Region Using Separate Words*

This example simple code configures one region:

```
; R1 = region number
; R2 = size/enable
; R3 = attributes
; R4 = address
LDR R0,=MPUNUMBER      ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0]       ; Region Number
STR R4, [R0, #0x4]       ; Region Base Address
STRH R2, [R0, #0x8]      ; Region Size and Enable
STRH R3, [R0, #0xA]      ; Region Attribute
```

Disable a region before writing new region settings to the MPU if you have previously enabled the region being changed. For example:

```
; R1 = region number
; R2 = size/enable
; R3 = attributes
; R4 = address
LDR R0,=MPUNUMBER      ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0]       ; Region Number
BIC R2, R2, #1          ; Disable
STRH R2, [R0, #0x8]      ; Region Size and Enable
STR R4, [R0, #0x4]       ; Region Base Address
STRH R3, [R0, #0xA]      ; Region Attribute
ORR R2, #1               ; Enable
STRH R2, [R0, #0x8]      ; Region Size and Enable
```

Software must use memory barrier instructions:

- Before MPU setup, if there might be outstanding memory transfers, such as buffered writes, that might be affected by the change in MPU settings.
- After MPU setup, if it includes memory transfers that must use the new MPU settings.

However, memory barrier instructions are not required if the MPU setup process starts by entering an exception handler, or is followed by an exception return, because the exception entry and exception return mechanism cause memory barrier behavior.

Software does not need any memory barrier instructions during MPU setup, because it accesses the MPU through the Private Peripheral Bus (PPB), which is a Strongly Ordered memory region.

For example, if all of the memory access behavior is intended to take effect immediately after the programming sequence, then a `DSB` instruction and an `ISB` instruction should be used. A `DSB` is required after changing MPU settings, such as at the end of context switch. An `ISB` is required if the code that programs the MPU region or regions is entered using a branch or call. If the programming sequence is entered using a return from exception, or by taking an exception, then an `ISB` is not required.

### ***Updating an MPU Region Using Multi-Word Writes***

The MPU can be programmed directly using multi-word writes, depending how the information is divided. Consider the following reprogramming:

```
; R1 = region number
; R2 = address
; R3 = size, attributes in one
LDR R0, =MPUNUMBER ; 0xE000ED98, MPU region number register
STR R1, [R0, #0x0] ; Region Number
STR R2, [R0, #0x4] ; Region Base Address
STR R3, [R0, #0x8] ; Region Attribute, Size and Enable
```

An `STM` instruction can be used to optimize this:

```
; R1 = region number
; R2 = address
; R3 = size, attributes in one
LDR R0, =MPUNUMBER ; 0xE000ED98, MPU region number register
STM R0, {R1-R3} ; Region number, address, attribute, size and enable
```

This operation can be done in two words for pre-packed information, meaning that the **MPU Region Base Address (MPUBASE)** register (see page 139) contains the required region number and has the `VALID` bit set. This method can be used when the data is statically packed, for example in a boot loader:

```
; R1 = address and region number in one
; R2 = size and attributes in one
LDR R0, =MPUBASE ; 0xE000ED9C, MPU Region Base register
STR R1, [R0, #0x0] ; Region base address and region number combined
; with VALID (bit 4) set
STR R2, [R0, #0x4] ; Region Attribute, Size and Enable
```

### ***Subregions***

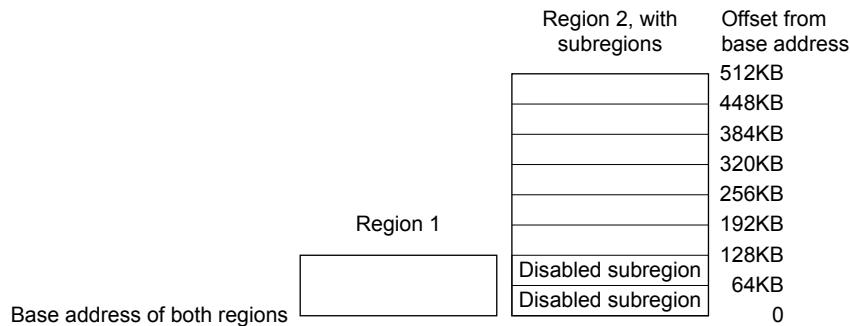
Regions of 256 bytes or more are divided into eight equal-sized subregions. Set the corresponding bit in the `SRD` field of the **MPU Region Attribute and Size (MPUATTR)** register (see page 141) to disable a subregion. The least-significant bit of the `SRD` field controls the first subregion, and the most-significant bit controls the last subregion. Disabling a subregion means another region overlapping the disabled range matches instead. If no other enabled region overlaps the disabled subregion, the MPU issues a fault.

Regions of 32, 64, and 128 bytes do not support subregions. With regions of these sizes, the `SRD` field must be configured to `0x00`, otherwise the MPU behavior is unpredictable.

### Example of SRD Use

Two regions with the same base address overlap. Region one is 128 KB, and region two is 512 KB. To ensure the attributes from region one apply to the first 128 KB region, configure the SRD field for region two to 0x03 to disable the first two subregions, as Figure 3-1 on page 92 shows.

**Figure 3-1. SRD Use Example**



#### 3.1.4.2 MPU Access Permission Attributes

The access permission bits, TEX, S, C, B, AP, and XN of the **MPUATTR** register, control access to the corresponding memory region. If an access is made to an area of memory without the required permissions, then the MPU generates a permission fault.

Table 3-3 on page 92 shows the encodings for the TEX, C, B, and S access permission bits. All encodings are shown for completeness, however the current implementation of the Cortex-M3 does not support the concept of cacheability or shareability. Refer to the section called “MPU Configuration for a Stellaris Microcontroller” on page 93 for information on programming the MPU for Stellaris implementations.

**Table 3-3. TEX, S, C, and B Bit Field Encoding**

TEX	S	C	B	Memory Type	Shareability	Other Attributes
000b	x <sup>a</sup>	0	0	Strongly Ordered	Shareable	-
000	x <sup>a</sup>	0	1	Device	Shareable	-
000	0	1	0	Normal	Not shareable	
000	1	1	0	Normal	Shareable	Outer and inner write-through. No write allocate.
000	0	1	1	Normal	Not shareable	
000	1	1	1	Normal	Shareable	
001	0	0	0	Normal	Not shareable	Outer and inner noncacheable.
001	1	0	0	Normal	Shareable	
001	x <sup>a</sup>	0	1	Reserved encoding	-	-
001	x <sup>a</sup>	1	0	Reserved encoding	-	-
001	0	1	1	Normal	Not shareable	Outer and inner write-back. Write and read allocate.
001	1	1	1	Normal	Shareable	
010	x <sup>a</sup>	0	0	Device	Not shareable	Nonshared Device.
010	x <sup>a</sup>	0	1	Reserved encoding	-	-
010	x <sup>a</sup>	1	x <sup>a</sup>	Reserved encoding	-	-

**Table 3-3. TEX, S, C, and B Bit Field Encoding (continued)**

<b>TEX</b>	<b>S</b>	<b>C</b>	<b>B</b>	<b>Memory Type</b>	<b>Shareability</b>	<b>Other Attributes</b>
1BB	0	A	A	Normal	Not shareable	Cached memory (BB = outer policy, AA = inner policy).
1BB	1	A	A	Normal	Shareable	See Table 3-4 for the encoding of the AA and BB bits.

a. The MPU ignores the value of this bit.

Table 3-4 on page 93 shows the cache policy for memory attribute encodings with a **TEX** value in the range of 0x4-0x7.

**Table 3-4. Cache Policy for Memory Attribute Encoding**

<b>Encoding, AA or BB</b>	<b>Corresponding Cache Policy</b>
00	Non-cacheable
01	Write back, write and read allocate
10	Write through, no write allocate
11	Write back, no write allocate

Table 3-5 on page 93 shows the **AP** encodings in the **MPUATTR** register that define the access permissions for privileged and unprivileged software.

**Table 3-5. AP Bit Field Encoding**

<b>AP Bit Field</b>	<b>Privileged Permissions</b>	<b>Unprivileged Permissions</b>	<b>Description</b>
000	No access	No access	All accesses generate a permission fault.
001	R/W	No access	Access from privileged software only.
010	R/W	RO	Writes by unprivileged software generate a permission fault.
011	R/W	R/W	Full access.
100	Unpredictable	Unpredictable	Reserved.
101	RO	No access	Reads by privileged software only.
110	RO	RO	Read-only, by privileged or unprivileged software.
111	RO	RO	Read-only, by privileged or unprivileged software.

#### **MPU Configuration for a Stellaris Microcontroller**

Stellaris microcontrollers have only a single processor and no caches. As a result, the MPU should be programmed as shown in Table 3-6 on page 93.

**Table 3-6. Memory Region Attributes for Stellaris Microcontrollers**

<b>Memory Region</b>	<b>TEX</b>	<b>S</b>	<b>C</b>	<b>B</b>	<b>Memory Type and Attributes</b>
Flash memory	000b	0	1	0	Normal memory, non-shareable, write-through
Internal SRAM	000b	1	1	0	Normal memory, shareable, write-through
External SRAM	000b	1	1	1	Normal memory, shareable, write-back, write-allocate
Peripherals	000b	1	0	1	Device memory, shareable

In current Stellaris microcontroller implementations, the shareability and cache policy attributes do not affect the system behavior. However, using these settings for the MPU regions can make the application code more portable. The values given are for typical situations.

### 3.1.4.3 MPU Mismatch

When an access violates the MPU permissions, the processor generates a memory management fault (see “Exceptions and Interrupts” on page 63 for more information). The **MFAULTSTAT** register indicates the cause of the fault. See page 126 for more information.

## 3.2 Register Map

Table 3-7 on page 94 lists the Cortex-M3 Peripheral SysTick, NVIC, MPU and SCB registers. The offset listed is a hexadecimal increment to the register's address, relative to the Core Peripherals base address of 0xE000.E000.

**Note:** Register spaces that are not used are reserved for future or internal use. Software should not modify any reserved memory address.

**Table 3-7. Peripherals Register Map**

Offset	Name	Type	Reset	Description	See page
<b>System Timer (SysTick) Registers</b>					
0x010	STCTRL	R/W	0x0000.0000	SysTick Control and Status Register	96
0x014	STRELOAD	R/W	0x0000.0000	SysTick Reload Value Register	98
0x018	STCURRENT	R/WC	0x0000.0000	SysTick Current Value Register	99
<b>Nested Vectored Interrupt Controller (NVIC) Registers</b>					
0x100	EN0	R/W	0x0000.0000	Interrupt 0-29 Set Enable	100
0x180	DIS0	R/W	0x0000.0000	Interrupt 0-29 Clear Enable	101
0x200	PEND0	R/W	0x0000.0000	Interrupt 0-29 Set Pending	102
0x280	UNPEND0	R/W	0x0000.0000	Interrupt 0-29 Clear Pending	103
0x300	ACTIVE0	RO	0x0000.0000	Interrupt 0-29 Active Bit	104
0x400	PRI0	R/W	0x0000.0000	Interrupt 0-3 Priority	105
0x404	PRI1	R/W	0x0000.0000	Interrupt 4-7 Priority	105
0x408	PRI2	R/W	0x0000.0000	Interrupt 8-11 Priority	105
0x40C	PRI3	R/W	0x0000.0000	Interrupt 12-15 Priority	105
0x410	PRI4	R/W	0x0000.0000	Interrupt 16-19 Priority	105
0x414	PRI5	R/W	0x0000.0000	Interrupt 20-23 Priority	105
0x418	PRI6	R/W	0x0000.0000	Interrupt 24-27 Priority	105
0x41C	PRI7	R/W	0x0000.0000	Interrupt 28-29 Priority	105
0xF00	SWTRIG	WO	0x0000.0000	Software Trigger Interrupt	107

**Table 3-7. Peripherals Register Map (continued)**

Offset	Name	Type	Reset	Description	See page
<b>System Control Block (SCB) Registers</b>					
0xD00	CPUID	RO	0x410F.C231	CPU ID Base	108
0xD04	INTCTRL	R/W	0x0000.0000	Interrupt Control and State	109
0xD08	VTABLE	R/W	0x0000.0000	Vector Table Offset	112
0xD0C	APINT	R/W	0xFA05.0000	Application Interrupt and Reset Control	113
0xD10	SYSCTRL	R/W	0x0000.0000	System Control	115
0xD14	CFGCTRL	R/W	0x0000.0000	Configuration and Control	117
0xD18	SYSPRI1	R/W	0x0000.0000	System Handler Priority 1	119
0xD1C	SYSPRI2	R/W	0x0000.0000	System Handler Priority 2	120
0xD20	SYSPRI3	R/W	0x0000.0000	System Handler Priority 3	121
0xD24	SYSHNDCTRL	R/W	0x0000.0000	System Handler Control and State	122
0xD28	FAULTSTAT	R/W1C	0x0000.0000	Configurable Fault Status	126
0xD2C	HFAULTSTAT	R/W1C	0x0000.0000	Hard Fault Status	132
0xD34	MMADDR	R/W	-	Memory Management Fault Address	133
0xD38	FAULTADDR	R/W	-	Bus Fault Address	134
<b>Memory Protection Unit (MPU) Registers</b>					
0xD90	MPUTYPE	RO	0x0000.0800	MPU Type	135
0xD94	MPUCTRL	R/W	0x0000.0000	MPU Control	136
0xD98	MPUNUMBER	R/W	0x0000.0000	MPU Region Number	138
0xD9C	MPUBASE	R/W	0x0000.0000	MPU Region Base Address	139
0xDA0	MPUATTR	R/W	0x0000.0000	MPU Region Attribute and Size	141
0xDA4	MPUBASE1	R/W	0x0000.0000	MPU Region Base Address Alias 1	139
0xDA8	MPUATTR1	R/W	0x0000.0000	MPU Region Attribute and Size Alias 1	141
0xDAC	MPUBASE2	R/W	0x0000.0000	MPU Region Base Address Alias 2	139
0xDB0	MPUATTR2	R/W	0x0000.0000	MPU Region Attribute and Size Alias 2	141
0xDB4	MPUBASE3	R/W	0x0000.0000	MPU Region Base Address Alias 3	139
0xDB8	MPUATTR3	R/W	0x0000.0000	MPU Region Attribute and Size Alias 3	141

### 3.3 System Timer (SysTick) Register Descriptions

This section lists and describes the System Timer registers, in numerical order by address offset.

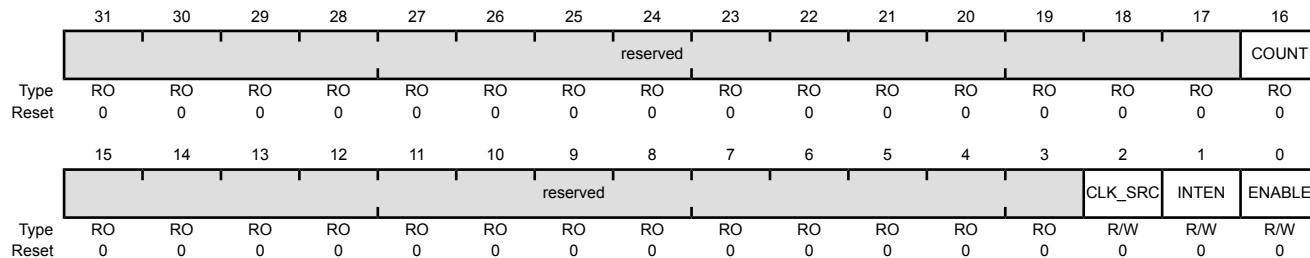
## Register 1: SysTick Control and Status Register (STCTRL), offset 0x010

**Note:** This register can only be accessed from privileged mode.

The SysTick **STCTRL** register enables the SysTick features.

### SysTick Control and Status Register (STCTRL)

Base 0xE000.E000  
Offset 0x010  
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:17	reserved	RO	0x000	Software should not rely on the value of 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	COUNT	RO	0	<p>Count Flag</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>The SysTick timer has not counted to 0 since the last time this bit was read.</td> </tr> <tr> <td>1</td> <td>The SysTick timer has counted to 0 since the last time this bit was read.</td> </tr> </table> <p>This bit is cleared by a read of the register or if the <b>STCURRENT</b> register is written with any value. If read by the debugger using the DAP, this bit is cleared only if the <b>MasterType</b> bit in the <b>AHB-AP Control Register</b> is clear. Otherwise, the COUNT bit is not changed by the debugger read. See the <i>ARM® Debug Interface V5 Architecture Specification</i> for more information on <b>MasterType</b>.</p>	Value	Description	0	The SysTick timer has not counted to 0 since the last time this bit was read.	1	The SysTick timer has counted to 0 since the last time this bit was read.
Value	Description									
0	The SysTick timer has not counted to 0 since the last time this bit was read.									
1	The SysTick timer has counted to 0 since the last time this bit was read.									
15:3	reserved	RO	0x000	Software should not rely on the value of 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	CLK_SRC	R/W	0	<p>Clock Source</p> <table> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>External reference clock. (Not implemented for most Stellaris microcontrollers.)</td> </tr> <tr> <td>1</td> <td>System clock</td> </tr> </table> <p>Because an external reference clock is not implemented, this bit must be set in order for SysTick to operate.</p>	Value	Description	0	External reference clock. (Not implemented for most Stellaris microcontrollers.)	1	System clock
Value	Description									
0	External reference clock. (Not implemented for most Stellaris microcontrollers.)									
1	System clock									

Bit/Field	Name	Type	Reset	Description
1	INTEN	R/W	0	Interrupt Enable
				Value      Description
			0	Interrupt generation is disabled. Software can use the COUNT bit to determine if the counter has ever reached 0.
			1	An interrupt is generated to the NVIC when SysTick counts to 0.
0	ENABLE	R/W	0	Enable
				Value      Description
			0	The counter is disabled.
			1	Enables SysTick to operate in a multi-shot way. That is, the counter loads the RELOAD value and begins counting down. On reaching 0, the COUNT bit is set and an interrupt is generated if enabled by INTEN. The counter then loads the RELOAD value again and begins counting.

## Register 2: SysTick Reload Value Register (STRELOAD), offset 0x014

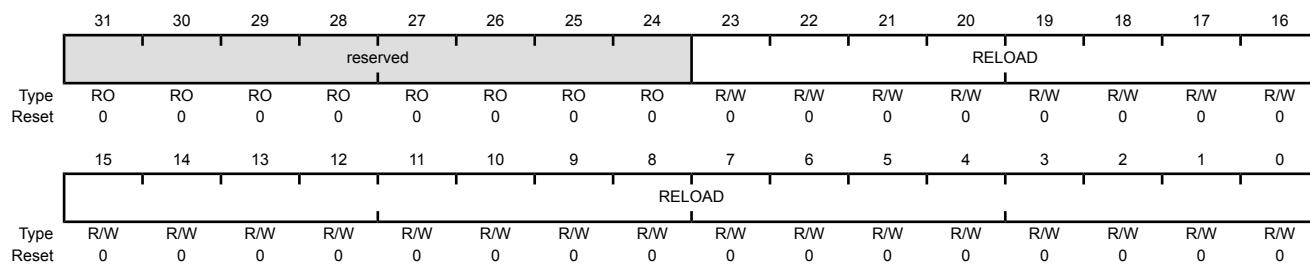
**Note:** This register can only be accessed from privileged mode.

The **STRELOAD** register specifies the start value to load into the **SysTick Current Value (STCURRENT)** register when the counter reaches 0. The start value can be between 0x1 and 0x00FF.FFFF. A start value of 0 is possible but has no effect because the SysTick interrupt and the COUNT bit are activated when counting from 1 to 0.

SysTick can be configured as a multi-shot timer, repeated over and over, firing every N+1 clock pulses, where N is any value from 1 to 0x00FF.FFFF. For example, if a tick interrupt is required every 100 clock pulses, 99 must be written into the RELOAD field.

### SysTick Reload Value Register (STRELOAD)

Base 0xE000.E000  
Offset 0x014  
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:24	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.
23:0	RELOAD	R/W	0x00.0000	Reload Value Value to load into the <b>SysTick Current Value (STCURRENT)</b> register when the counter reaches 0.

## Register 3: SysTick Current Value Register (STCURRENT), offset 0x018

**Note:** This register can only be accessed from privileged mode.

The **STCURRENT** register contains the current value of the SysTick counter.

### SysTick Current Value Register (STCURRENT)

Base 0xE000.E000  
Offset 0x018  
Type R/WC, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved																
Type	RO	R/WC														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CURRENT																
Type	R/WC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:24	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.
23:0	CURRENT	R/WC	0x00.0000	<p>Current Value</p> <p>This field contains the current value at the time the register is accessed. No read-modify-write protection is provided, so change with care.</p> <p>This register is write-clear. Writing to it with any value clears the register. Clearing this register also clears the COUNT bit of the <b>STCTRL</b> register.</p>

## 3.4 NVIC Register Descriptions

This section lists and describes the NVIC registers, in numerical order by address offset.

The NVIC registers can only be fully accessed from privileged mode, but interrupts can be pended while in unprivileged mode by enabling the **Configuration and Control (CFGCTRL)** register. Any other unprivileged mode access causes a bus fault.

Ensure software uses correctly aligned register accesses. The processor does not support unaligned accesses to NVIC registers.

An interrupt can enter the pending state even if it is disabled.

Before programming the **VTABLE** register to relocate the vector table, ensure the vector table entries of the new vector table are set up for fault handlers, NMI, and all enabled exceptions such as interrupts. For more information, see page 112.

## Register 4: Interrupt 0-29 Set Enable (EN0), offset 0x100

**Note:** This register can only be accessed from privileged mode.

The **EN0** register enables interrupts and shows which interrupts are enabled. Bit 0 corresponds to Interrupt 0; bit 29 corresponds to Interrupt 29.

See Table 2-9 on page 74 for interrupt assignments.

If a pending interrupt is enabled, the NVIC activates the interrupt based on its priority. If an interrupt is not enabled, asserting its interrupt signal changes the interrupt state to pending, but the NVIC never activates the interrupt, regardless of its priority.

### Interrupt 0-29 Set Enable (EN0)

Base 0xE000.E000  
Offset 0x100  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INT																
Type	RO	RO	R/W													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
INT																
Type	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
31:30	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.
29:0	INT	R/W	0x0000.0000	Interrupt Enable
				Value Description
			0	On a read, indicates the interrupt is disabled. On a write, no effect.
			1	On a read, indicates the interrupt is enabled. On a write, enables the interrupt.

A bit can only be cleared by setting the corresponding `INT[n]` bit in the **DISn** register.

## Register 5: Interrupt 0-29 Clear Enable (DIS0), offset 0x180

**Note:** This register can only be accessed from privileged mode.

The **DIS0** register disables interrupts. Bit 0 corresponds to Interrupt 0; bit 29 corresponds to Interrupt 29.

See Table 2-9 on page 74 for interrupt assignments.

### Interrupt 0-29 Clear Enable (DIS0)

Base 0xE000.E000

Offset 0x180

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	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
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
31:30	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.
29:0	INT	R/W	0x0000.0000	Interrupt Disable
				Value Description
				0 On a read, indicates the interrupt is disabled. On a write, no effect.
				1 On a read, indicates the interrupt is enabled. On a write, clears the corresponding INT[n] bit in the EN0 register, disabling interrupt [n].

## Register 6: Interrupt 0-29 Set Pending (PEND0), offset 0x200

**Note:** This register can only be accessed from privileged mode.

The **PEND0** register forces interrupts into the pending state and shows which interrupts are pending. Bit 0 corresponds to Interrupt 0; bit 29 corresponds to Interrupt 29.

See Table 2-9 on page 74 for interrupt assignments.

### Interrupt 0-29 Set Pending (PEND0)

Base 0xE000.E000  
Offset 0x200  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INT																
Type	RO	RO	R/W													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
INT																
Type	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						
31:30	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.						
29:0	INT	R/W	0x0000.0000	Interrupt Set Pending						
				<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>On a read, indicates that the interrupt is not pending. On a write, no effect.</td></tr> <tr> <td>1</td> <td>On a read, indicates that the interrupt is pending. On a write, the corresponding interrupt is set to pending even if it is disabled.</td></tr> </tbody> </table>	Value	Description	0	On a read, indicates that the interrupt is not pending. On a write, no effect.	1	On a read, indicates that the interrupt is pending. On a write, the corresponding interrupt is set to pending even if it is disabled.
Value	Description									
0	On a read, indicates that the interrupt is not pending. On a write, no effect.									
1	On a read, indicates that the interrupt is pending. On a write, the corresponding interrupt is set to pending even if it is disabled.									

If the corresponding interrupt is already pending, setting a bit has no effect.

A bit can only be cleared by setting the corresponding `INT[n]` bit in the **UNPEND0** register.

## Register 7: Interrupt 0-29 Clear Pending (UNPEND0), offset 0x280

**Note:** This register can only be accessed from privileged mode.

The **UNPEND0** register shows which interrupts are pending and removes the pending state from interrupts. Bit 0 corresponds to Interrupt 0; bit 29 corresponds to Interrupt 29.

See Table 2-9 on page 74 for interrupt assignments.

### Interrupt 0-29 Clear Pending (UNPEND0)

Base 0xE000.E000

Offset 0x280

Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INT																
Type	RO	RO	R/W													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
INT																
Type	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
31:30	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.
29:0	INT	R/W	0x0000.0000	Interrupt Clear Pending
		Value	Description	
		0	On a read, indicates that the interrupt is not pending. On a write, no effect.	
		1	On a read, indicates that the interrupt is pending. On a write, clears the corresponding INT[n] bit in the PEND0 register, so that interrupt [n] is no longer pending. Setting a bit does not affect the active state of the corresponding interrupt.	

## Register 8: Interrupt 0-29 Active Bit (ACTIVE0), offset 0x300

**Note:** This register can only be accessed from privileged mode.

The **ACTIVE0** register indicates which interrupts are active. Bit 0 corresponds to Interrupt 0; bit 29 corresponds to Interrupt 29.

See Table 2-9 on page 74 for interrupt assignments.

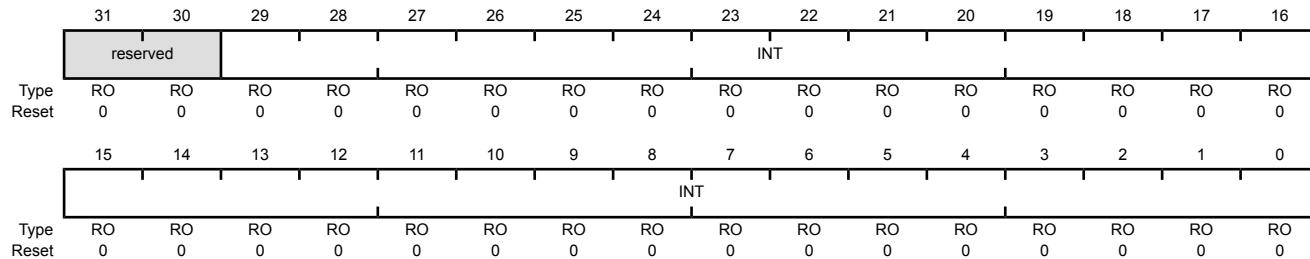
**Caution – Do not manually set or clear the bits in this register.**

### Interrupt 0-29 Active Bit (ACTIVE0)

Base 0xE000.E000

Offset 0x300

Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:30	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.
29:0	INT	RO	0x0000.0000	Interrupt Active
		Value	Description	
		0	The corresponding interrupt is not active.	
		1	The corresponding interrupt is active, or active and pending.	

**Register 9: Interrupt 0-3 Priority (PRI0), offset 0x400****Register 10: Interrupt 4-7 Priority (PRI1), offset 0x404****Register 11: Interrupt 8-11 Priority (PRI2), offset 0x408****Register 12: Interrupt 12-15 Priority (PRI3), offset 0x40C****Register 13: Interrupt 16-19 Priority (PRI4), offset 0x410****Register 14: Interrupt 20-23 Priority (PRI5), offset 0x414****Register 15: Interrupt 24-27 Priority (PRI6), offset 0x418****Register 16: Interrupt 28-29 Priority (PRI7), offset 0x41C**

**Note:** This register can only be accessed from privileged mode.

The **PRI<sub>n</sub>** registers provide 3-bit priority fields for each interrupt. These registers are byte accessible. Each register holds four priority fields that are assigned to interrupts as follows:

PRIn Register Bit Field	Interrupt
Bits 31:29	Interrupt [4n+3]
Bits 23:21	Interrupt [4n+2]
Bits 15:13	Interrupt [4n+1]
Bits 7:5	Interrupt [4n]

See Table 2-9 on page 74 for interrupt assignments.

Each priority level can be split into separate group priority and subpriority fields. The PRIGROUP field in the **Application Interrupt and Reset Control (APINT)** register (see page 113) indicates the position of the binary point that splits the priority and subpriority fields.

These registers can only be accessed from privileged mode.

**Interrupt 0-3 Priority (PRI0)**

Base 0xE000.E000

Offset 0x400

Type R/W, reset 0x0000.0000

Register Structure															
INTD				reserved				INTC				reserved			
Type	R/W	R/W	R/W	RO	RO	RO	RO	R/W	R/W	R/W	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
INTB				reserved				INTA				reserved			
Type	R/W	R/W	R/W	RO	RO	RO	RO	R/W	R/W	R/W	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	INTD	R/W	0x0	Interrupt Priority for Interrupt [4n+3] This field holds a priority value, 0-7, for the interrupt with the number [4n+3], where n is the number of the <b>Interrupt Priority</b> register (n=0 for <b>PRI0</b> , and so on). The lower the value, the greater the priority of the corresponding interrupt.
15:0	INTB	R/W	0x0	Interrupt Priority for Interrupt [4n+2]
15:0	INTC	R/W	0x0	Interrupt Priority for Interrupt [4n+1]
15:0	INTA	R/W	0x0	Interrupt Priority for Interrupt [4n]

Bit/Field	Name	Type	Reset	Description
28:24	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.
23:21	INTC	R/W	0x0	Interrupt Priority for Interrupt [4n+2] This field holds a priority value, 0-7, for the interrupt with the number [4n+2], where n is the number of the <b>Interrupt Priority</b> register (n=0 for <b>PRI0</b> , and so on). The lower the value, the greater the priority of the corresponding interrupt.
20:16	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.
15:13	INTB	R/W	0x0	Interrupt Priority for Interrupt [4n+1] This field holds a priority value, 0-7, for the interrupt with the number [4n+1], where n is the number of the <b>Interrupt Priority</b> register (n=0 for <b>PRI0</b> , and so on). The lower the value, the greater the priority of the corresponding interrupt.
12:8	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.
7:5	INTA	R/W	0x0	Interrupt Priority for Interrupt [4n] This field holds a priority value, 0-7, for the interrupt with the number [4n], where n is the number of the <b>Interrupt Priority</b> register (n=0 for <b>PRI0</b> , and so on). The lower the value, the greater the priority of the corresponding interrupt.
4:0	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 17: Software Trigger Interrupt (SWTRIG), offset 0xF00

**Note:** Only privileged software can enable unprivileged access to the **SWTRIG** register.

Writing an interrupt number to the **SWTRIG** register generates a Software Generated Interrupt (SGI). See Table 2-9 on page 74 for interrupt assignments.

When the **MAINPEND** bit in the **Configuration and Control (CFGCTRL)** register (see page 117) is set, unprivileged software can access the **SWTRIG** register.

### Software Trigger Interrupt (SWTRIG)

Base 0xE000.E000

Offset 0xF00

Type WO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved																
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	RO	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
31:5	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4:0	INTID	WO	0x00	Interrupt ID This field holds the interrupt ID of the required SGI. For example, a value of 0x3 generates an interrupt on IRQ3.

## 3.5 System Control Block (SCB) Register Descriptions

This section lists and describes the System Control Block (SCB) registers, in numerical order by address offset. The SCB registers can only be accessed from privileged mode.

All registers must be accessed with aligned word accesses except for the **FAULTSTAT** and **SYSPRI1-SYSPRI3** registers, which can be accessed with byte or aligned halfword or word accesses. The processor does not support unaligned accesses to system control block registers.

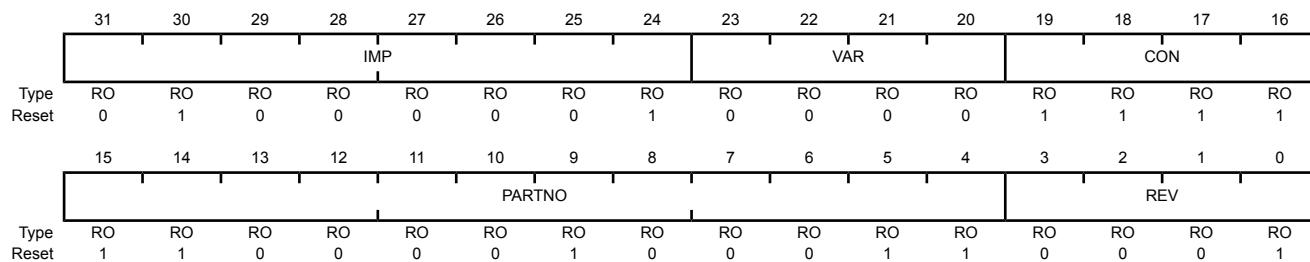
## Register 18: CPU ID Base (CPUID), offset 0xD00

**Note:** This register can only be accessed from privileged mode.

The **CPUID** register contains the ARM® Cortex™-M3 processor part number, version, and implementation information.

### CPU ID Base (CPUID)

Base 0xE000.E000  
Offset 0xD00  
Type RO, reset 0x410F.C231



Bit/Field	Name	Type	Reset	Description
31:24	IMP	RO	0x41	Implementer Code  Value Description 0x41 ARM
23:20	VAR	RO	0x0	Variant Number  Value Description 0x0 The rn value in the rnpn product revision identifier, for example, the 0 in r0p1.
19:16	CON	RO	0xF	Constant  Value Description 0xF Always reads as 0xF.
15:4	PARTNO	RO	0xC23	Part Number  Value Description 0xC23 Cortex-M3 processor.
3:0	REV	RO	0x1	Revision Number  Value Description 0x1 The pn value in the rnpn product revision identifier, for example, the 1 in r0p1.

## Register 19: Interrupt Control and State (INTCTRL), offset 0xD04

**Note:** This register can only be accessed from privileged mode.

The **INCTRL** register provides a set-pending bit for the NMI exception, and set-pending and clear-pending bits for the PendSV and SysTick exceptions. In addition, bits in this register indicate the exception number of the exception being processed, whether there are preempted active exceptions, the exception number of the highest priority pending exception, and whether any interrupts are pending.

When writing to **INCTRL**, the effect is unpredictable when writing a 1 to both the PENDSV and UNPENDSV bits, or writing a 1 to both the PENDSTSET and PENDSTCLR bits.

### Interrupt Control and State (INTCTRL)

Base 0xE000.E000  
Offset 0xD04  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NMISET	reserved		PENDSV	UNPENDSV	PENDSTSET	PENDSTCLR	reserved	ISRPRE	ISRPEND			reserved			VECPEND
Type	R/W	RO	RO	R/W	WO	R/W	WO	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
	VECPEND			RETBASE			reserved							VECACT		
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	NMISET	R/W	0	NMI Set Pending
Value Description				
0	On a read, indicates an NMI exception is not pending. On a write, no effect.			
1	On a read, indicates an NMI exception is pending. On a write, changes the NMI exception state to pending.			

Because NMI is the highest-priority exception, normally the processor enters the NMI exception handler as soon as it registers the setting of this bit, and clears this bit on entering the interrupt handler. A read of this bit by the NMI exception handler returns 1 only if the **NMI** signal is reasserted while the processor is executing that handler.

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	PENDSV	R/W	0	PendSV Set Pending
Value Description				
0	On a read, indicates a PendSV exception is not pending. On a write, no effect.			

Setting this bit is the only way to set the PendSV exception state to pending. This bit is cleared by writing a 1 to the UNPENDSV bit.

Bit/Field	Name	Type	Reset	Description						
27	UNPENDSV	WO	0	<p>PendSV Clear Pending</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>On a write, no effect.</td></tr> <tr> <td>1</td><td>On a write, removes the pending state from the PendSV exception.</td></tr> </tbody> </table> <p>This bit is write only; on a register read, its value is unknown.</p>	Value	Description	0	On a write, no effect.	1	On a write, removes the pending state from the PendSV exception.
Value	Description									
0	On a write, no effect.									
1	On a write, removes the pending state from the PendSV exception.									
26	PENDSTSET	R/W	0	<p>SysTick Set Pending</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>On a read, indicates a SysTick exception is not pending. On a write, no effect.</td></tr> <tr> <td>1</td><td>On a read, indicates a SysTick exception is pending. On a write, changes the SysTick exception state to pending.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the PENDSTCLR bit.</p>	Value	Description	0	On a read, indicates a SysTick exception is not pending. On a write, no effect.	1	On a read, indicates a SysTick exception is pending. On a write, changes the SysTick exception state to pending.
Value	Description									
0	On a read, indicates a SysTick exception is not pending. On a write, no effect.									
1	On a read, indicates a SysTick exception is pending. On a write, changes the SysTick exception state to pending.									
25	PENDSTCLR	WO	0	<p>SysTick Clear Pending</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>On a write, no effect.</td></tr> <tr> <td>1</td><td>On a write, removes the pending state from the SysTick exception.</td></tr> </tbody> </table> <p>This bit is write only; on a register read, its value is unknown.</p>	Value	Description	0	On a write, no effect.	1	On a write, removes the pending state from the SysTick exception.
Value	Description									
0	On a write, no effect.									
1	On a write, removes the pending state from the SysTick exception.									
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	ISRPRE	RO	0	<p>Debug Interrupt Handling</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The release from halt does not take an interrupt.</td></tr> <tr> <td>1</td><td>The release from halt takes an interrupt.</td></tr> </tbody> </table> <p>This bit is only meaningful in Debug mode and reads as zero when the processor is not in Debug mode.</p>	Value	Description	0	The release from halt does not take an interrupt.	1	The release from halt takes an interrupt.
Value	Description									
0	The release from halt does not take an interrupt.									
1	The release from halt takes an interrupt.									
22	ISRPEND	RO	0	<p>Interrupt Pending</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No interrupt is pending.</td></tr> <tr> <td>1</td><td>An interrupt is pending.</td></tr> </tbody> </table> <p>This bit provides status for all interrupts excluding NMI and Faults.</p>	Value	Description	0	No interrupt is pending.	1	An interrupt is pending.
Value	Description									
0	No interrupt is pending.									
1	An interrupt is pending.									
21:18	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																																						
17:12	VECPEND	RO	0x00	<p>Interrupt Pending Vector Number</p> <p>This field contains the exception number of the highest priority pending enabled exception. The value indicated by this field includes the effect of the <b>BASEPRI</b> and <b>FAULTMASK</b> registers, but not any effect of the <b>PRIMASK</b> register.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr><td>0x00</td><td>No exceptions are pending</td></tr> <tr><td>0x01</td><td>Reserved</td></tr> <tr><td>0x02</td><td>NMI</td></tr> <tr><td>0x03</td><td>Hard fault</td></tr> <tr><td>0x04</td><td>Memory management fault</td></tr> <tr><td>0x05</td><td>Bus fault</td></tr> <tr><td>0x06</td><td>Usage fault</td></tr> <tr><td>0x07-0xA</td><td>Reserved</td></tr> <tr><td>0x0B</td><td>SVCALL</td></tr> <tr><td>0x0C</td><td>Reserved for Debug</td></tr> <tr><td>0x0D</td><td>Reserved</td></tr> <tr><td>0x0E</td><td>PendSV</td></tr> <tr><td>0x0F</td><td>SysTick</td></tr> <tr><td>0x10</td><td>Interrupt Vector 0</td></tr> <tr><td>0x11</td><td>Interrupt Vector 1</td></tr> <tr><td>...</td><td>...</td></tr> <tr><td>0x2D</td><td>Interrupt Vector 29</td></tr> <tr><td>0x2E-0x3F</td><td>Reserved</td></tr> </tbody> </table>	Value	Description	0x00	No exceptions are pending	0x01	Reserved	0x02	NMI	0x03	Hard fault	0x04	Memory management fault	0x05	Bus fault	0x06	Usage fault	0x07-0xA	Reserved	0x0B	SVCALL	0x0C	Reserved for Debug	0x0D	Reserved	0x0E	PendSV	0x0F	SysTick	0x10	Interrupt Vector 0	0x11	Interrupt Vector 1	...	...	0x2D	Interrupt Vector 29	0x2E-0x3F	Reserved
Value	Description																																									
0x00	No exceptions are pending																																									
0x01	Reserved																																									
0x02	NMI																																									
0x03	Hard fault																																									
0x04	Memory management fault																																									
0x05	Bus fault																																									
0x06	Usage fault																																									
0x07-0xA	Reserved																																									
0x0B	SVCALL																																									
0x0C	Reserved for Debug																																									
0x0D	Reserved																																									
0x0E	PendSV																																									
0x0F	SysTick																																									
0x10	Interrupt Vector 0																																									
0x11	Interrupt Vector 1																																									
...	...																																									
0x2D	Interrupt Vector 29																																									
0x2E-0x3F	Reserved																																									
11	RETBASE	RO	0	<p>Return to Base</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr><td>0</td><td>There are preempted active exceptions to execute.</td></tr> <tr><td>1</td><td>There are no active exceptions, or the currently executing exception is the only active exception.</td></tr> </tbody> </table> <p>This bit provides status for all interrupts excluding NMI and Faults. This bit only has meaning if the processor is currently executing an ISR (the <b>Interrupt Program Status (IPSR)</b> register is non-zero).</p>	Value	Description	0	There are preempted active exceptions to execute.	1	There are no active exceptions, or the currently executing exception is the only active exception.																																
Value	Description																																									
0	There are preempted active exceptions to execute.																																									
1	There are no active exceptions, or the currently executing exception is the only active exception.																																									
10:6	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.																																						
5:0	VECACT	RO	0x00	<p>Interrupt Pending Vector Number</p> <p>This field contains the active exception number. The exception numbers can be found in the description for the <b>VECPEND</b> field. If this field is clear, the processor is in Thread mode. This field contains the same value as the <b>ISRNUM</b> field in the <b>IPSR</b> register.</p> <p>Subtract 16 from this value to obtain the IRQ number required to index into the <b>Interrupt Set Enable (ENn)</b>, <b>Interrupt Clear Enable (DISn)</b>, <b>Interrupt Set Pending (PENDn)</b>, <b>Interrupt Clear Pending (UNPENDn)</b>, and <b>Interrupt Priority (PRIn)</b> registers (see page 55).</p>																																						

## Register 20: Vector Table Offset (VTABLE), offset 0xD08

**Note:** This register can only be accessed from privileged mode.

The **VTABLE** register indicates the offset of the vector table base address from memory address 0x0000.0000.

### Vector Table Offset (VTABLE)

Base 0xE000.E000  
Offset 0xD08  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OFFSET																
Type	RO	RO	R/W													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OFFSET																
Type	R/W	RO														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																

Bit/Field	Name	Type	Reset	Description
31:30	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.
29	BASE	R/W	0	Vector Table Base
	Value	Description		
	0	The vector table is in the code memory region.		
	1	The vector table is in the SRAM memory region.		
28:8	OFFSET	R/W	0x000.00	Vector Table Offset  When configuring the OFFSET field, the offset must be aligned to the number of exception entries in the vector table. Because there are 29 interrupts, the offset must be aligned on a 256-byte boundary.
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 21: Application Interrupt and Reset Control (APINT), offset 0xD0C

**Note:** This register can only be accessed from privileged mode.

The APINT register provides priority grouping control for the exception model, endian status for data accesses, and reset control of the system. To write to this register, 0x05FA must be written to the VECTKEY field, otherwise the write is ignored.

The PRIGROUP field indicates the position of the binary point that splits the INT<sub>x</sub> fields in the **Interrupt Priority (PRIx)** registers into separate group priority and subpriority fields. Table 3-8 on page 113 shows how the PRIGROUP value controls this split. The bit numbers in the Group Priority Field and Subpriority Field columns in the table refer to the bits in the INTA field. For the INTB field, the corresponding bits are 15:13; for INTC, 23:21; and for INTD, 31:29.

**Note:** Determining preemption of an exception uses only the group priority field.

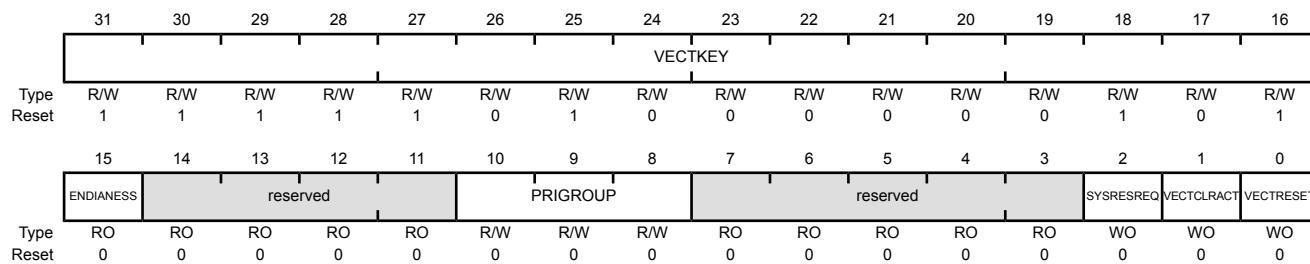
**Table 3-8. Interrupt Priority Levels**

PRIGROUP Bit Field	Binary Point <sup>a</sup>	Group Priority Field	Subpriority Field	Group Priorities	Subpriorities
0x0 - 0x4	bxxx.	[7:5]	None	8	1
0x5	bxx.y	[7:6]	[5]	4	2
0x6	bx.yy	[7]	[6:5]	2	4
0x7	b.yyy	None	[7:5]	1	8

a. INT<sub>x</sub> field showing the binary point. An x denotes a group priority field bit, and a y denotes a subpriority field bit.

### Application Interrupt and Reset Control (APINT)

Base 0xE000.E000  
Offset 0xD0C  
Type R/W, reset 0xFA05.0000



Bit/Field	Name	Type	Reset	Description
31:16	VECTKEY	R/W	0xFA05	Register Key This field is used to guard against accidental writes to this register. 0x05FA must be written to this field in order to change the bits in this register. On a read, 0xFA05 is returned.
15	ENDIANESS	RO	0	Data Endianess The Stellaris implementation uses only little-endian mode so this is cleared to 0.
14:11	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
10:8	PRIGROUP	R/W	0x0	Interrupt Priority Grouping This field determines the split of group priority from subpriority (see Table 3-8 on page 113 for more information).
7:3	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.
2	SYSRESREQ	WO	0	System Reset Request  Value Description 0 No effect. 1 Resets the core and all on-chip peripherals except the Debug interface.  This bit is automatically cleared during the reset of the core and reads as 0.
1	VECTCLRACT	WO	0	Clear Active NMI / Fault This bit is reserved for Debug use and reads as 0. This bit must be written as a 0, otherwise behavior is unpredictable.
0	VECTRESET	WO	0	System Reset This bit is reserved for Debug use and reads as 0. This bit must be written as a 0, otherwise behavior is unpredictable.

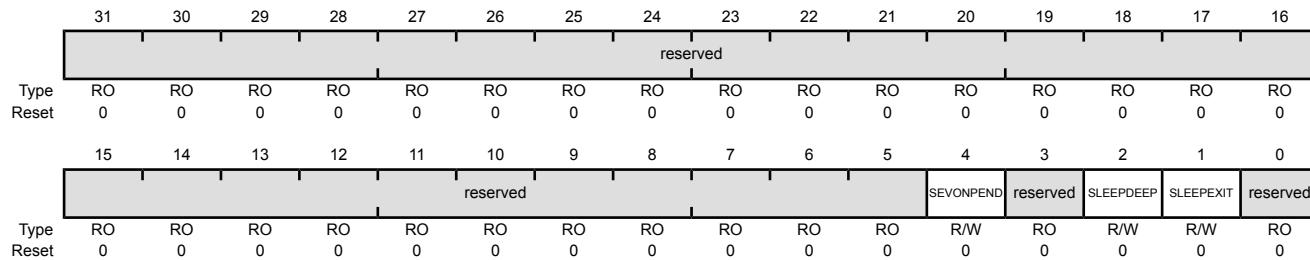
## Register 22: System Control (SYSCTRL), offset 0xD10

**Note:** This register can only be accessed from privileged mode.

The **SYSCTRL** register controls features of entry to and exit from low-power state.

### System Control (SYSCTRL)

Base 0xE000.E000  
Offset 0xD10  
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.00	Software should not rely on the value of 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	SEVONPEND	R/W	0	<p>Wake Up on Pending</p> <p>Value Description</p> <p>0 Only enabled interrupts or events can wake up the processor; disabled interrupts are excluded.</p> <p>1 Enabled events and all interrupts, including disabled interrupts, can wake up the processor.</p> <p>When an event or interrupt enters the pending state, the event signal wakes up the processor from WFE. If the processor is not waiting for an event, the event is registered and affects the next WFE.</p> <p>The processor also wakes up on execution of a SEV instruction or an external event.</p>
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	SLEEPDEEP	R/W	0	<p>Deep Sleep Enable</p> <p>Value Description</p> <p>0 Use Sleep mode as the low power mode.</p> <p>1 Use Deep-sleep mode as the low power mode.</p>

Bit/Field	Name	Type	Reset	Description				
1	SLEEP EXIT	R/W	0	<p>Sleep on ISR Exit</p> <p>Value Description</p> <table><tr><td>0</td><td>When returning from Handler mode to Thread mode, do not sleep when returning to Thread mode.</td></tr><tr><td>1</td><td>When returning from Handler mode to Thread mode, enter sleep or deep sleep on return from an ISR.</td></tr></table> <p>Setting this bit enables an interrupt-driven application to avoid returning to an empty main application.</p>	0	When returning from Handler mode to Thread mode, do not sleep when returning to Thread mode.	1	When returning from Handler mode to Thread mode, enter sleep or deep sleep on return from an ISR.
0	When returning from Handler mode to Thread mode, do not sleep when returning to Thread mode.							
1	When returning from Handler mode to Thread mode, enter sleep or deep sleep on return from an ISR.							
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.				

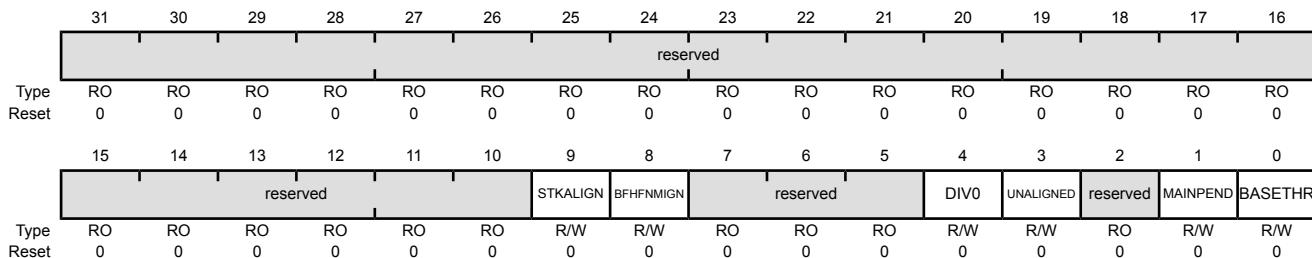
## Register 23: Configuration and Control (CFGCTRL), offset 0xD14

**Note:** This register can only be accessed from privileged mode.

The **CFGCTRL** register controls entry to Thread mode and enables: the handlers for NMI, hard fault and faults escalated by the **FAULTMASK** register to ignore bus faults; trapping of divide by zero and unaligned accesses; and access to the **SWTRIG** register by unprivileged software (see page 107).

### Configuration and Control (CFGCTRL)

Base 0xE000.E000  
Offset 0xD14  
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x0000.00	Software should not rely on the value of 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	STKALIGN	R/W	0	Stack Alignment on Exception Entry
	Value Description			
	0	The stack is 4-byte aligned.		
	1	The stack is 8-byte aligned.		
	On exception entry, the processor uses bit 9 of the stacked <b>PSR</b> to indicate the stack alignment. On return from the exception, it uses this stacked bit to restore the correct stack alignment.			
8	BFHFNIGN	R/W	0	Ignore Bus Fault in NMI and Fault
	This bit enables handlers with priority -1 or -2 to ignore data bus faults caused by load and store instructions. The setting of this bit applies to the hard fault, NMI, and <b>FAULTMASK</b> escalated handlers.			
	Value Description			
	0	Data bus faults caused by load and store instructions cause a lock-up.		
	1	Handlers running at priority -1 and -2 ignore data bus faults caused by load and store instructions.		
	Set this bit only when the handler and its data are in absolutely safe memory. The normal use of this bit is to probe system devices and bridges to detect control path problems and fix them.			
7:5	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.

Bit/Field	Name	Type	Reset	Description						
4	DIV0	R/W	0	<p>Trap on Divide by 0</p> <p>This bit enables faulting or halting when the processor executes an SDIV or UDIV instruction with a divisor of 0.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Do not trap on divide by 0. A divide by zero returns a quotient of 0.</td></tr> <tr> <td>1</td><td>Trap on divide by 0.</td></tr> </tbody> </table>	Value	Description	0	Do not trap on divide by 0. A divide by zero returns a quotient of 0.	1	Trap on divide by 0.
Value	Description									
0	Do not trap on divide by 0. A divide by zero returns a quotient of 0.									
1	Trap on divide by 0.									
3	UNALIGNED	R/W	0	<p>Trap on Unaligned Access</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Do not trap on unaligned halfword and word accesses.</td></tr> <tr> <td>1</td><td>Trap on unaligned halfword and word accesses. An unaligned access generates a usage fault.</td></tr> </tbody> </table> <p>Unaligned LDM, STM, LDRD, and STRD instructions always fault regardless of whether UNALIGNED is set.</p>	Value	Description	0	Do not trap on unaligned halfword and word accesses.	1	Trap on unaligned halfword and word accesses. An unaligned access generates a usage fault.
Value	Description									
0	Do not trap on unaligned halfword and word accesses.									
1	Trap on unaligned halfword and word accesses. An unaligned access generates a usage fault.									
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	MAINPEND	R/W	0	<p>Allow Main Interrupt Trigger</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Disables unprivileged software access to the <b>SWTRIG</b> register.</td></tr> <tr> <td>1</td><td>Enables unprivileged software access to the <b>SWTRIG</b> register (see page 107).</td></tr> </tbody> </table>	Value	Description	0	Disables unprivileged software access to the <b>SWTRIG</b> register.	1	Enables unprivileged software access to the <b>SWTRIG</b> register (see page 107).
Value	Description									
0	Disables unprivileged software access to the <b>SWTRIG</b> register.									
1	Enables unprivileged software access to the <b>SWTRIG</b> register (see page 107).									
0	BASETHR	R/W	0	<p>Thread State Control</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The processor can enter Thread mode only when no exception is active.</td></tr> <tr> <td>1</td><td>The processor can enter Thread mode from any level under the control of an EXC_RETURN value (see "Exception Return" on page 78 for more information).</td></tr> </tbody> </table>	Value	Description	0	The processor can enter Thread mode only when no exception is active.	1	The processor can enter Thread mode from any level under the control of an EXC_RETURN value (see "Exception Return" on page 78 for more information).
Value	Description									
0	The processor can enter Thread mode only when no exception is active.									
1	The processor can enter Thread mode from any level under the control of an EXC_RETURN value (see "Exception Return" on page 78 for more information).									

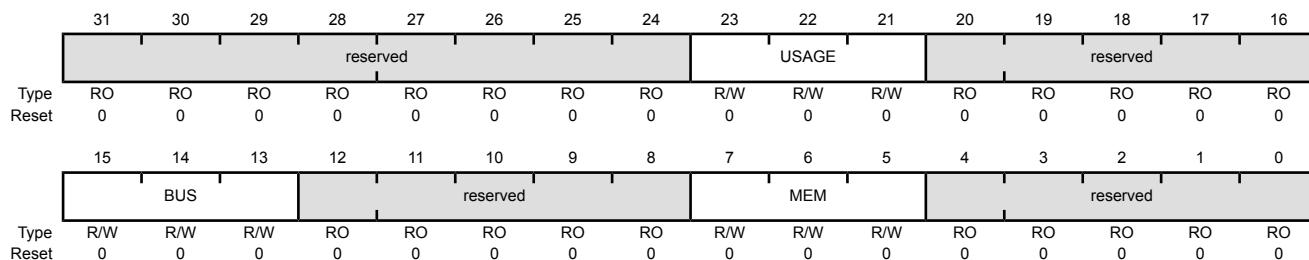
## Register 24: System Handler Priority 1 (SYSPRI1), offset 0xD18

**Note:** This register can only be accessed from privileged mode.

The **SYSPRI1** register configures the priority level, 0 to 7 of the usage fault, bus fault, and memory management fault exception handlers. This register is byte-accessible.

### System Handler Priority 1 (SYSPRI1)

Base 0xE000.E000  
Offset 0xD18  
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:24	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.
23:21	USAGE	R/W	0x0	Usage Fault Priority This field configures the priority level of the usage fault. Configurable priority values are in the range 0-7, with lower values having higher priority.
20:16	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.
15:13	BUS	R/W	0x0	Bus Fault Priority This field configures the priority level of the bus fault. Configurable priority values are in the range 0-7, with lower values having higher priority.
12:8	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.
7:5	MEM	R/W	0x0	Memory Management Fault Priority This field configures the priority level of the memory management fault. Configurable priority values are in the range 0-7, with lower values having higher priority.
4:0	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.

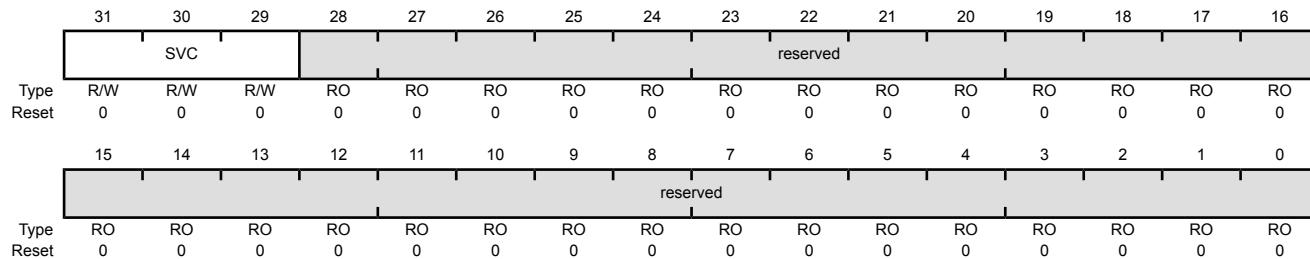
## Register 25: System Handler Priority 2 (SYSPRI2), offset 0xD1C

**Note:** This register can only be accessed from privileged mode.

The **SYSPRI2** register configures the priority level, 0 to 7 of the SVCall handler. This register is byte-accessible.

### System Handler Priority 2 (SYSPRI2)

Base 0xE000.E000  
Offset 0xD1C  
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:29	SVC	R/W	0x0	SVCall Priority This field configures the priority level of SVCall. Configurable priority values are in the range 0-7, with lower values having higher priority.
28:0	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.

## Register 26: System Handler Priority 3 (SYSPRI3), offset 0xD20

**Note:** This register can only be accessed from privileged mode.

The **SYSPRI3** register configures the priority level, 0 to 7 of the SysTick exception and PendSV handlers. This register is byte-accessible.

### System Handler Priority 3 (SYSPRI3)

Base 0xE000.E000  
Offset 0xD20  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TICK			reserved					PENDSV			reserved				
Type	R/W	R/W	R/W	RO	RO	RO	RO	RO	R/W	R/W	R/W	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					DEBUG				reserved			
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	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
31:29	TICK	R/W	0x0	SysTick Exception Priority This field configures the priority level of the SysTick exception. Configurable priority values are in the range 0-7, with lower values having higher priority.
28:24	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.
23:21	PENDSV	R/W	0x0	PendSV Priority This field configures the priority level of PendSV. Configurable priority values are in the range 0-7, with lower values having higher priority.
20:8	reserved	RO	0x000	Software should not rely on the value of 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	DEBUG	R/W	0x0	Debug Priority This field configures the priority level of Debug. Configurable priority values are in the range 0-7, with lower values having higher priority.
4:0	reserved	RO	0x0.0000	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.

## Register 27: System Handler Control and State (SYSHNDCTRL), offset 0xD24

**Note:** This register can only be accessed from privileged mode.

The **SYSHNDCTRL** register enables the system handlers, and indicates the pending status of the usage fault, bus fault, memory management fault, and SVC exceptions as well as the active status of the system handlers.

If a system handler is disabled and the corresponding fault occurs, the processor treats the fault as a hard fault.

This register can be modified to change the pending or active status of system exceptions. An OS kernel can write to the active bits to perform a context switch that changes the current exception type.

**Caution – Software that changes the value of an active bit in this register without correct adjustment to the stacked content can cause the processor to generate a fault exception. Ensure software that writes to this register retains and subsequently restores the current active status.**

**If the value of a bit in this register must be modified after enabling the system handlers, a read-modify-write procedure must be used to ensure that only the required bit is modified.**

### System Handler Control and State (SYSHNDCTRL)

Base 0xE000.E000  
Offset 0xD24  
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	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
Type	SVC	BUSP	MEMP	USAGEP	TICK	PNDSV	reserved	MON	SVCA	reserved	reserved	USGA	reserved	BUSA	MEMA	
Reset	R/W	R/W	R/W	R/W	R/W	R/W	RO	R/W	R/W	RO	RO	R/W	RO	R/W	R/W	R/W

Bit/Field	Name	Type	Reset	Description
31:19	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
18	USAGE	R/W	0	Usage Fault Enable
		Value	Description	
		0	Disables the usage fault exception.	
		1	Enables the usage fault exception.	
17	BUS	R/W	0	Bus Fault Enable
		Value	Description	
		0	Disables the bus fault exception.	
		1	Enables the bus fault exception.	

Bit/Field	Name	Type	Reset	Description
16	MEM	R/W	0	<p>Memory Management Fault Enable</p> <p>Value Description</p> <p>0      Disables the memory management fault exception.</p> <p>1      Enables the memory management fault exception.</p>
15	SVC	R/W	0	<p>SVC Call Pending</p> <p>Value Description</p> <p>0      An SVC call exception is not pending.</p> <p>1      An SVC call exception is pending.</p> <p>This bit can be modified to change the pending status of the SVC call exception.</p>
14	BUSP	R/W	0	<p>Bus Fault Pending</p> <p>Value Description</p> <p>0      A bus fault exception is not pending.</p> <p>1      A bus fault exception is pending.</p> <p>This bit can be modified to change the pending status of the bus fault exception.</p>
13	MEMP	R/W	0	<p>Memory Management Fault Pending</p> <p>Value Description</p> <p>0      A memory management fault exception is not pending.</p> <p>1      A memory management fault exception is pending.</p> <p>This bit can be modified to change the pending status of the memory management fault exception.</p>
12	USAGEP	R/W	0	<p>Usage Fault Pending</p> <p>Value Description</p> <p>0      A usage fault exception is not pending.</p> <p>1      A usage fault exception is pending.</p> <p>This bit can be modified to change the pending status of the usage fault exception.</p>
11	TICK	R/W	0	<p>SysTick Exception Active</p> <p>Value Description</p> <p>0      A SysTick exception is not active.</p> <p>1      A SysTick exception is active.</p> <p>This bit can be modified to change the active status of the SysTick exception, however, see the Caution above before setting this bit.</p>

Bit/Field	Name	Type	Reset	Description						
10	PNDSV	R/W	0	<p>PendSV Exception Active</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>A PendSV exception is not active.</td></tr> <tr> <td>1</td><td>A PendSV exception is active.</td></tr> </tbody> </table> <p>This bit can be modified to change the active status of the PendSV exception, however, see the Caution above before setting this bit.</p>	Value	Description	0	A PendSV exception is not active.	1	A PendSV exception is active.
Value	Description									
0	A PendSV exception is not active.									
1	A PendSV exception is active.									
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	MON	R/W	0	<p>Debug Monitor Active</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The Debug monitor is not active.</td></tr> <tr> <td>1</td><td>The Debug monitor is active.</td></tr> </tbody> </table>	Value	Description	0	The Debug monitor is not active.	1	The Debug monitor is active.
Value	Description									
0	The Debug monitor is not active.									
1	The Debug monitor is active.									
7	SVCA	R/W	0	<p>SVC Call Active</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>SVC call is not active.</td></tr> <tr> <td>1</td><td>SVC call is active.</td></tr> </tbody> </table> <p>This bit can be modified to change the active status of the SVC call exception, however, see the Caution above before setting this bit.</p>	Value	Description	0	SVC call is not active.	1	SVC call is active.
Value	Description									
0	SVC call is not active.									
1	SVC call is active.									
6:4	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	USGA	R/W	0	<p>Usage Fault Active</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Usage fault is not active.</td></tr> <tr> <td>1</td><td>Usage fault is active.</td></tr> </tbody> </table> <p>This bit can be modified to change the active status of the usage fault exception, however, see the Caution above before setting this bit.</p>	Value	Description	0	Usage fault is not active.	1	Usage fault is active.
Value	Description									
0	Usage fault is not active.									
1	Usage fault is active.									
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	BUSA	R/W	0	<p>Bus Fault Active</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Bus fault is not active.</td></tr> <tr> <td>1</td><td>Bus fault is active.</td></tr> </tbody> </table> <p>This bit can be modified to change the active status of the bus fault exception, however, see the Caution above before setting this bit.</p>	Value	Description	0	Bus fault is not active.	1	Bus fault is active.
Value	Description									
0	Bus fault is not active.									
1	Bus fault is active.									

Bit/Field	Name	Type	Reset	Description				
0	MEMA	R/W	0	Memory Management Fault Active				
<p>Value Description</p> <table><tr><td>0</td><td>Memory management fault is not active.</td></tr><tr><td>1</td><td>Memory management fault is active.</td></tr></table>					0	Memory management fault is not active.	1	Memory management fault is active.
0	Memory management fault is not active.							
1	Memory management fault is active.							
<p>This bit can be modified to change the active status of the memory management fault exception, however, see the Caution above before setting this bit.</p>								

## Register 28: Configurable Fault Status (FAULTSTAT), offset 0xD28

**Note:** This register can only be accessed from privileged mode.

The **FAULTSTAT** register indicates the cause of a memory management fault, bus fault, or usage fault. Each of these functions is assigned to a subregister as follows:

- **Usage Fault Status (UFAULTSTAT)**, bits 31:16
- **Bus Fault Status (BFAULTSTAT)**, bits 15:8
- **Memory Management Fault Status (MFAULTSTAT)**, bits 7:0

**FAULTSTAT** is byte accessible. **FAULTSTAT** or its subregisters can be accessed as follows:

- The complete **FAULTSTAT** register, with a word access to offset 0xD28
- The **MFAULTSTAT**, with a byte access to offset 0xD28
- The **MFAULTSTAT** and **BFAULTSTAT**, with a halfword access to offset 0xD28
- The **BFAULTSTAT**, with a byte access to offset 0xD29
- The **UFAULTSTAT**, with a halfword access to offset 0xD2A

Bits are cleared by writing a 1 to them.

In a fault handler, the true faulting address can be determined by:

1. Read and save the **Memory Management Fault Address (MMADDR)** or **Bus Fault Address (FAULTADDR)** value.
2. Read the **MMARV** bit in **MFAULTSTAT**, or the **BFARV** bit in **BFAULTSTAT** to determine if the **MMADDR** or **FAULTADDR** contents are valid.

Software must follow this sequence because another higher priority exception might change the **MMADDR** or **FAULTADDR** value. For example, if a higher priority handler preempts the current fault handler, the other fault might change the **MMADDR** or **FAULTADDR** value.

### Configurable Fault Status (FAULTSTAT)

Base 0xE000.E000

Offset 0xD28

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	R/W1C	R/W1C	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C
Reset							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
Type	R/W1C	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	RO	RO	R/W1C	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
31:26	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						
25	DIV0	R/W1C	0	<p>Divide-by-Zero Usage Fault</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No divide-by-zero fault has occurred, or divide-by-zero trapping is not enabled.</td></tr> <tr> <td>1</td><td>The processor has executed an SDIV or UDIV instruction with a divisor of 0.</td></tr> </tbody> </table> <p>When this bit is set, the <b>PC</b> value stacked for the exception return points to the instruction that performed the divide by zero.</p> <p>Trapping on divide-by-zero is enabled by setting the DIV0 bit in the <b>Configuration and Control (CFGCTRL)</b> register (see page 117).</p> <p>This bit is cleared by writing a 1 to it.</p>	Value	Description	0	No divide-by-zero fault has occurred, or divide-by-zero trapping is not enabled.	1	The processor has executed an SDIV or UDIV instruction with a divisor of 0.
Value	Description									
0	No divide-by-zero fault has occurred, or divide-by-zero trapping is not enabled.									
1	The processor has executed an SDIV or UDIV instruction with a divisor of 0.									
24	UNALIGN	R/W1C	0	<p>Unaligned Access Usage Fault</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No unaligned access fault has occurred, or unaligned access trapping is not enabled.</td></tr> <tr> <td>1</td><td>The processor has made an unaligned memory access.</td></tr> </tbody> </table> <p>Unaligned LDM, STM, LDRD, and STRD instructions always fault regardless of the configuration of this bit.</p> <p>Trapping on unaligned access is enabled by setting the UNALIGNED bit in the <b>CFGCTRL</b> register (see page 117).</p> <p>This bit is cleared by writing a 1 to it.</p>	Value	Description	0	No unaligned access fault has occurred, or unaligned access trapping is not enabled.	1	The processor has made an unaligned memory access.
Value	Description									
0	No unaligned access fault has occurred, or unaligned access trapping is not enabled.									
1	The processor has made an unaligned memory access.									
23:20	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.						
19	NOCP	R/W1C	0	<p>No Coprocessor Usage Fault</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>A usage fault has not been caused by attempting to access a coprocessor.</td></tr> <tr> <td>1</td><td>The processor has attempted to access a coprocessor.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to it.</p>	Value	Description	0	A usage fault has not been caused by attempting to access a coprocessor.	1	The processor has attempted to access a coprocessor.
Value	Description									
0	A usage fault has not been caused by attempting to access a coprocessor.									
1	The processor has attempted to access a coprocessor.									
18	INVPC	R/W1C	0	<p>Invalid PC Load Usage Fault</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>A usage fault has not been caused by attempting to load an invalid <b>PC</b> value.</td></tr> <tr> <td>1</td><td>The processor has attempted an illegal load of EXC_RETURN to the <b>PC</b> as a result of an invalid context or an invalid EXC_RETURN value.</td></tr> </tbody> </table> <p>When this bit is set, the <b>PC</b> value stacked for the exception return points to the instruction that tried to perform the illegal load of the <b>PC</b>.</p> <p>This bit is cleared by writing a 1 to it.</p>	Value	Description	0	A usage fault has not been caused by attempting to load an invalid <b>PC</b> value.	1	The processor has attempted an illegal load of EXC_RETURN to the <b>PC</b> as a result of an invalid context or an invalid EXC_RETURN value.
Value	Description									
0	A usage fault has not been caused by attempting to load an invalid <b>PC</b> value.									
1	The processor has attempted an illegal load of EXC_RETURN to the <b>PC</b> as a result of an invalid context or an invalid EXC_RETURN value.									

Bit/Field	Name	Type	Reset	Description
17	INVSTAT	R/W1C	0	<p>Invalid State Usage Fault</p> <p>Value Description</p> <p>0 A usage fault has not been caused by an invalid state.</p> <p>1 The processor has attempted to execute an instruction that makes illegal use of the <b>EPSR</b> register.</p> <p>When this bit is set, the <b>PC</b> value stacked for the exception return points to the instruction that attempted the illegal use of the <b>Execution Program Status Register (EPSR)</b> register.</p> <p>This bit is not set if an undefined instruction uses the <b>EPSR</b> register.</p> <p>This bit is cleared by writing a 1 to it.</p>
16	UNDEF	R/W1C	0	<p>Undefined Instruction Usage Fault</p> <p>Value Description</p> <p>0 A usage fault has not been caused by an undefined instruction.</p> <p>1 The processor has attempted to execute an undefined instruction.</p> <p>When this bit is set, the <b>PC</b> value stacked for the exception return points to the undefined instruction.</p> <p>An undefined instruction is an instruction that the processor cannot decode.</p> <p>This bit is cleared by writing a 1 to it.</p>
15	BFARV	R/W1C	0	<p>Bus Fault Address Register Valid</p> <p>Value Description</p> <p>0 The value in the <b>Bus Fault Address (FAULTADDR)</b> register is not a valid fault address.</p> <p>1 The <b>FAULTADDR</b> register is holding a valid fault address.</p> <p>This bit is set after a bus fault, where the address is known. Other faults can clear this bit, such as a memory management fault occurring later. If a bus fault occurs and is escalated to a hard fault because of priority, the hard fault handler must clear this bit. This action prevents problems if returning to a stacked active bus fault handler whose <b>FAULTADDR</b> register value has been overwritten.</p> <p>This bit is cleared by writing a 1 to it.</p>
14:13	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.

Bit/Field	Name	Type	Reset	Description						
12	BSTKE	R/W1C	0	<p>Stack Bus Fault</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No bus fault has occurred on stacking for exception entry.</td></tr> <tr> <td>1</td><td>Stacking for an exception entry has caused one or more bus faults.</td></tr> </tbody> </table> <p>When this bit is set, the <b>SP</b> is still adjusted but the values in the context area on the stack might be incorrect. A fault address is not written to the <b>FAULTADDR</b> register.</p> <p>This bit is cleared by writing a 1 to it.</p>	Value	Description	0	No bus fault has occurred on stacking for exception entry.	1	Stacking for an exception entry has caused one or more bus faults.
Value	Description									
0	No bus fault has occurred on stacking for exception entry.									
1	Stacking for an exception entry has caused one or more bus faults.									
11	BUSTKE	R/W1C	0	<p>Unstack Bus Fault</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No bus fault has occurred on unstacking for a return from exception.</td></tr> <tr> <td>1</td><td>Unstacking for a return from exception has caused one or more bus faults.</td></tr> </tbody> </table> <p>This fault is chained to the handler. Thus, when this bit is set, the original return stack is still present. The <b>SP</b> is not adjusted from the failing return, a new save is not performed, and a fault address is not written to the <b>FAULTADDR</b> register.</p> <p>This bit is cleared by writing a 1 to it.</p>	Value	Description	0	No bus fault has occurred on unstacking for a return from exception.	1	Unstacking for a return from exception has caused one or more bus faults.
Value	Description									
0	No bus fault has occurred on unstacking for a return from exception.									
1	Unstacking for a return from exception has caused one or more bus faults.									
10	IMPRE	R/W1C	0	<p>Imprecise Data Bus Error</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>An imprecise data bus error has not occurred.</td></tr> <tr> <td>1</td><td>A data bus error has occurred, but the return address in the stack frame is not related to the instruction that caused the error.</td></tr> </tbody> </table> <p>When this bit is set, a fault address is not written to the <b>FAULTADDR</b> register.</p> <p>This fault is asynchronous. Therefore, if the fault is detected when the priority of the current process is higher than the bus fault priority, the bus fault becomes pending and becomes active only when the processor returns from all higher-priority processes. If a precise fault occurs before the processor enters the handler for the imprecise bus fault, the handler detects that both the <b>IMPRE</b> bit is set and one of the precise fault status bits is set.</p> <p>This bit is cleared by writing a 1 to it.</p>	Value	Description	0	An imprecise data bus error has not occurred.	1	A data bus error has occurred, but the return address in the stack frame is not related to the instruction that caused the error.
Value	Description									
0	An imprecise data bus error has not occurred.									
1	A data bus error has occurred, but the return address in the stack frame is not related to the instruction that caused the error.									
9	PRECISE	R/W1C	0	<p>Precise Data Bus Error</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>A precise data bus error has not occurred.</td></tr> <tr> <td>1</td><td>A data bus error has occurred, and the <b>PC</b> value stacked for the exception return points to the instruction that caused the fault.</td></tr> </tbody> </table> <p>When this bit is set, the fault address is written to the <b>FAULTADDR</b> register.</p> <p>This bit is cleared by writing a 1 to it.</p>	Value	Description	0	A precise data bus error has not occurred.	1	A data bus error has occurred, and the <b>PC</b> value stacked for the exception return points to the instruction that caused the fault.
Value	Description									
0	A precise data bus error has not occurred.									
1	A data bus error has occurred, and the <b>PC</b> value stacked for the exception return points to the instruction that caused the fault.									

Bit/Field	Name	Type	Reset	Description
8	IBUS	R/W1C	0	<p>Instruction Bus Error</p> <p>Value Description</p> <ul style="list-style-type: none"> <li>0 An instruction bus error has not occurred.</li> <li>1 An instruction bus error has occurred.</li> </ul> <p>The processor detects the instruction bus error on prefetching an instruction, but sets this bit only if it attempts to issue the faulting instruction.</p> <p>When this bit is set, a fault address is not written to the <b>FAULTADDR</b> register.</p> <p>This bit is cleared by writing a 1 to it.</p>
7	MMARV	R/W1C	0	<p>Memory Management Fault Address Register Valid</p> <p>Value Description</p> <ul style="list-style-type: none"> <li>0 The value in the <b>Memory Management Fault Address (MMADDR)</b> register is not a valid fault address.</li> <li>1 The <b>MMADDR</b> register is holding a valid fault address.</li> </ul> <p>If a memory management fault occurs and is escalated to a hard fault because of priority, the hard fault handler must clear this bit. This action prevents problems if returning to a stacked active memory management fault handler whose <b>MMADDR</b> register value has been overwritten.</p> <p>This bit is cleared by writing a 1 to it.</p>
6:5	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	MSTKE	R/W1C	0	<p>Stack Access Violation</p> <p>Value Description</p> <ul style="list-style-type: none"> <li>0 No memory management fault has occurred on stacking for exception entry.</li> <li>1 Stacking for an exception entry has caused one or more access violations.</li> </ul> <p>When this bit is set, the <b>SP</b> is still adjusted but the values in the context area on the stack might be incorrect. A fault address is not written to the <b>MMADDR</b> register.</p> <p>This bit is cleared by writing a 1 to it.</p>

Bit/Field	Name	Type	Reset	Description						
3	MUSTKE	R/W1C	0	<p>Unstack Access Violation</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No memory management fault has occurred on unstacking for a return from exception.</td></tr> <tr> <td>1</td><td>Unstacking for a return from exception has caused one or more access violations.</td></tr> </tbody> </table> <p>This fault is chained to the handler. Thus, when this bit is set, the original return stack is still present. The <b>SP</b> is not adjusted from the failing return, a new save is not performed, and a fault address is not written to the <b>MMADDR</b> register.</p> <p>This bit is cleared by writing a 1 to it.</p>	Value	Description	0	No memory management fault has occurred on unstacking for a return from exception.	1	Unstacking for a return from exception has caused one or more access violations.
Value	Description									
0	No memory management fault has occurred on unstacking for a return from exception.									
1	Unstacking for a return from exception has caused one or more access violations.									
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	DERR	R/W1C	0	<p>Data Access Violation</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>A data access violation has not occurred.</td></tr> <tr> <td>1</td><td>The processor attempted a load or store at a location that does not permit the operation.</td></tr> </tbody> </table> <p>When this bit is set, the <b>PC</b> value stacked for the exception return points to the faulting instruction and the address of the attempted access is written to the <b>MMADDR</b> register.</p> <p>This bit is cleared by writing a 1 to it.</p>	Value	Description	0	A data access violation has not occurred.	1	The processor attempted a load or store at a location that does not permit the operation.
Value	Description									
0	A data access violation has not occurred.									
1	The processor attempted a load or store at a location that does not permit the operation.									
0	IERR	R/W1C	0	<p>Instruction Access Violation</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>An instruction access violation has not occurred.</td></tr> <tr> <td>1</td><td>The processor attempted an instruction fetch from a location that does not permit execution.</td></tr> </tbody> </table> <p>This fault occurs on any access to an XN region, even when the MPU is disabled or not present.</p> <p>When this bit is set, the <b>PC</b> value stacked for the exception return points to the faulting instruction and the address of the attempted access is not written to the <b>MMADDR</b> register.</p> <p>This bit is cleared by writing a 1 to it.</p>	Value	Description	0	An instruction access violation has not occurred.	1	The processor attempted an instruction fetch from a location that does not permit execution.
Value	Description									
0	An instruction access violation has not occurred.									
1	The processor attempted an instruction fetch from a location that does not permit execution.									

## Register 29: Hard Fault Status (HFAULTSTAT), offset 0xD2C

**Note:** This register can only be accessed from privileged mode.

The **HFAULTSTAT** register gives information about events that activate the hard fault handler.

Bits are cleared by writing a 1 to them.

### Hard Fault Status (HFAULTSTAT)

Base 0xE000.E000

Offset 0xD2C

Type R/W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DBG	FORCED								reserved						
Type	R/W1C	R/W1C	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					VECT	reserved
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	DBG	R/W1C	0	Debug Event This bit is reserved for Debug use. This bit must be written as a 0, otherwise behavior is unpredictable.
30	FORCED	R/W1C	0	Forced Hard Fault  Value Description 0 No forced hard fault has occurred. 1 A forced hard fault has been generated by escalation of a fault with configurable priority that cannot be handled, either because of priority or because it is disabled.  When this bit is set, the hard fault handler must read the other fault status registers to find the cause of the fault. This bit is cleared by writing a 1 to it.
29:2	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	VECT	R/W1C	0	Vector Table Read Fault  Value Description 0 No bus fault has occurred on a vector table read. 1 A bus fault occurred on a vector table read.  This error is always handled by the hard fault handler. When this bit is set, the <b>PC</b> value stacked for the exception return points to the instruction that was preempted by the exception. This bit is cleared by writing a 1 to it.
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 30: Memory Management Fault Address (MMADDR), offset 0xD34

**Note:** This register can only be accessed from privileged mode.

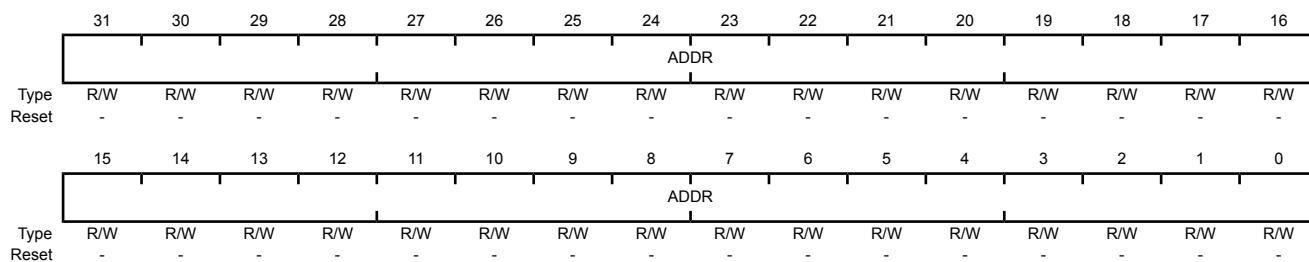
The **MMADDR** register contains the address of the location that generated a memory management fault. When an unaligned access faults, the address in the **MMADDR** register is the actual address that faulted. Because a single read or write instruction can be split into multiple aligned accesses, the fault address can be any address in the range of the requested access size. Bits in the **Memory Management Fault Status (MFAULTSTAT)** register indicate the cause of the fault and whether the value in the **MMADDR** register is valid (see page 126).

### Memory Management Fault Address (MMADDR)

Base 0xE000.E000

Offset 0xD34

Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31:0	ADDR	R/W	-	Fault Address When the MMARV bit of MFAULTSTAT is set, this field holds the address of the location that generated the memory management fault.

## Register 31: Bus Fault Address (FAULTADDR), offset 0xD38

**Note:** This register can only be accessed from privileged mode.

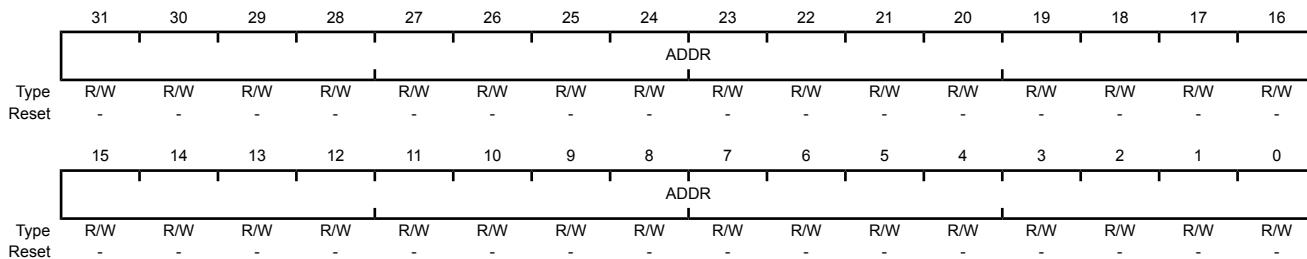
The **FAULTADDR** register contains the address of the location that generated a bus fault. When an unaligned access faults, the address in the **FAULTADDR** register is the one requested by the instruction, even if it is not the address of the fault. Bits in the **Bus Fault Status (BFAULTSTAT)** register indicate the cause of the fault and whether the value in the **FAULTADDR** register is valid (see page 126).

### Bus Fault Address (FAULTADDR)

Base 0xE000.E000

Offset 0xD38

Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31:0	ADDR	R/W	-	Fault Address When the FAULTADDRV bit of <b>BFAULTSTAT</b> is set, this field holds the address of the location that generated the bus fault.

## 3.6 Memory Protection Unit (MPU) Register Descriptions

This section lists and describes the Memory Protection Unit (MPU) registers, in numerical order by address offset.

The MPU registers can only be accessed from privileged mode.

## Register 32: MPU Type (MPUTYPE), offset 0xD90

**Note:** This register can only be accessed from privileged mode.

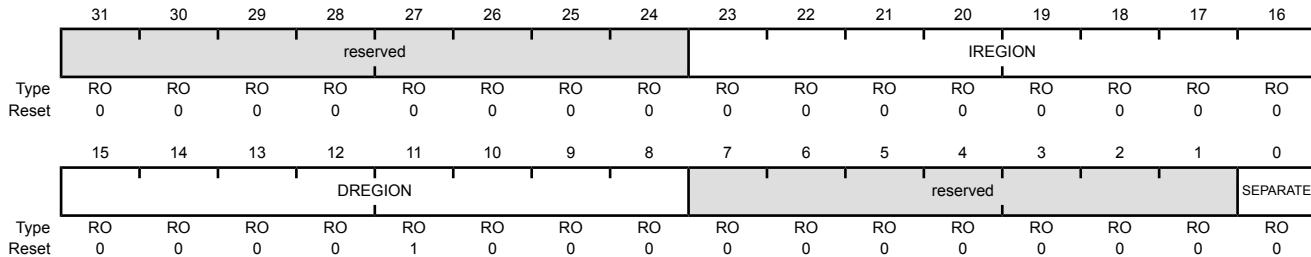
The **MPUTYPE** register indicates whether the MPU is present, and if so, how many regions it supports.

### MPU Type (MPUTYPE)

Base 0xE000.E000

Offset 0xD90

Type RO, reset 0x0000.0800



Bit/Field	Name	Type	Reset	Description
31:24	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.
23:16	IREGION	RO	0x00	<p>Number of I Regions</p> <p>This field indicates the number of supported MPU instruction regions. This field always contains 0x00. The MPU memory map is unified and is described by the DREGION field.</p>
15:8	DREGION	RO	0x08	<p>Number of D Regions</p> <p>Value Description</p> <p>0x08 Indicates there are eight supported MPU data regions.</p>
7:1	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	SEPARATE	RO	0	<p>Separate or Unified MPU</p> <p>Value Description</p> <p>0 Indicates the MPU is unified.</p>

## Register 33: MPU Control (MPUCTRL), offset 0xD94

**Note:** This register can only be accessed from privileged mode.

The **MPUCTRL** register enables the MPU, enables the default memory map background region, and enables use of the MPU when in the hard fault, Non-maskable Interrupt (NMI), and **Fault Mask Register (FAULTMASK)** escalated handlers.

When the `ENABLE` and `PRIVDEFEN` bits are both set:

- For privileged accesses, the default memory map is as described in “Memory Model” on page 63. Any access by privileged software that does not address an enabled memory region behaves as defined by the default memory map.
  - Any access by unprivileged software that does not address an enabled memory region causes a memory management fault.

Execute Never (XN) and Strongly Ordered rules always apply to the System Control Space regardless of the value of the ENABLE bit.

When the `ENABLE` bit is set, at least one region of the memory map must be enabled for the system to function unless the `PRIVDEFEN` bit is set. If the `PRIVDEFEN` bit is set and no regions are enabled, then only privileged software can operate.

When the `ENABLE` bit is clear, the system uses the default memory map, which has the same memory attributes as if the MPU is not implemented (see Table 2-5 on page 65 for more information). The default memory map applies to accesses from both privileged and unprivileged software.

When the MPU is enabled, accesses to the System Control Space and vector table are always permitted. Other areas are accessible based on regions and whether `PRIVDEFEN` is set.

Unless HFNMIEA is set, the MPU is not enabled when the processor is executing the handler for an exception with priority –1 or –2. These priorities are only possible when handling a hard fault or NMI exception or when **FAULTMASK** is enabled. Setting the HFNMIEA bit enables the MPU when operating with these two priorities.

## MPU Control (MPUCTRL)

Base 0xE000.E000

Offset 0xD94

Type R/W, reset 0x0000.0000

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of 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						
2	PRIVDEFEN	R/W	0	<p>MPU Default Region This bit enables privileged software access to the default memory map.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>If the MPU is enabled, this bit disables use of the default memory map. Any memory access to a location not covered by any enabled region causes a fault.</td></tr> <tr> <td>1</td><td>If the MPU is enabled, this bit enables use of the default memory map as a background region for privileged software accesses.</td></tr> </tbody> </table> <p>When this bit is set, the background region acts as if it is region number -1. Any region that is defined and enabled has priority over this default map. If the MPU is disabled, the processor ignores this bit.</p>	Value	Description	0	If the MPU is enabled, this bit disables use of the default memory map. Any memory access to a location not covered by any enabled region causes a fault.	1	If the MPU is enabled, this bit enables use of the default memory map as a background region for privileged software accesses.
Value	Description									
0	If the MPU is enabled, this bit disables use of the default memory map. Any memory access to a location not covered by any enabled region causes a fault.									
1	If the MPU is enabled, this bit enables use of the default memory map as a background region for privileged software accesses.									
1	HFNMIENA	R/W	0	<p>MPU Enabled During Faults This bit controls the operation of the MPU during hard fault, NMI, and <b>FAULTMASK</b> handlers.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The MPU is disabled during hard fault, NMI, and <b>FAULTMASK</b> handlers, regardless of the value of the ENABLE bit.</td></tr> <tr> <td>1</td><td>The MPU is enabled during hard fault, NMI, and <b>FAULTMASK</b> handlers.</td></tr> </tbody> </table> <p>When the MPU is disabled and this bit is set, the resulting behavior is unpredictable.</p>	Value	Description	0	The MPU is disabled during hard fault, NMI, and <b>FAULTMASK</b> handlers, regardless of the value of the ENABLE bit.	1	The MPU is enabled during hard fault, NMI, and <b>FAULTMASK</b> handlers.
Value	Description									
0	The MPU is disabled during hard fault, NMI, and <b>FAULTMASK</b> handlers, regardless of the value of the ENABLE bit.									
1	The MPU is enabled during hard fault, NMI, and <b>FAULTMASK</b> handlers.									
0	ENABLE	R/W	0	<p>MPU Enable</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The MPU is disabled.</td></tr> <tr> <td>1</td><td>The MPU is enabled.</td></tr> </tbody> </table> <p>When the MPU is disabled and the HFNMIENA bit is set, the resulting behavior is unpredictable.</p>	Value	Description	0	The MPU is disabled.	1	The MPU is enabled.
Value	Description									
0	The MPU is disabled.									
1	The MPU is enabled.									

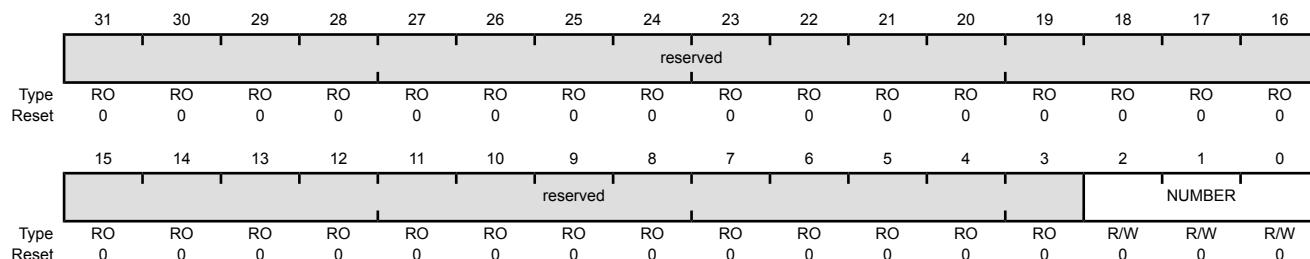
## Register 34: MPU Region Number (MPUNUMBER), offset 0xD98

**Note:** This register can only be accessed from privileged mode.

The **MPUNUMBER** register selects which memory region is referenced by the **MPU Region Base Address (MPUBASE)** and **MPU Region Attribute and Size (MPUATTR)** registers. Normally, the required region number should be written to this register before accessing the **MPUBASE** or the **MPUATTR** register. However, the region number can be changed by writing to the **MPUBASE** register with the **VALID** bit set (see page 139). This write updates the value of the **REGION** field.

### MPU Region Number (MPUNUMBER)

Base 0xE000.E000  
Offset 0xD98  
Type R/W, reset 0x0000.0000



**Register 35: MPU Region Base Address (MPUBASE), offset 0xD9C****Register 36: MPU Region Base Address Alias 1 (MPUBASE1), offset 0xDA4****Register 37: MPU Region Base Address Alias 2 (MPUBASE2), offset 0xDAC****Register 38: MPU Region Base Address Alias 3 (MPUBASE3), offset 0xDB4**

**Note:** This register can only be accessed from privileged mode.

The **MPUBASE** register defines the base address of the MPU region selected by the **MPU Region Number (MPUNUMBER)** register and can update the value of the **MPUNUMBER** register. To change the current region number and update the **MPUNUMBER** register, write the **MPUBASE** register with the **VALID** bit set.

The **ADDR** field is bits 31:N of the **MPUBASE** register. Bits (N-1):5 are reserved. The region size, as specified by the **SIZE** field in the **MPU Region Attribute and Size (MPUATTR)** register, defines the value of N where:

$$N = \log_2(\text{Region size in bytes})$$

If the region size is configured to 4 GB in the **MPUATTR** register, there is no valid **ADDR** field. In this case, the region occupies the complete memory map, and the base address is 0x0000.0000.

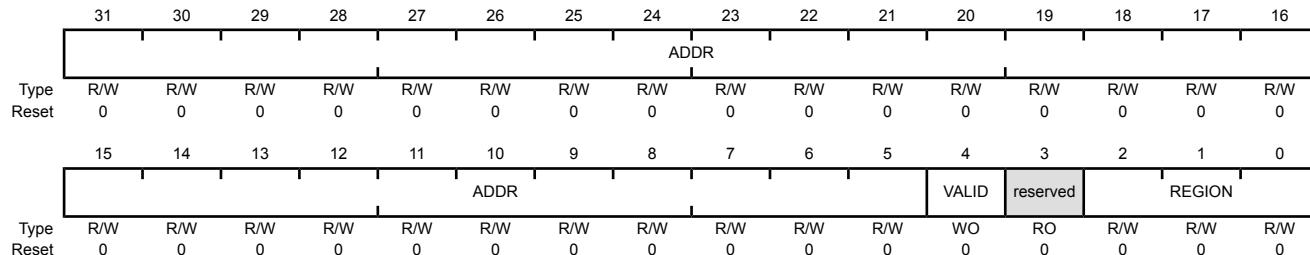
The base address is aligned to the size of the region. For example, a 64-KB region must be aligned on a multiple of 64 KB, for example, at 0x0001.0000 or 0x0002.0000.

**MPU Region Base Address (MPUBASE)**

Base 0xE000.E000

Offset 0xD9C

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:5	ADDR	R/W	0x0000.0000	<p>Base Address Mask</p> <p>Bits 31:N in this field contain the region base address. The value of N depends on the region size, as shown above. The remaining bits (N-1):5 are reserved.</p> <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>

Bit/Field	Name	Type	Reset	Description
4	VALID	WO	0	Region Number Valid  Value Description 0 The <b>MPUNUMBER</b> register is not changed and the processor updates the base address for the region specified in the <b>MPUNUMBER</b> register and ignores the value of the REGION field. 1 The <b>MPUNUMBER</b> register is updated with the value of the REGION field and the base address is updated for the region specified in the REGION field.  This bit is always read as 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.
2:0	REGION	R/W	0x0	Region Number On a write, contains the value to be written to the <b>MPUNUMBER</b> register. On a read, returns the current region number in the <b>MPUNUMBER</b> register.

**Register 39: MPU Region Attribute and Size (MPUATTR), offset 0xDA0****Register 40: MPU Region Attribute and Size Alias 1 (MPUATTR1), offset 0xDA8****Register 41: MPU Region Attribute and Size Alias 2 (MPUATTR2), offset 0xDB0****Register 42: MPU Region Attribute and Size Alias 3 (MPUATTR3), offset 0xDB8**

**Note:** This register can only be accessed from privileged mode.

The **MPUATTR** register defines the region size and memory attributes of the MPU region specified by the **MPU Region Number (MPUNUMBER)** register and enables that region and any subregions.

The **MPUATTR** register is accessible using word or halfword accesses with the most-significant halfword holding the region attributes and the least-significant halfword holds the region size and the region and subregion enable bits.

The MPU access permission attribute bits, XN, AP, TEX, S, C, and B, control access to the corresponding memory region. If an access is made to an area of memory without the required permissions, then the MPU generates a permission fault.

The SIZE field defines the size of the MPU memory region specified by the **MPUNUMBER** register as follows:

$$(\text{Region size in bytes}) = 2^{(\text{SIZE}+1)}$$

The smallest permitted region size is 32 bytes, corresponding to a SIZE value of 4. Table 3-9 on page 141 gives example SIZE values with the corresponding region size and value of N in the **MPU Region Base Address (MPUBASE)** register.

**Table 3-9. Example SIZE Field Values**

SIZE Encoding	Region Size	Value of N <sup>a</sup>	Note
00100b (0x4)	32 B	5	Minimum permitted size
01001b (0x9)	1 KB	10	-
10011b (0x13)	1 MB	20	-
11101b (0x1D)	1 GB	30	-
11111b (0x1F)	4 GB	No valid ADDR field in <b>MPUBASE</b> ; the region occupies the complete memory map.	Maximum possible size

a. Refers to the N parameter in the **MPUBASE** register (see page 139).

#### MPU Region Attribute and Size (MPUATTR)

Base 0xE000.E000

Offset 0xDA0

Type R/W, reset 0x0000.0000

MPU Region Attribute and Size (MPUATTR)															
Base 0xE000.E000, Offset 0xDA0, Type R/W, Reset 0x0000.0000															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	RO	RO	RO	R/W	RO	R/W	R/W	RO	RO	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
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRD										reserved		SIZE		ENABLE	
Type	R/W	RO	RO	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

Bit/Field	Name	Type	Reset	Description
31:29	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.
28	XN	R/W	0	Instruction Access Disable  Value Description 0 Instruction fetches are enabled. 1 Instruction fetches are disabled.
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	AP	R/W	0	Access Privilege For information on using this bit field, see Table 3-5 on page 93.
23:22	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.
21:19	TEX	R/W	0x0	Type Extension Mask For information on using this bit field, see Table 3-3 on page 92.
18	S	R/W	0	Shareable For information on using this bit, see Table 3-3 on page 92.
17	C	R/W	0	Cacheable For information on using this bit, see Table 3-3 on page 92.
16	B	R/W	0	Bufferable For information on using this bit, see Table 3-3 on page 92.
15:8	SRD	R/W	0x00	Subregion Disable Bits  Value Description 0 The corresponding subregion is enabled. 1 The corresponding subregion is disabled.  Region sizes of 128 bytes and less do not support subregions. When writing the attributes for such a region, configure the SRD field as 0x00. See the section called "Subregions" on page 91 for more information.
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.
5:1	SIZE	R/W	0x0	Region Size Mask The SIZE field defines the size of the MPU memory region specified by the MPUNUMBER register. Refer to Table 3-9 on page 141 for more information.

---

Bit/Field	Name	Type	Reset	Description
0	ENABLE	R/W	0	Region Enable
				Value Description
			0	The region is disabled.
			1	The region is enabled.

## 4 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 five pins: `TRST`, `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 Stellaris® 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 Stellaris JTAG instructions select the Stellaris `TDO` outputs. The multiplexer is controlled by the Stellaris JTAG controller, which has comprehensive programming for the ARM, Stellaris, and unimplemented JTAG instructions.

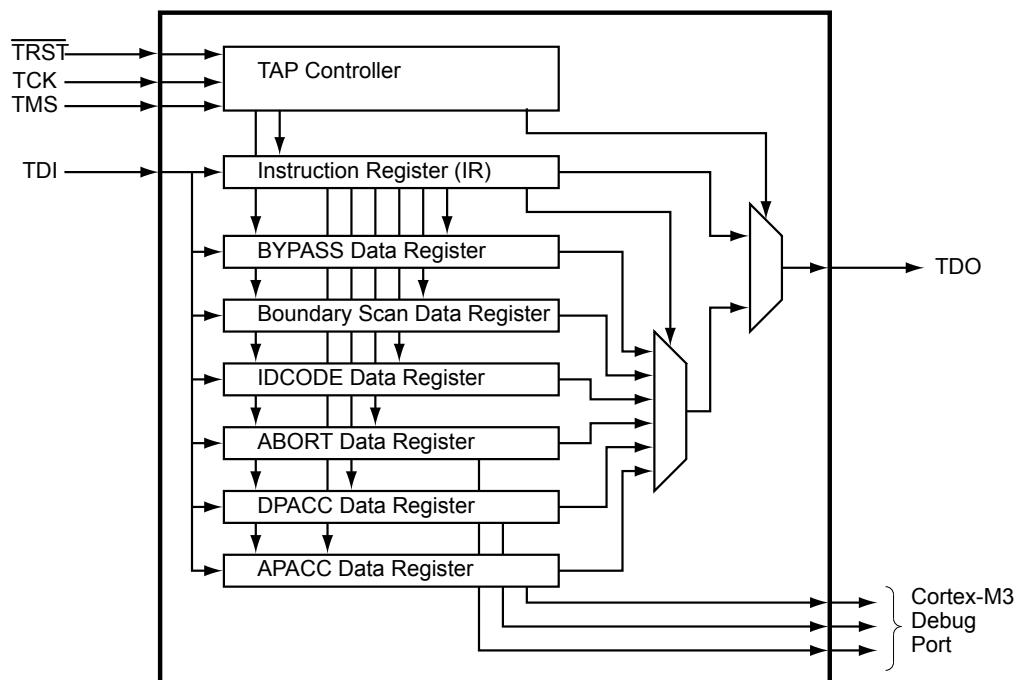
The Stellaris JTAG module has the following features:

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

See the *ARM® Debug Interface V5 Architecture Specification* for more information on the ARM JTAG controller.

## 4.1 Block Diagram

Figure 4-1. JTAG Module Block Diagram



## 4.2 Signal Description

Table 4-1 on page 145 and Table 4-2 on page 146 list the external signals of the JTAG/SWD controller and describe the function of each. The JTAG/SWD controller signals are alternate functions for some GPIO signals, however note that the reset state of the pins is for the JTAG/SWD function. The column in the table below titled "Pin Assignment" lists the GPIO pin placement for the JTAG/SWD controller signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 253) is set to choose the JTAG/SWD function. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 232.

Table 4-1. JTAG\_SWD\_SWO Signals (48QFP)

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
SWCLK	40	I	TTL	JTAG/SWD CLK.
SWDIO	39	I/O	TTL	JTAG TMS and SWDIO.
SWO	37	O	TTL	JTAG TDO and SWO.
TCK	40	I	TTL	JTAG/SWD CLK.
TDI	38	I	TTL	JTAG TDI.
TDO	37	O	TTL	JTAG TDO and SWO.
TMS	39	I/O	TTL	JTAG TMS and SWDIO.
TRST	41	I	TTL	JTAG TRST.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

**Table 4-2. JTAG\_SWD\_SWO Signals (48QFN)**

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
SWCLK	40	I	TTL	JTAG/SWD CLK.
SWDIO	39	I/O	TTL	JTAG TMS and SWDIO.
SWO	37	O	TTL	JTAG TDO and SWO.
TCK	40	I	TTL	JTAG/SWD CLK.
TDI	38	I	TTL	JTAG TDI.
TDO	37	O	TTL	JTAG TDO and SWO.
TMS	39	I/O	TTL	JTAG TMS and SWDIO.
TRST	41	I	TTL	JTAG TRST.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

## 4.3 Functional Description

A high-level conceptual drawing of the JTAG module is shown in Figure 4-1 on page 145. 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  $\overline{\text{TRST}}$ , TCK and TMS inputs. The current state of the TAP controller depends on the current value of  $\overline{\text{TRST}}$  and 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 4-4 on page 151 for a list of implemented instructions).

See “JTAG and Boundary Scan” on page 553 for JTAG timing diagrams.

### 4.3.1 JTAG Interface Pins

The JTAG interface consists of five standard pins:  $\overline{\text{TRST}}$ , TCK, TMS, TDI, and TDO. These pins and their associated reset state are given in Table 4-3 on page 146. Detailed information on each pin follows.

**Table 4-3. JTAG Port Pins Reset State**

Pin Name	Data Direction	Internal Pull-Up	Internal Pull-Down	Drive Strength	Drive Value
$\overline{\text{TRST}}$	Input	Enabled	Disabled	N/A	N/A
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

#### 4.3.1.1 Test Reset Input ( $\overline{\text{TRST}}$ )

The  $\overline{\text{TRST}}$  pin is an asynchronous active Low input signal for initializing and resetting the JTAG TAP controller and associated JTAG circuitry. When  $\overline{\text{TRST}}$  is asserted, the TAP controller resets to the Test-Logic-Reset state and remains there while  $\overline{\text{TRST}}$  is asserted. When the TAP controller enters the Test-Logic-Reset state, the JTAG Instruction Register (IR) resets to the default instruction, IDCODE.

By default, the internal pull-up resistor on the  $\overline{\text{TRST}}$  pin is enabled after reset. Changes to the pull-up resistor settings on GPIO Port B should ensure that the internal pull-up resistor remains enabled on PB7/ $\overline{\text{TRST}}$ ; otherwise JTAG communication could be lost.

#### 4.3.1.2 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.

#### 4.3.1.3 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 Instruction Register (IR) resets to the default instruction, IDCODE. Therefore, this sequence can be used as a reset mechanism, similar to asserting  $\overline{\text{TRST}}$ . The JTAG Test Access Port state machine can be seen in its entirety in Figure 4-2 on page 149.

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.

#### 4.3.1.4 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.

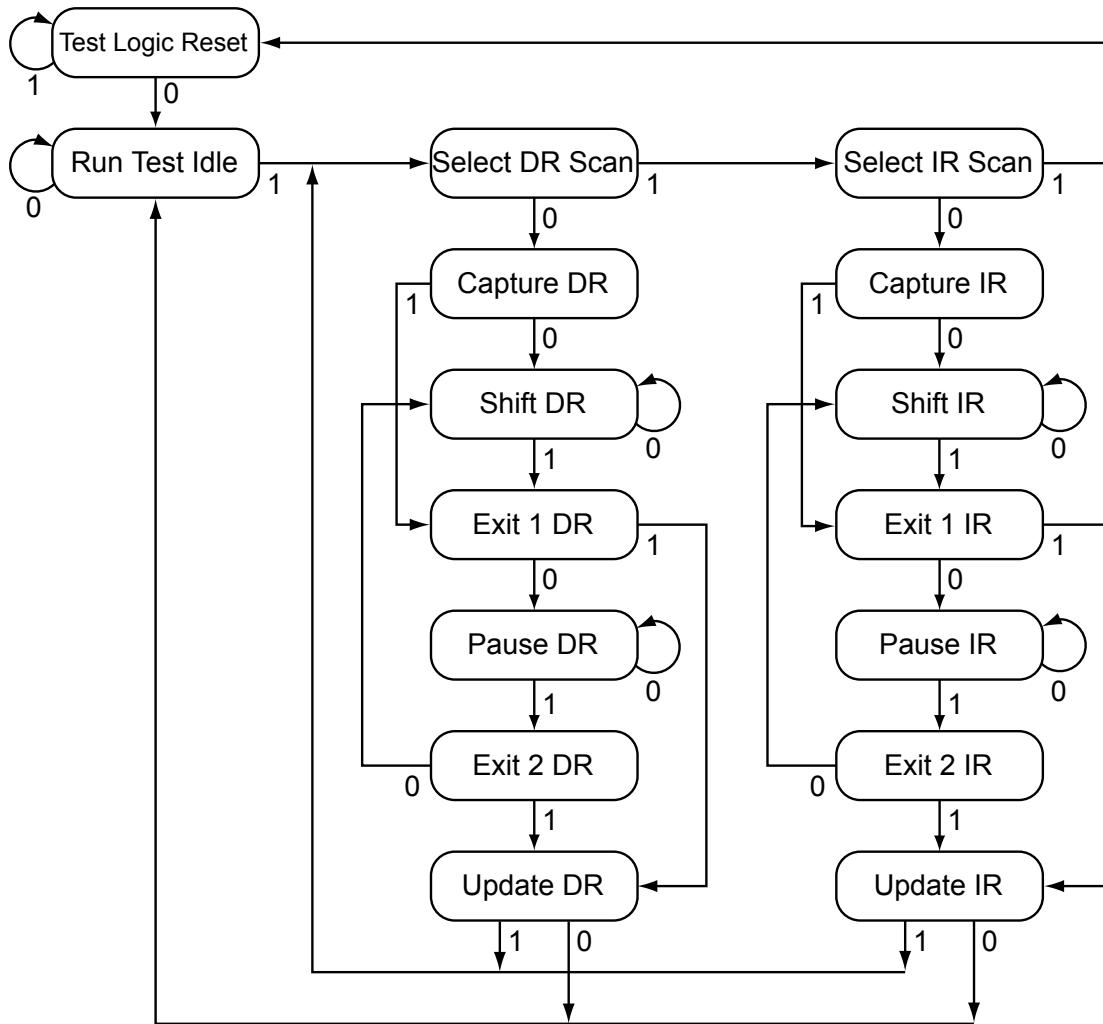
#### 4.3.1.5 Test Data Output (TDO)

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.

#### 4.3.2 JTAG TAP Controller

The JTAG TAP controller state machine is shown in Figure 4-2 on page 149. The TAP controller state machine is reset to the Test-Logic-Reset state on the assertion of a Power-On-Reset (POR) or the assertion of TRST. 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 4-2. Test Access Port State Machine**

### 4.3.3 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 151.

### 4.3.4 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.

#### 4.3.4.1 GPIO Functionality

When the microcontroller is reset with either a POR or  $\overline{RST}$ , the JTAG port pins default to their JTAG configurations. The default configuration includes enabling the pull-up resistors (setting **GPIOPUR** to 1 for **PB7** and **PC[3:0]**) and enabling the alternate hardware function (setting **GPIOAFSEL** to 1 for **PB7** and **PC[3:0]**) on the JTAG pins.

It is possible for software to configure these pins as GPIOs after reset by writing 0s to **PB7** and **PC[3:0]** in the **GPIOAFSEL** register. If the user does not require the JTAG port for debugging or board-level testing, this provides five more GPIOs for use in the design.

**Caution – If the JTAG pins are used as GPIOs in a design, **PB7** and **PC2** cannot have external pull-down resistors connected to both of them at the same time. If both pins are pulled Low during reset, the controller has unpredictable behavior. If this happens, remove one or both of the pull-down resistors, and apply  $\overline{RST}$  or power-cycle the part.**

**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.**

---

#### 4.3.4.2 Communication with JTAG/SWD

Because the debug clock and the system clock can be running at different frequencies, care must be taken to maintain reliable communication with the JTAG/SWD interface. In the Capture-DR state, the result of the previous transaction, if any, is returned, together with a 3-bit ACK response. Software should check the ACK response to see if the previous operation has completed before initiating a new transaction. Alternatively, if the system clock is at least 8 times faster than the debug clock (TCK or SWCLK), the previous operation has enough time to complete and the ACK bits do not have to be checked.

#### 4.3.4.3 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 switching 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, Capture IR, Exit1 IR, Update IR, Run Test Idle, Select DR, Select IR, Capture IR, Exit1 IR, Update IR, Run Test Idle, Select DR, Select IR, and Test-Logic-Reset states.

Stepping through the JTAG TAP Instruction Register (IR) load sequences of the TAP state machine twice without shifting in a new instruction enables the SWD interface and disables the JTAG interface. For more information on this operation and the SWD interface, see the *ARM® Debug Interface V5 Architecture Specification*.

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.

## 4.4 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 five JTAG pins ( $PB7$  and  $PC[3:0]$ ) for their alternate function using the **GPIOAFSEL** register. In addition to enabling the alternate functions, any other changes to the GPIO pad configurations on the five JTAG pins ( $PB7$  and  $PC[3:0]$ ) should be reverted to their default settings.

## 4.5 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.

### 4.5.1 Instruction Register (IR)

The JTAG TAP Instruction Register (IR) is a four-bit serial scan chain connected between the JTAG TDI and TDO pins with a parallel load register. 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 4-4 on page 151. A detailed explanation of each instruction, along with its associated Data Register, follows.

**Table 4-4. 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.

#### 4.5.1.1 EXTEST Instruction

The EXTEST instruction is not associated with its own 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. While the EXTEST instruction is present in the Instruction Register, the Boundary Scan

Data Register can be accessed to sample and shift out the current data and load new data into the Boundary Scan Data Register.

#### 4.5.1.2 INTEST Instruction

The INTEST instruction is not associated with its own 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. It is important to note that although the `RST` input pin is on the Boundary Scan Data Register chain, it is only observable. While the INTEXT instruction is present in the Instruction Register, the Boundary Scan Data Register can be accessed to sample and shift out the current data and load new data into the Boundary Scan Data Register.

#### 4.5.1.3 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 154 for more information.

#### 4.5.1.4 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 154 for more information.

#### 4.5.1.5 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 154 for more information.

#### 4.5.1.6 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 154 for more information.

#### 4.5.1.7 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, TRST is asserted, or the Test-Logic-Reset state is entered. Please see “IDCODE Data Register” on page 153 for more information.

#### 4.5.1.8 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 153 for more information.

### 4.5.2 Data Registers

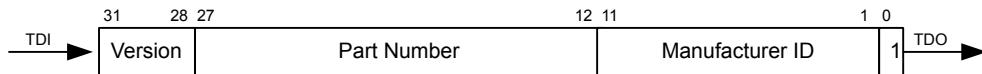
The JTAG module contains six Data Registers. These include: IDCODE, BYPASS, Boundary Scan, APACC, DPACC, and ABORT serial Data Register chains. Each of these Data Registers is discussed in the following sections.

#### 4.5.2.1 IDCODE Data Register

The format for the 32-bit IDCODE Data Register defined by the *IEEE Standard 1149.1* is shown in Figure 4-3 on page 153. 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 0x1BA0.0477. This allows the debuggers to automatically configure themselves to work correctly with the Cortex-M3 during debug.

**Figure 4-3. IDCODE Register Format**

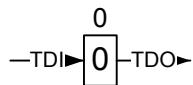


#### 4.5.2.2 BYPASS Data Register

The format for the 1-bit BYPASS Data Register defined by the *IEEE Standard 1149.1* is shown in Figure 4-4 on page 154. 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 4-4. BYPASS Register Format**

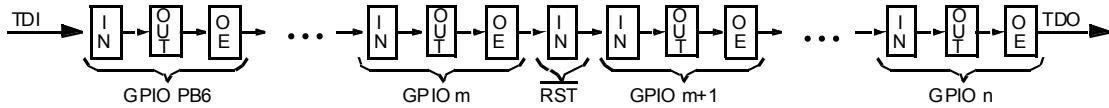


#### 4.5.2.3 Boundary Scan Data Register

The format of the Boundary Scan Data Register is shown in Figure 4-5 on page 154. Each GPIO pin, starting with a GPIO pin next to 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.

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 4-5. Boundary Scan Register Format**



#### 4.5.2.4 APACC Data Register

The format for the 35-bit APACC Data Register defined by ARM is described in the *ARM® Debug Interface V5 Architecture Specification*.

#### 4.5.2.5 DPACC Data Register

The format for the 35-bit DPACC Data Register defined by ARM is described in the *ARM® Debug Interface V5 Architecture Specification*.

#### 4.5.2.6 ABORT Data Register

The format for the 35-bit ABORT Data Register defined by ARM is described in the *ARM® Debug Interface V5 Architecture Specification*.

## 5 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.

### 5.1 Signal Description

Table 5-1 on page 155 and Table 5-2 on page 155 list the external signals of the System Control module and describe the function of each. The **NMI** signal is the alternate function for and functions as a GPIO after reset. under commit protection and require a special process to be configured as any alternate function or to subsequently return to the GPIO function. The column in the table below titled "Pin Assignment" lists the GPIO pin placement for the **NMI** signal. The **AFSEL** bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 253) should be set to choose the NMI function. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 232. The remaining signals (with the word "fixed" in the Pin Assignment column) have a fixed pin assignment and function.

**Table 5-1. System Control & Clocks Signals (48QFP)**

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
OSC0	9	I	Analog	Main oscillator crystal input or an external clock reference input.
OSC1	10	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
<b>RST</b>	5	I	TTL	System reset input.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

**Table 5-2. System Control & Clocks Signals (48QFN)**

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
OSC0	9	I	Analog	Main oscillator crystal input or an external clock reference input.
OSC1	10	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
<b>RST</b>	5	I	TTL	System reset input.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

### 5.2 Functional Description

The System Control module provides the following capabilities:

- Device identification (see "Device Identification" on page 155)
- Local control, such as reset (see "Reset Control" on page 156), power (see "Power Control" on page 160) and clock control (see "Clock Control" on page 160)
- System control (Run, Sleep, and Deep-Sleep modes); see "System Control" on page 164

#### 5.2.1 Device Identification

Several 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-DC4** registers.

## 5.2.2 Reset Control

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

### 5.2.2.1 Reset Sources

The controller has six sources of reset:

1. External reset input pin ( $\overline{RST}$ ) assertion; see “External  $\overline{RST}$  Pin” on page 157.
2. Power-on reset (POR); see “Power-On Reset (POR)” on page 156.
3. Internal brown-out (BOR) detector; see “Brown-Out Reset (BOR)” on page 158.
4. Software-initiated reset (with the software reset registers); see “Software Reset” on page 159.
5. A watchdog timer reset condition violation; see “Watchdog Timer Reset” on page 159.
6. Internal low drop-out (LDO) regulator output.

Table 5-3 provides a summary of results of the various reset operations.

**Table 5-3. Reset Sources**

Reset Source	Core Reset?	JTAG Reset?	On-Chip Peripherals Reset?
Power-On Reset	Yes	Yes	Yes
$\overline{RST}$	Yes	Pin Config Only	Yes
Brown-Out Reset	Yes	No	Yes
Software System Request Reset <sup>a</sup>	Yes	No	Yes
Software Peripheral Reset	No	No	Yes <sup>b</sup>
Watchdog Reset	Yes	No	Yes
LDO Reset	Yes	No	Yes

a. By using the SYSRESREQ bit in the ARM Cortex-M3 Application Interrupt and Reset Control (APINT) register

b. Programmable on a module-by-module basis using the Software Reset Control Registers.

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 external reset is the cause, and then all the other bits in the **RESC** register are cleared.

**Note:** The main oscillator is used for external resets and power-on resets; the internal oscillator is used during the internal process by internal reset and clock verification circuitry.

### 5.2.2.2 Power-On Reset (POR)

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

The internal Power-On Reset (POR) circuit monitors the power supply voltage ( $V_{DD}$ ) and generates a reset signal to all of the internal logic including JTAG when the power supply ramp reaches a threshold value ( $V_{TH}$ ). The microcontroller must be operating within the specified operating parameters when the on-chip power-on reset pulse is complete. The 3.3-V power supply to the microcontroller must reach 3.0 V within 10 msec of  $V_{DD}$  crossing 2.0 V to guarantee proper operation. For applications that require the use of an external reset signal to hold the microcontroller in reset longer than the internal POR, the  $\overline{RST}$  input may be used as discussed in “External  $\overline{RST}$  Pin” on page 157.

The Power-On Reset sequence is as follows:

1. The microcontroller waits for internal POR to go inactive.
2. The internal reset is released and the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

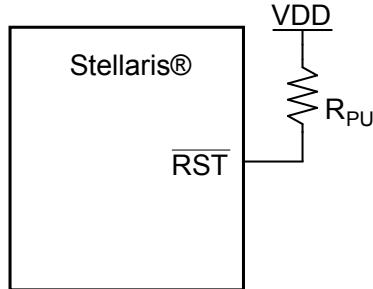
The internal POR is only active on the initial power-up of the microcontroller. The Power-On Reset timing is shown in Figure 19-6 on page 556.

### 5.2.2.3 External $\overline{\text{RST}}$ Pin

**Note:** It is recommended that the trace for the  $\overline{\text{RST}}$  signal must be kept as short as possible. Be sure to place any components connected to the  $\overline{\text{RST}}$  signal as close to the microcontroller as possible.

If the application only uses the internal POR circuit, the  $\overline{\text{RST}}$  input must be connected to the power supply ( $V_{DD}$ ) through an optional pull-up resistor (0 to 100K  $\Omega$ ) as shown in Figure 5-1 on page 157.

**Figure 5-1. Basic RST Configuration**

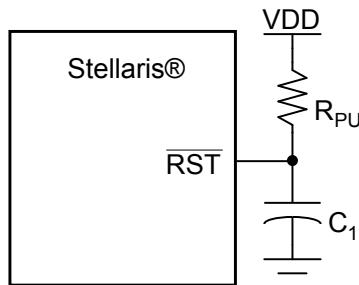


$$R_{PU} = 0 \text{ to } 100 \text{ k}\Omega$$

The external reset pin ( $\overline{\text{RST}}$ ) resets the microcontroller including the core and all the on-chip peripherals except the JTAG TAP controller (see “JTAG Interface” on page 144). The external reset sequence is as follows:

1. The external reset pin ( $\overline{\text{RST}}$ ) is asserted for the duration specified by  $T_{MIN}$  and then de-asserted (see “Reset” on page 555).
2. The internal reset is released and the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

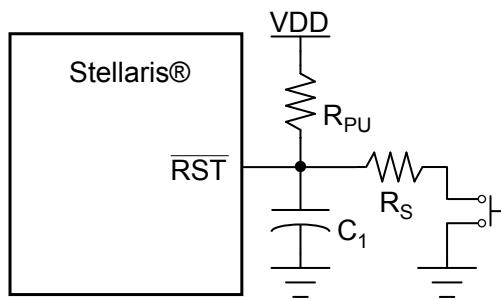
To improve noise immunity and/or to delay reset at power up, the  $\overline{\text{RST}}$  input may be connected to an RC network as shown in Figure 5-2 on page 158.

**Figure 5-2. External Circuitry to Extend Power-On Reset**

$R_{PU} = 1\text{ k}\Omega$  to  $100\text{ k}\Omega$

$C_1 = 1\text{ nF}$  to  $10\text{ }\mu\text{F}$

If the application requires the use of an external reset switch, Figure 5-3 on page 158 shows the proper circuitry to use.

**Figure 5-3. Reset Circuit Controlled by Switch**

Typical  $R_{PU} = 10\text{ k}\Omega$

Typical  $R_S = 470\text{ }\Omega$

$C_1 = 10\text{ nF}$

The  $R_{PU}$  and  $C_1$  components define the power-on delay.

The external reset timing is shown in Figure 19-5 on page 556.

#### 5.2.2.4 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 ( $V_{BTH}$ ). The circuit is provided to guard against improper operation of logic and peripherals that operate off the power supply voltage ( $V_{DD}$ ) and not the LDO voltage. If a brown-out condition is detected, the system may generate a controller interrupt or a system reset. The BOR circuit has a digital filter that protects against noise-related detection for the interrupt condition. This feature may be optionally enabled.

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 sequence is as follows:

1. When  $V_{DD}$  drops below  $V_{BTH}$ , an internal BOR condition is set.
2. If the **BORWT** bit in the **PBORCTL** register is set and **BORIOR** is not set, the BOR condition is resampled, after a delay specified by **BORTIM**, to determine if the original condition was caused by noise. If the BOR condition is not met the second time, then no further action is taken.
3. If the BOR condition exists, an internal reset is asserted.
4. The internal reset is released and the controller fetches and loads the initial stack pointer, the initial program counter, the first instruction designated by the program counter, and begins execution.
5. The internal BOR condition is reset after 500  $\mu$ s to prevent another BOR condition from being set before software has a chance to investigate the original cause.

The internal Brown-Out Reset timing is shown in Figure 19-7 on page 556.

#### 5.2.2.5 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 **SRCCRn** 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 164). 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 19-8 on page 557.

#### 5.2.2.6 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 19-9 on page 557.

### 5.2.2.7 Low Drop-Out (LDO)

A reset can be initiated when the internal low drop-out (LDO) regulator output goes unregulated. This is initially disabled and may be enabled by software. LDO is controlled with the **LDO Power Control (LDOPCTL)** register. The LDO reset sequence is as follows:

1. LDO goes unregulated and the `LDOARST` bit in the **LDOARST** register is set.
2. An internal reset is asserted.
3. The internal reset is released and the controller fetches and loads the initial stack pointer, the initial program counter, the first instruction designated by the program counter, and begins execution.

The LDO reset timing is shown in Figure 19-10 on page 557.

## 5.2.3 Power Control

The Stellaris® microcontroller provides an integrated LDO regulator that is used to provide power to the majority of the controller's internal logic. For power reduction, 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.

## 5.2.4 Clock Control

System control determines the control of clocks in this part.

### 5.2.4.1 Fundamental Clock Sources

There are multiple 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  $12\text{ MHz} \pm 30\%$ .
- **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. The crystal value allowed depends on whether the main oscillator is used as the clock reference source to the PLL. If so, the crystal must be one of the supported frequencies between 3.579545 MHz through 8.192 MHz (inclusive). If the PLL is not being used, the crystal may be any one of the supported frequencies between 1 MHz and 8.192 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 175).

The internal system clock (SysClk), is derived from any of the above sources plus two others: the output of the main internal PLL, and the internal oscillator divided by four (3 MHz ± 30%). The frequency of the PLL clock reference must be in the range of 3.579545 MHz to 8.192 MHz (inclusive). Table 5-4 on page 161 shows how the various clock sources can be used in a system.

**Table 5-4. Clock Source Options**

Clock Source	Drive PLL?	Used as SysClk?		
Internal Oscillator (12 MHz)	Yes	BYPASS = 0, OSCSRC = 0x1	Yes	BYPASS = 1, OSCSRC = 0x1
Internal Oscillator divide by 4 (3 MHz)	Yes	BYPASS = 0, OSCSRC = 0x2	Yes	BYPASS = 1, OSCSRC = 0x2
Main Oscillator	Yes	BYPASS = 0, OSCSRC = 0x0	Yes	BYPASS = 1, OSCSRC = 0x0

### 5.2.4.2 Clock Configuration

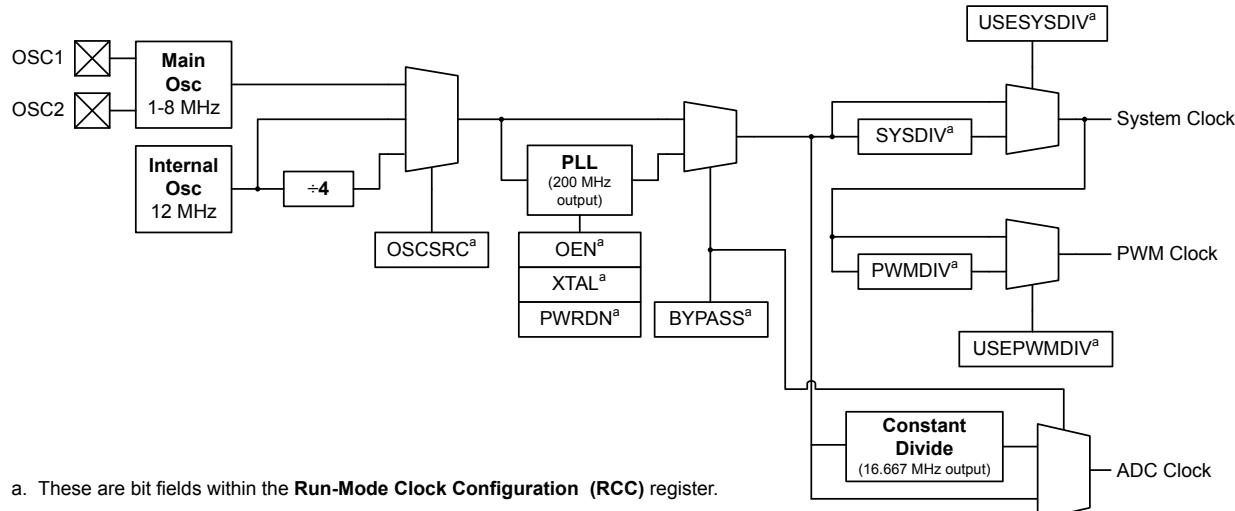
Nearly all of the control for the clocks is provided by the **Run-Mode Clock Configuration (RCC)** register. This register controls the following clock functionality:

- Source of clocks in sleep and deep-sleep modes
- System clock derived from PLL or other clock source
- Enabling/disabling of oscillators and PLL
- Clock divisors
- Crystal input selection

Figure 5-4 on page 161 shows the logic for the main clock tree. The peripheral blocks are driven by the system clock signal and can be individually enabled/disabled. The ADC clock signal is automatically divided down to 16.667 MHz for proper ADC operation. The PWM clock signal is a synchronous divide of the system clock to provide the PWM circuit with more range (set with PWMDIV in RCC).

**Note:** When the ADC module is in operation, the system clock must be at least 16.667 MHz.

**Figure 5-4. Main Clock Tree**



In the **RCC** register, the **SYSDIV** field specifies which divisor is used to generate the system clock from either the PLL output or the oscillator source (depending on how the **BYPASS** bit in this register is configured). Table 5-5 shows how the **SYSDIV** encoding affects the system clock frequency, depending on whether the PLL is used (**BYPASS=0**) or another clock source is used (**BYPASS=1**). The divisor is equivalent to the **SYSDIV** encoding plus 1. For a list of possible clock sources, see Table 5-4 on page 161.

**Table 5-5. Possible System Clock Frequencies Using the SYSDIV Field**

<b>SYSDIV</b>	<b>Divisor</b>	<b>Frequency (BYPASS=0)</b>	<b>Frequency (BYPASS=1)</b>	<b>StellarisWare Parameter<sup>a</sup></b>
0x0	/1	reserved	Clock source frequency/2	SYSCTL_SYSDIV_1 <sup>b</sup>
0x1	/2	reserved	Clock source frequency/2	SYSCTL_SYSDIV_2
0x2	/3	reserved	Clock source frequency/3	SYSCTL_SYSDIV_3
0x3	/4	50 MHz	Clock source frequency/4	SYSCTL_SYSDIV_4
0x4	/5	40 MHz	Clock source frequency/5	SYSCTL_SYSDIV_5
0x5	/6	33.33 MHz	Clock source frequency/6	SYSCTL_SYSDIV_6
0x6	/7	28.57 MHz	Clock source frequency/7	SYSCTL_SYSDIV_7
0x7	/8	25 MHz	Clock source frequency/8	SYSCTL_SYSDIV_8
0x8	/9	22.22 MHz	Clock source frequency/9	SYSCTL_SYSDIV_9
0x9	/10	20 MHz	Clock source frequency/10	SYSCTL_SYSDIV_10
0xA	/11	18.18 MHz	Clock source frequency/11	SYSCTL_SYSDIV_11
0xB	/12	16.67 MHz	Clock source frequency/12	SYSCTL_SYSDIV_12
0xC	/13	15.38 MHz	Clock source frequency/13	SYSCTL_SYSDIV_13
0xD	/14	14.29 MHz	Clock source frequency/14	SYSCTL_SYSDIV_14
0xE	/15	13.33 MHz	Clock source frequency/15	SYSCTL_SYSDIV_15
0xF	/16	12.5 MHz (default)	Clock source frequency/16	SYSCTL_SYSDIV_16

a. This parameter is used in functions such as `SysCtlClockSet()` in the Stellaris Peripheral Driver Library.

b. `SYSCTL_SYSDIV_1` does not set the `USESYSDIV` bit. As a result, using this parameter without enabling the PLL results in the system clock having the same frequency as the clock source.

### 5.2.4.3 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 8.192 MHz, otherwise, the range of supported crystals is 1 to 8.192 MHz.

The **XTAL** bit in the **RCC** register (see page 175) 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.

### 5.2.4.4 Main PLL Frequency Configuration

The main PLL is disabled by default during power-on reset and is enabled later by software if required. Software configures the main PLL input reference clock source, 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 179). The internal translation provides a translation within  $\pm 1\%$  of the targeted PLL VCO frequency.

The Crystal Value field (**XTAL**) in the **Run-Mode Clock Configuration (RCC)** register (see page 175) describes the available crystal choices and default programming of the **PLLCFG** register. Any time the **XTAL** field changes, the new settings are translated and the internal PLL settings are updated.

#### 5.2.4.5 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** register fields (see page 175).

#### 5.2.4.6 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  $T_{READY}$  (see Table 19-7 on page 553). During the relock time, the affected PLL is not usable as a clock reference.

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  $T_{READY}$  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,  $\sim 600 \mu s$  at an 8.192 MHz external oscillator clock). Hardware is provided to keep the PLL from being used as a system clock until the  $T_{READY}$  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** 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** register until the main PLL is stable ( $T_{READY}$  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 **PLLRI** bit in the **Raw Interrupt Status (RIS)** register, and enabling the PLL Lock interrupt.

#### 5.2.4.7 Clock Verification Timers

There are three identical clock verification circuits that can be enabled through software. The circuit checks the faster clock by a slower clock using timers:

- The main oscillator checks the PLL.
- The main oscillator checks the internal oscillator.
- The internal oscillator divided by 64 checks the main oscillator.

If the verification timer function is enabled and a failure is detected, the main clock tree is immediately switched to a working clock and an interrupt is generated to the controller. Software can then determine the course of action to take. The actual failure indication and clock switching does not

clear without a write to the **CLKVCLR** register, an external reset, or a POR reset. The clock verification timers are controlled by the **PLLVER**, **IOSCVER**, and **MOSCVER** bits in the **RCC** register.

### 5.2.5 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. The **DC1**, **DC2** and **DC4** registers act as a write mask for the **RCGCn**, **SCGCn**, and **DCGCn** registers.

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

- **Run Mode.** In Run mode, the controller actively executes code. 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.** In Sleep mode, the clock frequency of the active peripherals is unchanged, but the processor and the memory subsystem are not clocked and therefore no longer execute code. 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 “Power Management” on page 81 for more details.

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.** 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. 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 “Power Management” on page 81 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 **DSLPCLKCFG** register if one is enabled. When the **DSLPCLKCFG** 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. 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.

**Caution – If the Cortex-M3 Debug Access Port (DAP) has been enabled, and the device wakes from a low power sleep or deep-sleep mode, the core may start executing code before all clocks to peripherals have been restored to their run mode configuration. The DAP is usually enabled by software tools accessing the JTAG or SWD interface when debugging or flash programming. If this condition occurs, a Hard Fault is triggered when software accesses a peripheral with an invalid clock.**

A software delay loop can be used at the beginning of the interrupt routine that is used to wake up a system from a WFI (Wait For Interrupt) instruction. This stalls the execution of any code that accesses a peripheral register that might cause a fault. This loop can be removed for production software as the DAP is most likely not enabled during normal execution.

Because the DAP is disabled by default (power on reset), the user can also power-cycle the device. The DAP is not enabled unless it is enabled through the JTAG or SWD interface.

## 5.3 Initialization and Configuration

The PLL is configured using direct register writes to the **RCC** register. 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 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** and **OEN** bits in **RCC**. Setting the **XTAL** field automatically pulls valid PLL configuration data for the appropriate crystal, and clearing the **PWRDN** and **OEN** bits powers and enables the PLL and its output.
3. Select the desired system divider (**SYSDIV**) in **RCC** 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 **PLLRLIS** bit in the **Raw Interrupt Status (RIS)** register.
5. Enable use of the PLL by clearing the **BYPASS** bit in **RCC**.

**Note:** If the **BYPASS** bit is cleared before the PLL locks, it is possible to render the device unusable.

## 5.4 Register Map

Table 5-6 on page 165 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. Software should not modify any reserved memory address.

**Table 5-6. System Control Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	DID0	RO	-	Device Identification 0	167
0x004	DID1	RO	-	Device Identification 1	183
0x008	DC0	RO	0x001F.001F	Device Capabilities 0	185

**Table 5-6. System Control Register Map (continued)**

Offset	Name	Type	Reset	Description	See page
0x010	DC1	RO	0x0011.32BF	Device Capabilities 1	186
0x014	DC2	RO	0x0107.1013	Device Capabilities 2	188
0x018	DC3	RO	0xBF0F.01FF	Device Capabilities 3	190
0x01C	DC4	RO	0x0000.001F	Device Capabilities 4	192
0x030	PBORCTL	R/W	0x0000.7FFD	Power-On and Brown-Out Reset Control	169
0x034	LDOPCTL	R/W	0x0000.0000	LDO Power Control	170
0x040	SRCR0	R/W	0x00000000	Software Reset Control 0	210
0x044	SRCR1	R/W	0x00000000	Software Reset Control 1	211
0x048	SRCR2	R/W	0x00000000	Software Reset Control 2	213
0x050	RIS	RO	0x0000.0000	Raw Interrupt Status	171
0x054	IMC	R/W	0x0000.0000	Interrupt Mask Control	172
0x058	MISC	R/W1C	0x0000.0000	Masked Interrupt Status and Clear	173
0x05C	RESC	R/W	-	Reset Cause	174
0x060	RCC	R/W	0x078E.3AC0	Run-Mode Clock Configuration	175
0x064	PLLCFG	RO	-	XTAL to PLL Translation	179
0x100	RCGC0	R/W	0x00000040	Run Mode Clock Gating Control Register 0	193
0x104	RCGC1	R/W	0x00000000	Run Mode Clock Gating Control Register 1	199
0x108	RCGC2	R/W	0x00000000	Run Mode Clock Gating Control Register 2	205
0x110	SCGC0	R/W	0x00000040	Sleep Mode Clock Gating Control Register 0	195
0x114	SCGC1	R/W	0x00000000	Sleep Mode Clock Gating Control Register 1	201
0x118	SCGC2	R/W	0x00000000	Sleep Mode Clock Gating Control Register 2	206
0x120	DCGC0	R/W	0x00000040	Deep Sleep Mode Clock Gating Control Register 0	197
0x124	DCGC1	R/W	0x00000000	Deep Sleep Mode Clock Gating Control Register 1	203
0x128	DCGC2	R/W	0x00000000	Deep Sleep Mode Clock Gating Control Register 2	208
0x144	DSLPCLKCFG	R/W	0x0780.0000	Deep Sleep Clock Configuration	180
0x150	CLKVCLR	R/W	0x0000.0000	Clock Verification Clear	181
0x160	LDOARST	R/W	0x0000.0000	Allow Unregulated LDO to Reset the Part	182

## 5.5 Register Descriptions

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

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

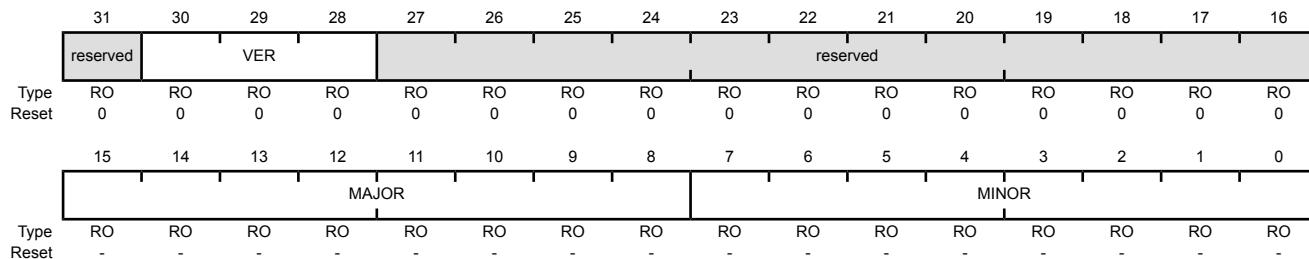
This register identifies the version of the microcontroller. Each microcontroller is uniquely identified by the combined values of the CLASS field in the DID0 register and the PARTNO field in the DID1 register.

### Device Identification 0 (DID0)

Base 0x400F.E000

Offset 0x000

Type 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	0x0	DID0 Version This field defines the DID0 register format version. The version number is numeric. The value of the VER field is encoded as follows:
		Value	Description	
	0x0	Initial DID0 register format definition for Stellaris® Sandstorm-class devices.		
27:16	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.
15:8	MAJOR	RO	-	Major Revision 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:
		Value	Description	
	0x0	Revision A (initial device)		
	0x1	Revision B (first base layer revision)		
	0x2	Revision C (second base layer revision)		
		and so on.		

Bit/Field	Name	Type	Reset	Description
7:0	MINOR	RO	-	Minor Revision This field specifies the minor revision number of the device. The minor revision reflects changes to the metal layers of the design. The MINOR field value is reset when the MAJOR field is changed. This field is numeric and is encoded as follows:
Value Description				
0x0 Initial device, or a major revision update.				
0x1 First metal layer change.				
0x2 Second metal layer change.				
and so on.				

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

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

### Power-On and 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
	BORTIM															BORIOR BORWT
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	0	1

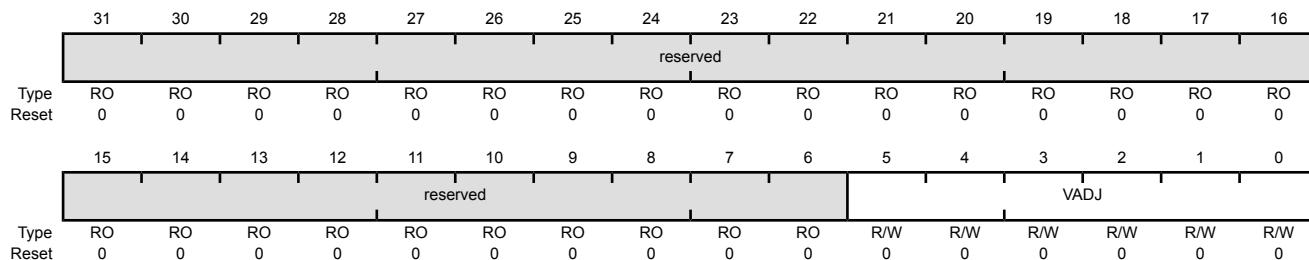
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:2	BORTIM	R/W	0x1FFF	<p>BOR Time Delay</p> <p>This field specifies the number of internal oscillator clocks delayed before the BOR output is resampled if the BORWT bit is set.</p> <p>The width of this field is derived by the <math>t_{BOR}</math> width of 500 <math>\mu</math>s and the internal oscillator (IOSC) frequency of 12 MHz <math>\pm</math> 30%. At +30%, the counter value has to exceed 7,800.</p>
1	BORIOR	R/W	0	<p>BOR Interrupt or Reset</p> <p>This bit controls how a BOR event is signaled to the controller. If set, a reset is signaled. Otherwise, an interrupt is signaled.</p>
0	BORWT	R/W	1	<p>BOR Wait and Check for Noise</p> <p>This bit specifies the response to a brown-out signal assertion if BORIOR is not set.</p> <p>If BORWT is set to 1 and BORIOR is cleared to 0, the controller waits BORTIM IOSC periods and resamples the BOR output. If still asserted, a BOR interrupt is signalled. If no longer asserted, the initial assertion is suppressed (attributable to noise).</p> <p>If BORWT is 0, BOR assertions do not resample the output and any condition is reported immediately if enabled.</p>

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

The VADJ field in this register adjusts the on-chip output voltage ( $V_{OUT}$ ).

#### 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.
5:0	VADJ	R/W	0x0	<p>LDO Output Voltage</p> <p>This field sets the on-chip output voltage. The programming values for the VADJ field are provided below.</p>

Value	$V_{OUT}$ (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															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	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	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 <sub>READY</sub> Timer asserts.
5	CLRIS	RO	0	Current Limit Raw Interrupt Status This bit is set if the LDO's CLE output asserts.
4	IOFRIS	RO	0	Internal Oscillator Fault Raw Interrupt Status This bit is set if an internal oscillator fault is detected.
3	MOFRIS	RO	0	Main Oscillator Fault Raw Interrupt Status This bit is set if a main oscillator fault is detected.
2	LDORIS	RO	0	LDO Power Unregulated Raw Interrupt Status This bit is set if a LDO voltage is unregulated.
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 BORIM bit in the IMC register is set and the BORIOR bit in the PBORCTL register is cleared.
0	PLLFRIS	RO	0	PLL Fault Raw Interrupt Status This bit is set if a PLL fault is detected (stops oscillating).

## 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															PLLLIM
Type	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

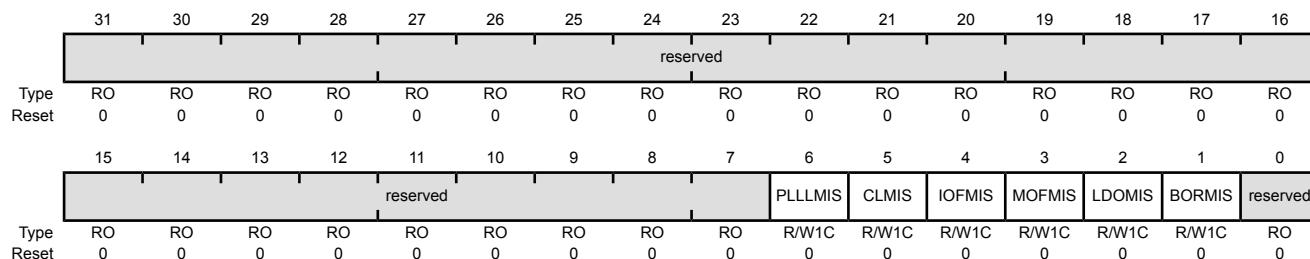
Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PLLLIM	R/W	0	PLL Lock Interrupt Mask This bit specifies whether a PLL Lock interrupt is promoted to a controller interrupt. If set, an interrupt is generated if <b>PLLLRIS</b> in <b>RIS</b> is set; otherwise, an interrupt is not generated.
5	CLIM	R/W	0	Current Limit Interrupt Mask This bit specifies whether a current limit detection is promoted to a controller interrupt. If set, an interrupt is generated if <b>CLRIS</b> is set; otherwise, an interrupt is not generated.
4	IOFIM	R/W	0	Internal Oscillator Fault Interrupt Mask This bit specifies whether an internal oscillator fault detection is promoted to a controller interrupt. If set, an interrupt is generated if <b>IOFRIS</b> is set; otherwise, an interrupt is not generated.
3	MOFIM	R/W	0	Main Oscillator Fault Interrupt Mask This bit specifies whether a main oscillator fault detection is promoted to a controller interrupt. If set, an interrupt is generated if <b>MOFRIS</b> is set; otherwise, an interrupt is not generated.
2	LDOIM	R/W	0	LDO Power Unregulated Interrupt Mask This bit specifies whether an LDO unregulated power situation is promoted to a controller interrupt. If set, an interrupt is generated if <b>LDRIS</b> is set; otherwise, an interrupt is not generated.
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 <b>BORRIS</b> is set; otherwise, an interrupt is not generated.
0	PLLFIM	R/W	0	PLL Fault Interrupt Mask This bit specifies whether a PLL fault detection is promoted to a controller interrupt. If set, an interrupt is generated if <b>PLLFRIS</b> is set; otherwise, an interrupt is not generated.

## 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 171).

### Masked Interrupt Status and Clear (MISC)

Base 0x400F.E000  
Offset 0x058  
Type R/W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:7	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 <sub>READY</sub> timer asserts. The interrupt is cleared by writing a 1 to this bit.
5	CLMIS	R/W1C	0	Current Limit Masked Interrupt Status This bit is set if the LDO's CLE output asserts. The interrupt is cleared by writing a 1 to this bit.
4	IOFMIS	R/W1C	0	Internal Oscillator Fault Masked Interrupt Status This bit is set if an internal oscillator fault is detected. The interrupt is cleared by writing a 1 to this bit.
3	MOFMIS	R/W1C	0	Main Oscillator Fault Masked Interrupt Status This bit is set if a main oscillator fault is detected. The interrupt is cleared by writing a 1 to this bit.
2	LDOMIS	R/W1C	0	LDO Power Unregulated Masked Interrupt Status This bit is set if LDO power is unregulated. The interrupt is cleared by writing a 1 to this bit.
1	BORMIS	R/W1C	0	BOR Masked Interrupt Status This bit is the masked interrupt status for any brown-out conditions. If set, a brown-out condition was detected. An interrupt is reported if the BORIM bit in the IMC register is set and the BORIOR bit in the <b>PBORCTL</b> register is cleared. The interrupt is cleared by writing a 1 to this bit.
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 field specifies the cause of the reset event to software. The reset value is determined by the cause of the reset. When an external reset is the cause (EXT is set), all other reset bits are cleared. However, if the reset is due to any other cause, the remaining bits are sticky, allowing software to see all causes.

### 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															
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	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-

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.
5	LDO	R/W	-	LDO Reset When set, indicates the LDO circuit has lost regulation and has generated a reset event.
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 0x078E.3AC0

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

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 (RCCGn)</b> registers are used when the controller enters a sleep mode.</p> <p>The <b>RCCGn</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>
26:23	SYSDIV	R/W	0xF	<p>System Clock Divisor</p> <p>Specifies which divisor is used to generate the system clock from either the PLL output or the oscillator source (depending on how the <b>BYPASS</b> bit in this register is configured). See Table 5-5 on page 162 for bit encodings.</p> <p>The PLL VCO frequency is 200 MHz.</p> <p>If the <b>SYSDIV</b> value is less than <b>MINSYSDIV</b> (see page 186), and the PLL is being used, then the <b>MINSYSDIV</b> value is used as the divisor.</p> <p>If the PLL is not being used, the <b>SYSDIV</b> value can be less than <b>MINSYSDIV</b>.</p>
22	USESYSDIV	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> <p>If the <b>USERCC2</b> bit in the <b>RCC2</b> register is set, then the <b>SYSDIV2</b> field in the <b>RCC2</b> register is used as the system clock divider rather than the <b>SYSDIV</b> field in this register.</p>

Bit/Field	Name	Type	Reset	Description
21	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.
20	USEPWMDIV	R/W	0	Enable PWM Clock Divisor Use the PWM clock divider as the source for the PWM clock.
19:17	PWMDIV	R/W	0x7	PWM Unit Clock Divisor  This field specifies the binary divisor used to predivide the system clock down for use as the timing reference for the PWM module. This clock is only power 2 divide and rising edge is synchronous without phase shift from the system clock.  Value Divisor 0x0 /2 0x1 /4 0x2 /8 0x3 /16 0x4 /32 0x5 /64 0x6 /64 0x7 /64 (default)
16:14	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	PLL Power Down  This bit connects to the PLL PWRDN input. The reset value of 1 powers down the PLL. See Table 5-7 on page 178 for PLL mode control.
12	OEN	R/W	1	PLL Output Enable  This bit specifies whether the PLL output driver is enabled. If cleared, the driver transmits the PLL clock to the output. Otherwise, the PLL clock does not oscillate outside the PLL module.  <b>Note:</b> Both PWRDN and OEN must be cleared to run the PLL.
11	BYPASS	R/W	1	PLL Bypass  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.  See Table 5-5 on page 162 for programming guidelines.  <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.
10	PLLVER	R/W	0	PLL Verification  This bit controls the PLL verification timer function. If set, the verification timer is enabled and an interrupt is generated if the PLL becomes inoperative. Otherwise, the verification timer is not enabled.

Bit/Field	Name	Type	Reset	Description																																																			
9:6	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> <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>0x0</td> <td>1.000</td> <td>reserved</td> </tr> <tr> <td>0x1</td> <td>1.8432</td> <td>reserved</td> </tr> <tr> <td>0x2</td> <td>2.000</td> <td>reserved</td> </tr> <tr> <td>0x3</td> <td>2.4576</td> <td>reserved</td> </tr> <tr> <td>0x4</td> <td></td> <td>3.579545 MHz</td> </tr> <tr> <td>0x5</td> <td></td> <td>3.6864 MHz</td> </tr> <tr> <td>0x6</td> <td></td> <td>4 MHz</td> </tr> <tr> <td>0x7</td> <td></td> <td>4.096 MHz</td> </tr> <tr> <td>0x8</td> <td></td> <td>4.9152 MHz</td> </tr> <tr> <td>0x9</td> <td></td> <td>5 MHz</td> </tr> <tr> <td>0xA</td> <td></td> <td>5.12 MHz</td> </tr> <tr> <td>0xB</td> <td></td> <td>6 MHz (reset value)</td> </tr> <tr> <td>0xC</td> <td></td> <td>6.144 MHz</td> </tr> <tr> <td>0xD</td> <td></td> <td>7.3728 MHz</td> </tr> <tr> <td>0xE</td> <td></td> <td>8 MHz</td> </tr> <tr> <td>0xF</td> <td></td> <td>8.192 MHz</td> </tr> </tbody> </table>	Value	Crystal Frequency (MHz) Not Using the PLL	Crystal Frequency (MHz) Using the PLL	0x0	1.000	reserved	0x1	1.8432	reserved	0x2	2.000	reserved	0x3	2.4576	reserved	0x4		3.579545 MHz	0x5		3.6864 MHz	0x6		4 MHz	0x7		4.096 MHz	0x8		4.9152 MHz	0x9		5 MHz	0xA		5.12 MHz	0xB		6 MHz (reset value)	0xC		6.144 MHz	0xD		7.3728 MHz	0xE		8 MHz	0xF		8.192 MHz
Value	Crystal Frequency (MHz) Not Using the PLL	Crystal Frequency (MHz) Using the PLL																																																					
0x0	1.000	reserved																																																					
0x1	1.8432	reserved																																																					
0x2	2.000	reserved																																																					
0x3	2.4576	reserved																																																					
0x4		3.579545 MHz																																																					
0x5		3.6864 MHz																																																					
0x6		4 MHz																																																					
0x7		4.096 MHz																																																					
0x8		4.9152 MHz																																																					
0x9		5 MHz																																																					
0xA		5.12 MHz																																																					
0xB		6 MHz (reset value)																																																					
0xC		6.144 MHz																																																					
0xD		7.3728 MHz																																																					
0xE		8 MHz																																																					
0xF		8.192 MHz																																																					
5:4	OSCSRC	R/W	0x0	<p>Oscillator Source</p> <p>Selects the input source for the OSC. The values are:</p> <table> <thead> <tr> <th>Value</th> <th>Input Source</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>MOSC Main oscillator (default)</td> </tr> <tr> <td>0x1</td> <td>IOSC Internal oscillator</td> </tr> <tr> <td>0x2</td> <td>IOSC/4 Internal oscillator / 4 (this is necessary if used as input to PLL)</td> </tr> <tr> <td>0x3</td> <td>reserved</td> </tr> </tbody> </table>	Value	Input Source	0x0	MOSC Main oscillator (default)	0x1	IOSC Internal oscillator	0x2	IOSC/4 Internal oscillator / 4 (this is necessary if used as input to PLL)	0x3	reserved																																									
Value	Input Source																																																						
0x0	MOSC Main oscillator (default)																																																						
0x1	IOSC Internal oscillator																																																						
0x2	IOSC/4 Internal oscillator / 4 (this is necessary if used as input to PLL)																																																						
0x3	reserved																																																						
3	IOSCVER	R/W	0	<p>Internal Oscillator Verification Timer</p> <p>This bit controls the internal oscillator verification timer function. If set, the verification timer is enabled and an interrupt is generated if the timer becomes inoperative. Otherwise, the verification timer is not enabled.</p>																																																			
2	MOSCVER	R/W	0	<p>Main Oscillator Verification Timer</p> <p>This bit controls the main oscillator verification timer function. If set, the verification timer is enabled and an interrupt is generated if the timer becomes inoperative. Otherwise, the verification timer is not enabled.</p>																																																			

---

Bit/Field	Name	Type	Reset	Description
1	IOSCDIS	R/W	0	Internal Oscillator Disable 0: Internal oscillator (IOSC) is enabled. 1: Internal oscillator is disabled.
0	MOSCDIS	R/W	0	Main Oscillator Disable 0: Main oscillator is enabled (default). 1: Main oscillator is disabled .

**Table 5-7. PLL Mode Control**

PWRDN	OEN	Mode
1	X	Power down
0	0	Normal

## 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 175).

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

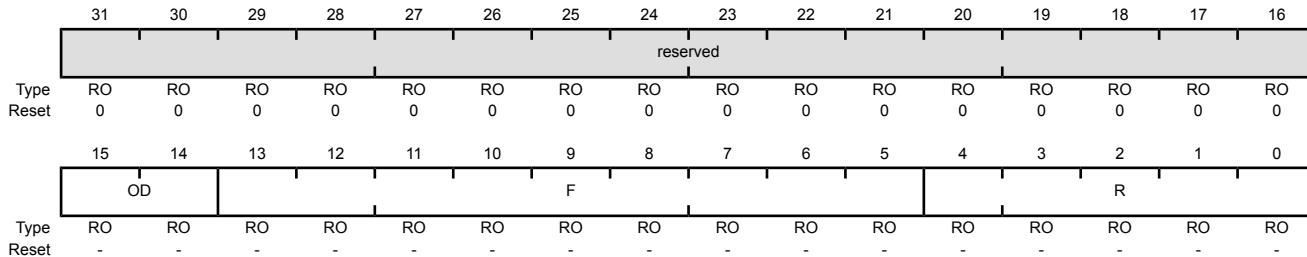
$$\text{PLLFreq} = \text{OSCFreq} * (\text{F} + 2) / (\text{R} + 2)$$

### XTAL to PLL Translation (PLLCFG)

Base 0x400F.E000

Offset 0x064

Type RO, reset -



Bit/Field	Name	Type	Reset	Description
31:16	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.
15:14	OD	RO	-	PLL OD Value This field specifies the value supplied to the PLL's OD input.
				Value Description 0x0 Divide by 1 0x1 Divide by 2 0x2 Divide by 4 0x3 Reserved
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: Deep Sleep Clock Configuration (DSLPCLKCFG), offset 0x144

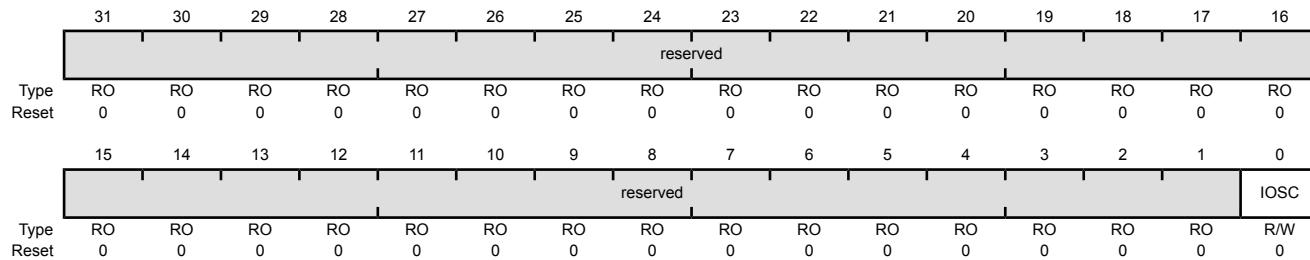
This register is used to automatically switch from the main oscillator to the internal oscillator when entering Deep-Sleep mode. The system clock source is the main oscillator by default. When this register is set, the internal oscillator is powered up and the main oscillator is powered down. When the 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.

### Deep Sleep Clock Configuration (DSLPCLKCFG)

Base 0x400F.E000

Offset 0x144

Type R/W, reset 0x0780.0000



## Register 11: Clock Verification Clear (CLKVCLR), offset 0x150

This register is provided as a means of clearing the clock verification circuits by software. Since the clock verification circuits force a known good clock to control the process, the controller is allowed the opportunity to solve the problem and clear the verification fault. This register clears all clock verification faults. To clear a clock verification fault, the VERCLR bit must be set and then cleared by software. This bit is not self-clearing.

### Clock Verification Clear (CLKVCLR)

Base 0x400F.E000  
Offset 0x150  
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															VERCLR
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

Bit/Field	Name	Type	Reset	Description
31:1	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.
0	VERCLR	R/W	0	Clock Verification Clear Clears clock verification faults.

## Register 12: Allow Unregulated LDO to Reset the Part (LDOARST), offset 0x160

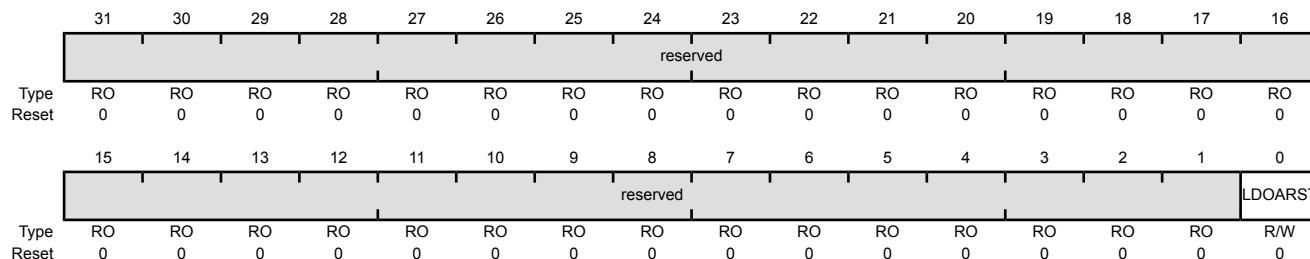
This register is provided as a means of allowing the LDO to reset the part if the voltage goes unregulated. Use this register to choose whether to automatically reset the part if the LDO goes unregulated, based on the design tolerance for LDO fluctuation.

### Allow Unregulated LDO to Reset the Part (LDOARST)

Base 0x400F.E000

Offset 0x160

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	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.
0	LDOARST	R/W	0	LDO Reset When set, allows unregulated LDO output to reset the part.

## Register 13: Device Identification 1 (DID1), offset 0x004

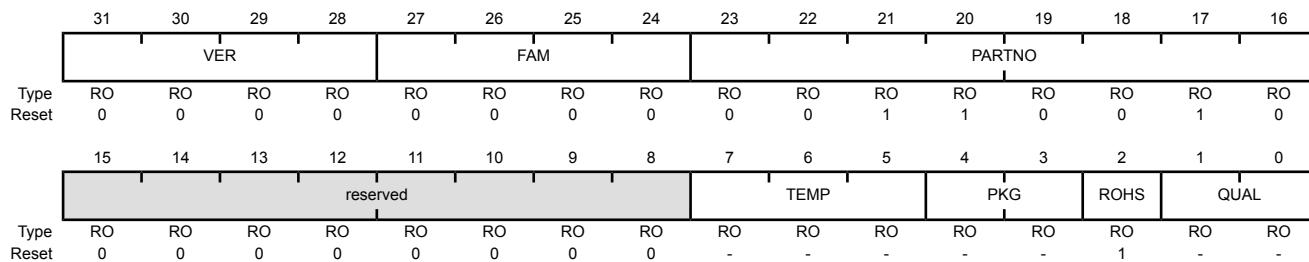
This register identifies the device family, part number, temperature range, pin count, and package type. Each microcontroller is uniquely identified by the combined values of the CLASS field in the **DID0** register and the PARTNO field in the **DID1** register.

### Device Identification 1 (DID1)

Base 0x400F.E000

Offset 0x004

Type RO, reset -



Bit/Field	Name	Type	Reset	Description				
31:28	VER	RO	0x0	<p>DID1 Version</p> <p>This field defines the <b>DID1</b> register format version. The version number is numeric. The value of the <b>VER</b> field is encoded as follows (all other encodings are reserved):</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Initial <b>DID1</b> register format definition, indicating a Stellaris LM3Snnn device.</td> </tr> </tbody> </table>	Value	Description	0x0	Initial <b>DID1</b> register format definition, indicating a Stellaris LM3Snnn device.
Value	Description							
0x0	Initial <b>DID1</b> register format definition, indicating a Stellaris LM3Snnn device.							
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> <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	0x32	<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> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x32</td> <td>LM3S811</td> </tr> </tbody> </table>	Value	Description	0x32	LM3S811
Value	Description							
0x32	LM3S811							
15:8	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.				

Bit/Field	Name	Type	Reset	Description								
7:5	TEMP	RO	-	<p>Temperature Range</p> <p>This field specifies the temperature rating of the device. The value is encoded as follows (all other encodings are reserved):</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Commercial temperature range (0°C to 70°C)</td></tr> <tr> <td>0x1</td><td>Industrial temperature range (-40°C to 85°C)</td></tr> <tr> <td>0x2</td><td>Extended temperature range (-40°C to 105°C)</td></tr> </tbody> </table>	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)
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	-	<p>Package Type</p> <p>This field specifies the package type. The value is encoded as follows (all other encodings are reserved):</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>28-pin SOIC package</td></tr> <tr> <td>0x1</td><td>48-pin LQFP package</td></tr> <tr> <td>0x3</td><td>48-pin QFN package</td></tr> </tbody> </table>	Value	Description	0x0	28-pin SOIC package	0x1	48-pin LQFP package	0x3	48-pin QFN package
Value	Description											
0x0	28-pin SOIC package											
0x1	48-pin LQFP package											
0x3	48-pin QFN package											
2	ROHS	RO	1	<p>RoHS-Compliance</p> <p>This bit specifies whether the device is RoHS-compliant. A 1 indicates the part is RoHS-compliant.</p>								
1:0	QUAL	RO	-	<p>Qualification Status</p> <p>This field specifies the qualification status of the device. The value is encoded as follows (all other encodings are reserved):</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Engineering Sample (unqualified)</td></tr> <tr> <td>0x1</td><td>Pilot Production (unqualified)</td></tr> <tr> <td>0x2</td><td>Fully Qualified</td></tr> </tbody> </table>	Value	Description	0x0	Engineering Sample (unqualified)	0x1	Pilot Production (unqualified)	0x2	Fully Qualified
Value	Description											
0x0	Engineering Sample (unqualified)											
0x1	Pilot Production (unqualified)											
0x2	Fully Qualified											

## Register 14: Device Capabilities 0 (DC0), offset 0x008

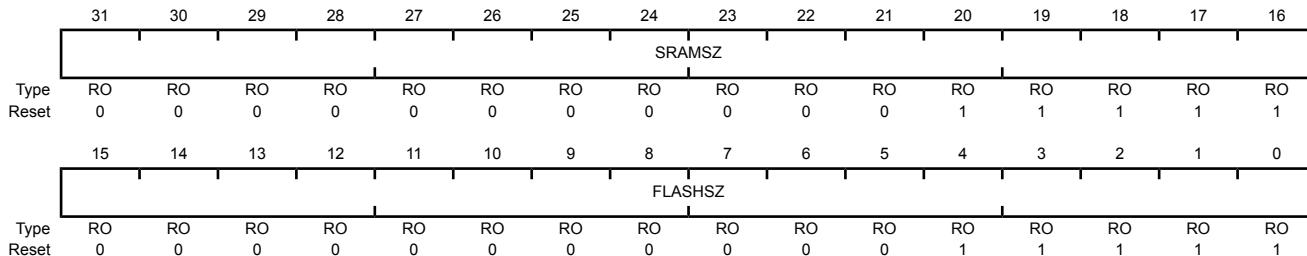
This register is predefined by the part and can be used to verify features.

### Device Capabilities 0 (DC0)

Base 0x400F.E000

Offset 0x008

Type RO, reset 0x001F.001F



Bit/Field                  Name                  Type                  Reset                  Description

31:16                  SRAMSZ                  RO                  0x001F                  SRAM Size  
Indicates the size of the on-chip SRAM memory.

Value                  Description  
0x001F                  8 KB of SRAM

15:0                  FLASHSZ                  RO                  0x001F                  Flash Size  
Indicates the size of the on-chip flash memory.

Value                  Description  
0x001F                  64 KB of Flash

## Register 15: Device Capabilities 1 (DC1), offset 0x010

This register provides a list of features available in the system. The Stellaris family uses this register format to indicate the availability of the following family features in the specific device: PWM, ADC, Watchdog timer, and debug capabilities. This register also indicates the maximum clock frequency and maximum ADC sample rate. The format of this register is consistent with the **RCGC0**, **SCGC0**, and **DCCG0** clock control registers and the **SRCCR0** software reset control register.

### Device Capabilities 1 (DC1)

Base 0x400F.E000

Offset 0x010

Type RO, reset 0x0011.32BF

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

Bit/Field	Name	Type	Reset	Description
31:21	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.
20	PWM	RO	1	PWM Module Present When set, indicates that the PWM module is present.
19:17	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 <b>SYSDIV</b> 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 Indicates the maximum rate at which the ADC samples data.  Value Description 0x2 500K samples/second

Bit/Field	Name	Type	Reset	Description
7	MPU	RO	1	MPU Present When set, indicates that the Cortex-M3 Memory Protection Unit (MPU) module is present. See the "Cortex-M3 Peripherals" chapter in the Stellaris Data Sheet for details on the MPU.
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.
5	TEMPSNS	RO	1	Temp Sensor Present When set, indicates that the on-chip temperature sensor is present.
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 16: Device Capabilities 2 (DC2), offset 0x014

This register provides a list of features available in the system. The Stellaris family uses this register format to indicate the availability of the following family features in the specific device: Analog Comparators, General-Purpose Timers, I2Cs, QEIs, SSIs, and UARTs. The format of this register is consistent with the **RCGC1**, **SCGC1**, and **DCGC1** clock control registers and the **SRCR1** software reset control register.

### Device Capabilities 2 (DC2)

Base 0x400F.E000

Offset 0x014

Type RO, reset 0x0107.1013

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Type	RO	COMP0		reserved					TIMER2	TIMER1	TIMER0						
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Type	RO		reserved			SSI0		reserved	UART1	UART0							
Reset	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	1	

Bit/Field	Name	Type	Reset	Description
31:25	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.
24	COMP0	RO	1	Analog Comparator 0 Present When set, indicates that analog comparator 0 is present.
23:19	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.
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.
15:13	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.

Bit/Field	Name	Type	Reset	Description
3:2	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	UART1	RO	1	UART1 Present When set, indicates that UART module 1 is present.
0	UART0	RO	1	UART0 Present When set, indicates that UART module 0 is present.

## Register 17: Device Capabilities 3 (DC3), offset 0x018

This register provides a list of features available in the system. The Stellaris family uses this register format to indicate the availability of the following family features in the specific device: Analog Comparator I/Os, CCP I/Os, ADC I/Os, and PWM I/Os.

### Device Capabilities 3 (DC3)

Base 0x400F.E000  
Offset 0x018  
Type RO, reset 0xBF0F.01FF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	32KHZ	reserved	CCP5	CCP4	CCP3	CCP2	CCP1	CCP0	reserved	reserved	reserved	reserved	ADC3	ADC2	ADC1	ADC0
Reset	RO 1	RO 0	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 0	RO 0	RO 0	RO 0	RO 1	RO 1	RO 1	RO 1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	C0O	C0PLUS	C0MINUS	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0
Reset	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1

Bit/Field	Name	Type	Reset	Description
31	32KHZ	RO	1	32KHz Input Clock Available When set, indicates the 32KHz pin or 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.
23:20	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	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.

Bit/Field	Name	Type	Reset	Description
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.
15:9	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	C0O	RO	1	C0o Pin Present When set, indicates that the analog comparator 0 output pin is present.
7	C0PLUS	RO	1	C0+ Pin Present When set, indicates that the analog comparator 0 (+) input pin is present.
6	C0MINUS	RO	1	C0- Pin Present When set, indicates that the analog comparator 0 (-) input pin is present.
5	PWM5	RO	1	PWM5 Pin Present When set, indicates that the PWM pin 5 is present.
4	PWM4	RO	1	PWM4 Pin Present When set, indicates that the PWM pin 4 is present.
3	PWM3	RO	1	PWM3 Pin Present When set, indicates that the PWM pin 3 is present.
2	PWM2	RO	1	PWM2 Pin Present When set, indicates that the PWM pin 2 is present.
1	PWM1	RO	1	PWM1 Pin Present When set, indicates that the PWM pin 1 is present.
0	PWM0	RO	1	PWM0 Pin Present When set, indicates that the PWM pin 0 is present.

## Register 18: Device Capabilities 4 (DC4), offset 0x01C

This register provides a list of features available in the system. The Stellaris family uses this register format to indicate the availability of GPIOs in the specific device. The format of this register is consistent with the **RCGC2**, **SCGC2**, and **DCGC2** clock control registers and the **SRCR2** software reset control register.

### Device Capabilities 4 (DC4)

Base 0x400F.E000  
Offset 0x01C  
Type RO, reset 0x0000.001F

	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	1	1	1	1

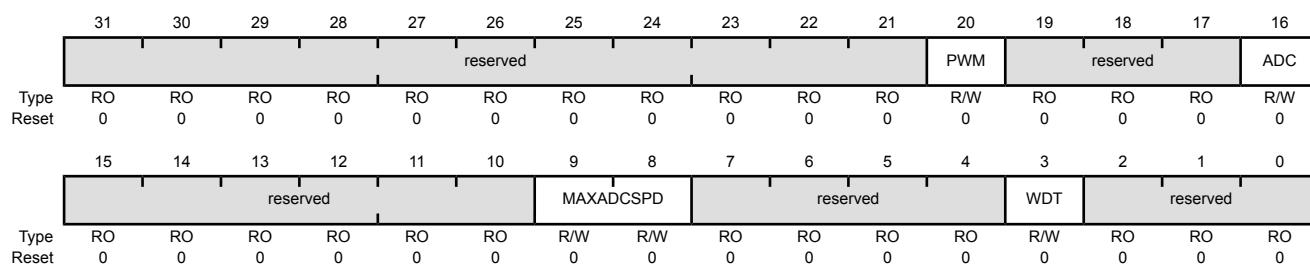
Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	GPIOE	RO	1	GPIO Port E Present When set, indicates that 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 19: 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



Bit/Field	Name	Type	Reset	Description
31:21	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.
20	PWM	R/W	0	PWM Clock Gating Control  This bit controls the clock gating for the PWM module. 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.
19:17	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.

Bit/Field	Name	Type	Reset	Description								
9:8	MAXADCSPD	R/W	0	<p>ADC Sample Speed</p> <p>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:</p> <table> <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											
7:4	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	<p>WDT Clock Gating Control</p> <p>This bit controls the clock gating for the WDT module. 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.</p>								
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 20: 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 0x400FE000  
Offset 0x110  
Type R/W, reset 0x00000040

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved												PWM	reserved		ADC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	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				WDT	reserved		
Type	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	RO	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
31:21	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.
20	PWM	R/W	0	PWM Clock Gating Control  This bit controls the clock gating for the PWM module. 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.
19:17	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.

Bit/Field	Name	Type	Reset	Description								
9:8	MAXADCSPD	R/W	0	<p>ADC Sample Speed</p> <p>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:</p> <table><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											
7:4	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	<p>WDT Clock Gating Control</p> <p>This bit controls the clock gating for the WDT module. 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.</p>								
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 21: 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 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.

### 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												PWM	reserved		ADC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	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												WDT	reserved		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	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
31:21	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.
20	PWM	R/W	0	PWM Clock Gating Control  This bit controls the clock gating for the PWM module. 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.
19:17	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.
15:4	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	WDT	R/W	0	<b>WDT Clock Gating Control</b> This bit controls the clock gating for the WDT module. 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.
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 22: 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
Type	RO	RO	RO	RO	RO	RO	RO	R/W	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
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RO	RO	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
31:25	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.
24	COMP0	R/W	0	Analog Comparator 0 Clock Gating  This bit controls the clock gating for analog comparator 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.
23:19	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.
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.

Bit/Field	Name	Type	Reset	Description
16	TIMER0	R/W	0	<p><b>Timer 0 Clock Gating Control</b></p> <p>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.</p>
15:13	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>
12	I2C0	R/W	0	<p><b>I2C0 Clock Gating Control</b></p> <p>This bit controls the clock gating for I2C 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.</p>
11:5	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>
4	SSI0	R/W	0	<p><b>SSI0 Clock Gating Control</b></p> <p>This bit controls the clock gating for SSI 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.</p>
3:2	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>
1	UART1	R/W	0	<p><b>UART1 Clock Gating Control</b></p> <p>This bit controls the clock gating for UART 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.</p>
0	UART0	R/W	0	<p><b>UART0 Clock Gating Control</b></p> <p>This bit controls the clock gating for UART 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.</p>

## Register 23: 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 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 **DGCG1** 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 0x400FE000  
Offset 0x114  
Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	RO	RO	RO	RO	RO	RO	R/W	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
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RO	RO	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
	reserved			I2C0	reserved						SSI0	reserved		UART1	UART0	
Type	RO	RO	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
31:25	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.
24	COMP0	R/W	0	Analog Comparator 0 Clock Gating  This bit controls the clock gating for analog comparator 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.
23:19	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.
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.

Bit/Field	Name	Type	Reset	Description
16	TIMER0	R/W	0	<p><b>Timer 0 Clock Gating Control</b></p> <p>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.</p>
15:13	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>
12	I2C0	R/W	0	<p><b>I2C0 Clock Gating Control</b></p> <p>This bit controls the clock gating for I2C 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.</p>
11:5	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>
4	SSI0	R/W	0	<p><b>SSI0 Clock Gating Control</b></p> <p>This bit controls the clock gating for SSI 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.</p>
3:2	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>
1	UART1	R/W	0	<p><b>UART1 Clock Gating Control</b></p> <p>This bit controls the clock gating for UART 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.</p>
0	UART0	R/W	0	<p><b>UART0 Clock Gating Control</b></p> <p>This bit controls the clock gating for UART 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.</p>

## Register 24: 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 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.

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

Base 0x400FE000  
Offset 0x124  
Type R/W, reset 0x00000000

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

Bit/Field	Name	Type	Reset	Description
31:25	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.
24	COMP0	R/W	0	Analog Comparator 0 Clock Gating  This bit controls the clock gating for analog comparator 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.
23:19	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.
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.

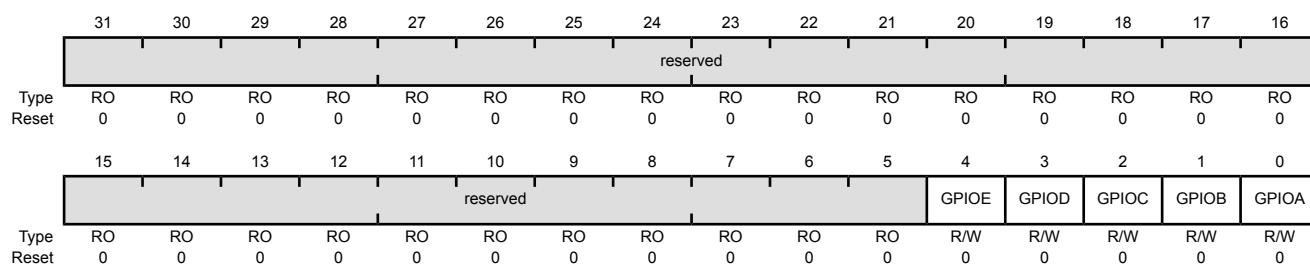
Bit/Field	Name	Type	Reset	Description
16	TIMER0	R/W	0	<p><b>Timer 0 Clock Gating Control</b></p> <p>This bit controls the clock gating for General-Purpose Timer module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.</p>
15:13	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>
12	I2C0	R/W	0	<p><b>I2C0 Clock Gating Control</b></p> <p>This bit controls the clock gating for I2C 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.</p>
11:5	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>
4	SSI0	R/W	0	<p><b>SSI0 Clock Gating Control</b></p> <p>This bit controls the clock gating for SSI 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.</p>
3:2	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>
1	UART1	R/W	0	<p><b>UART1 Clock Gating Control</b></p> <p>This bit controls the clock gating for UART 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.</p>
0	UART0	R/W	0	<p><b>UART0 Clock Gating Control</b></p> <p>This bit controls the clock gating for UART 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.</p>

## Register 25: 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 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. **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



Bit/Field	Name	Type	Reset	Description
31:5	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 unclocked and disabled. If the unit is unclocked, 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 unclocked and disabled. If the unit is unclocked, 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 unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
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 unclocked and disabled. If the unit is unclocked, 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 unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

## Register 26: 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 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. **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															
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	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:5	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 unclocked and disabled. If the unit is unclocked, 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 unclocked and disabled. If the unit is unclocked, 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 unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
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 unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

Bit/Field	Name	Type	Reset	Description
0	GPIOA	R/W	0	<p>Port A Clock Gating Control</p> <p>This bit controls the clock gating for Port A. 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.</p>

## Register 27: 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 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. **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															
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	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:5	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 unclocked and disabled. If the unit is unclocked, 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 unclocked and disabled. If the unit is unclocked, 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 unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
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 unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

Bit/Field	Name	Type	Reset	Description
0	GPIOA	R/W	0	<p>Port A Clock Gating Control</p> <p>This bit controls the clock gating for Port A. 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.</p>

## Register 28: 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																
Type	RO	R/W	RO	RO	RO	R/W										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	RO	R/W	RO	RO	RO	R/W										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	RO	R/W	RO	RO	RO	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
31:21	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.
20	PWM	R/W	0	PWM Reset Control Reset control for PWM module.
19:17	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.
15:4	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 29: 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

Bit/Field	Name	Type	Reset	Description
31:25	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.
24	COMP0	R/W	0	Analog Comp 0 Reset Control Reset control for analog comparator 0.
23:19	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.
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.
15:13	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.
3:2	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.

Bit/Field	Name	Type	Reset	Description
1	UART1	R/W	0	UART1 Reset Control Reset control for UART unit 1.
0	UART0	R/W	0	UART0 Reset Control Reset control for UART unit 0.

## Register 30: 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											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	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
31:5	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.

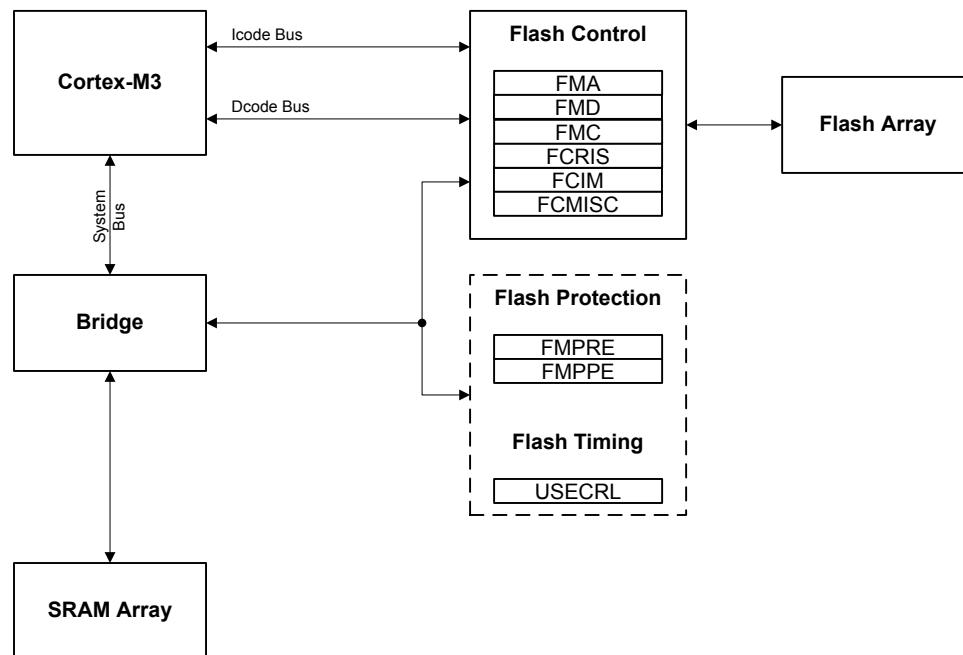
# 6 Internal Memory

The LM3S811 microcontroller comes with 8 KB of bit-banded SRAM and 64 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.

## 6.1 Block Diagram

Figure 6-1 on page 214 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 6-1. Flash Block Diagram**



## 6.2 Functional Description

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

### 6.2.1 SRAM Memory

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, see “Bit-Banding” on page 67.

## 6.2.2 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.

See also “Serial Flash Loader” on page 563 for a preprogrammed flash-resident utility used to download code to the flash memory of a device without the use of a debug interface.

### 6.2.2.1 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.

### 6.2.2.2 Flash Memory Protection

The user is provided two forms of flash protection per 2-KB flash blocks in two 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 a bit is set, the corresponding block may be executed or read by software or debuggers. If a bit is cleared, the corresponding block may only be executed, and contents of the memory block are prohibited from being read as data.

The policies may be combined as shown in Table 6-1 on page 215.

**Table 6-1. 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.

**Table 6-1. Flash Protection Policy Combinations (continued)**

FMPPEn	FMPREN	Protection
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.

A Flash memory access that attempts to read a read-protected block (**FMPREN** bit is set) is prohibited and generates a bus fault. A Flash memory access that attempts to program or erase a program-protected block (**FMPPEn** bit is set) is prohibited and can optionally generate an interrupt (by setting the **AMASK** bit in the **Flash Controller Interrupt Mask (FCIM)** register) 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. These settings create a policy of open access and programmability. The register bits may be changed by clearing 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. The changes are committed using the **Flash Memory Control (FMC)** register.

### 6.2.2.3 Execute-Only Protection

Execute-only protection prevents both modification and visibility to a protected flash block. This mode is intended to be used in situations where a device requires debug capability, yet portions of the application space must be protected from external access. An example of this is a company who wishes to sell Stellaris devices with their proprietary software pre-programmed, yet allow the end user to add custom code to an unprotected region of the flash (such as a motor control module with a customizable motor configuration section in flash).

Literal data introduces a complication to the protection mechanism. When C code is compiled and linked, literal data (constants, and so on) is typically placed in the text section, between functions, by the compiler. The literal data is accessed at run time through the use of the LDR instruction, which loads the data from memory using a PC-relative memory address. The execution of the LDR instruction generates a read transaction across the Cortex-M3's DCode bus, which is subject to the execute-only protection mechanism. If the accessed block is marked as execute only, the transaction is blocked, and the processor is prevented from loading the constant data and, therefore, inhibiting correct execution. Therefore, using execute-only protection requires that literal data be handled differently. There are three ways to address this:

1. Use a compiler that allows literal data to be collected into a separate section that is put into one or more read-enabled flash blocks. Note that the LDR instruction may use a PC-relative address—in which case the literal pool cannot be located outside the span of the offset—or the software may reserve a register to point to the base address of the literal pool and the LDR offset is relative to the beginning of the pool.
2. Use a compiler that generates literal data from arithmetic instruction immediate data and subsequent computation.
3. Use method 1 or 2, but in assembly language, if the compiler does not support either method.

#### 6.2.2.4 Read-Only Protection

Read-only protection prevents the contents of the flash block from being re-programmed, while still allowing the content to be read by processor or the debug interface. Note that if a **FMPREn** bit is cleared, all read accesses to the Flash memory block are disallowed, including any data accesses. Care must be taken not to store required data in a Flash memory block that has the associated **FMPREn** bit cleared.

The read-only mode does not prevent read access to the stored program, but it does provide protection against accidental (or malicious) erasure or programming. Read-only is especially useful for utilities like the boot loader when the debug interface is permanently disabled. In such combinations, the boot loader, which provides access control to the Flash memory, is protected from being erased or modified.

#### 6.2.2.5 Permanently Disabling Debug

For extremely sensitive applications, the debug interface to the processor and peripherals can be permanently disabled, blocking all accesses to the device through the JTAG or SWD interfaces. With the debug interface disabled, it is still possible to perform standard IEEE instructions (such as boundary scan operations), but access to the processor and peripherals is blocked.

The two most-significant bits of the **FMPRE** register are the **DBG** bits, and control whether or not the debug interface is turned on or off. Since the **DBG** bits are part of the **FMPRE** register, the user loses the capability to mark the upper two flash blocks in a 64 KB flash device as execute-only.

The debug interface should not be permanently disabled without providing some mechanism—such as the boot loader—to provide customer-installable updates or bug fixes. Disabling the debug interface is permanent and cannot be reversed.

#### 6.2.2.6 Interrupts

The Flash memory controller can generate interrupts when the following conditions are observed:

- Programming Interrupt - signals when a program or erase action is complete.
- Access Interrupt - signals when a program or erase action has been attempted on a 2-kB block of memory that is protected by its corresponding **FMPPEn** bit.

The interrupt events that can trigger a controller-level interrupt are defined in the **Flash Controller Masked Interrupt Status (FCMIS)** register (see page 226) by setting the corresponding **MASK** bits. If interrupts are not used, the raw interrupt status is always visible via the **Flash Controller Raw Interrupt Status (FCRIS)** register (see page 225).

Interrupts are always cleared (for both the **FCMIS** and **FCRIS** registers) by writing a 1 to the corresponding bit in the **Flash Controller Masked Interrupt Status and Clear (FCMISC)** register (see page 227).

#### 6.2.2.7 Flash Memory Protection by Disabling Debug Access

Flash memory may also be protected by permanently disabling access to the Debug Access Port (DAP) through the JTAG and SWD interfaces. Access is disabled by clearing the **DBG** field of the **FMPRE** register.

If the **DBG** field in the **Flash Memory Protection Read Enable (FMPRE)** register is programmed to 0x2, access to the DAP is enabled through the JTAG and SWD interfaces. If clear, access to the DAP is disabled. The **DBG** field programming becomes permanent and irreversible after a commit sequence is performed.

In the initial state provided from the factory, access is enabled in order to facilitate code development and debug. Access to the DAP may be disabled at the end of the manufacturing flow, once all tests have passed and software has been loaded. This change does not take effect until the next power-up of the device. Note that it is recommended that disabling access to the DAP be combined with a mechanism for providing end-user installable updates (if necessary) such as the Stellaris boot loader.

**Important:** Once the `DBG` field is cleared and committed, this field can never be restored to the factory-programmed value—which means the JTAG/SWD interface to the debug module can never be re-enabled. This sequence does NOT disable the JTAG controller, it only disables the access of the DAP through the JTAG or SWD interfaces. The JTAG interface remains functional and access to the Test Access Port remains enabled, allowing the user to execute the IEEE JTAG-defined instructions (for example, to perform boundary scan operations).

When using the **FMPRE** bits to protect Flash memory from being read as data (to mark sets of 2-KB blocks of Flash memory as execute-only), these one-time-programmable bits should be written at the same time that the debug disable bits are programmed. Mechanisms to execute the one-time code sequence to disable all debug access include:

- Selecting the debug disable option in the Stellaris boot loader
- Loading the debug disable sequence into SRAM and running it once from SRAM after programming the final end application code into Flash memory

## 6.3 Flash Memory Initialization and Configuration

This section shows examples for using the flash controller to perform various operations on the contents of the flash memory.

### 6.3.1 Changing Flash Protection Bits

As discussed in “Flash Memory Protection” on page 215, changes to the protection bits must be committed before they take effect. The sequence below is used change and commit a block protection bit in the **FMPRE** or **FMPPE** registers. The sequence to change and commit a bit in software is as follows:

1. The **Flash Memory Protection Read Enable (FMPRE)** and **Flash Memory Protection Program Enable (FMPPE)** registers are written, changing the intended bit(s). The action of these changes can be tested by software while in this state.
2. The **Flash Memory Address (FMA)** register (see page 221) bit 0 is set to 1 if the **FMPPE** register is to be committed; otherwise, a 0 commits the **FMPRE** register.
3. The **Flash Memory Control (FMC)** register (see page 223) is written with the `COMT` bit set. This initiates a write sequence and commits the changes.

There is a special sequence to change and commit the `DBG` bits in the **Flash Memory Protection Read Enable (FMPRE)** register. This sequence also sets and commits any changes from 1 to 0 in the block protection bits (for execute-only) in the **FMPRE** register.

1. The **Flash Memory Protection Read Enable (FMPRE)** register is written, changing the intended bit(s). The action of these changes can be tested by software while in this state.
2. The **Flash Memory Address (FMA)** register (see page 221) is written with a value of 0x900.

3. The **Flash Memory Control (FMC)** register (see page 223) is written with the COMT bit set. This initiates a write sequence and commits the changes.

Below is an example code sequence to permanently disable the JTAG and SWD interface to the debug module using DriverLib:

```
#include "hw_types.h"
#include "hw_flash.h"
void
permanently_disable_jtag_swd(void)
{
    //
    // Clear the DBG field of the FMPRE register. Note that the value
    // used in this instance does not affect the state of the BlockN
    // bits, but were the value different, all bits in the FMPRE are
    // affected by this function!
    //
    HWREG(FLASH_FMPRE) &= 0xffffffff;
    //
    // The following sequence activates the one-time
    // programming of the FMPRE register.
    //
    HWREG(FLASH_FMA) = 0x900;
    HWREG(FLASH_FMC) = (FLASH_FMC_WRKEY | FLASH_FMC_COMT);
    //
    // Wait until the operation is complete.
    //
    while (HWREG(FLASH_FMC) & FLASH_FMC_COMT)
    {
    }
}
```

## 6.3.2 Flash Programming

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

During a Flash memory operation (write, page erase, or mass erase) access to the Flash memory is inhibited. As a result, instruction and literal fetches are held off until the Flash memory operation is complete. If instruction execution is required during a Flash memory operation, the code that is executing must be placed in SRAM and executed from there while the flash operation is in progress.

### 6.3.2.1 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.

### 6.3.2.2 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.

#### 6.3.2.3 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.

## 6.4 Register Map

Table 6-2 on page 220 lists the Flash memory and control registers. The offset listed is a hexadecimal increment to the register's address. The **FMA**, **FMD**, **FMC**, **FCRIS**, **FCIM**, and **FCMISC** register offsets are relative to the Flash memory control base address of 0x400F.D000. The Flash memory protection register offsets are relative to the System Control base address of 0x400F.E000.

**Table 6-2. Flash Register Map**

Offset	Name	Type	Reset	Description	See page
<b>Flash Memory Control Registers (Flash Control Offset)</b>					
0x000	FMA	R/W	0x0000.0000	Flash Memory Address	221
0x004	FMD	R/W	0x0000.0000	Flash Memory Data	222
0x008	FMC	R/W	0x0000.0000	Flash Memory Control	223
0x00C	FCRIS	RO	0x0000.0000	Flash Controller Raw Interrupt Status	225
0x010	FCIM	R/W	0x0000.0000	Flash Controller Interrupt Mask	226
0x014	FCMISC	R/W1C	0x0000.0000	Flash Controller Masked Interrupt Status and Clear	227
<b>Flash Memory Protection Registers (System Control Offset)</b>					
0x130	FMPRE	R/W	0xBFFF.FFFF	Flash Memory Protection Read Enable	230
0x134	FMPPE	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable	231
0x140	USECRL	R/W	0x31	USec Reload	229

## 6.5 Flash Register Descriptions (Flash Control Offset)

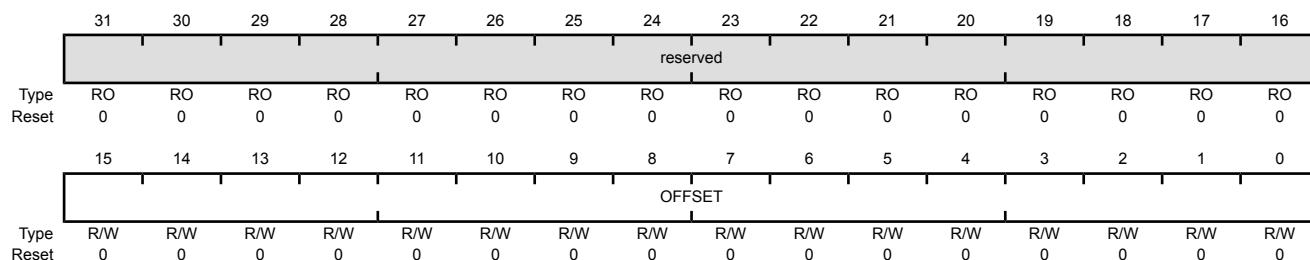
This section lists and describes the Flash Memory registers, in numerical order by address offset. Registers in this section are relative to the Flash control base address of 0x400F.D000.

## Register 1: 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



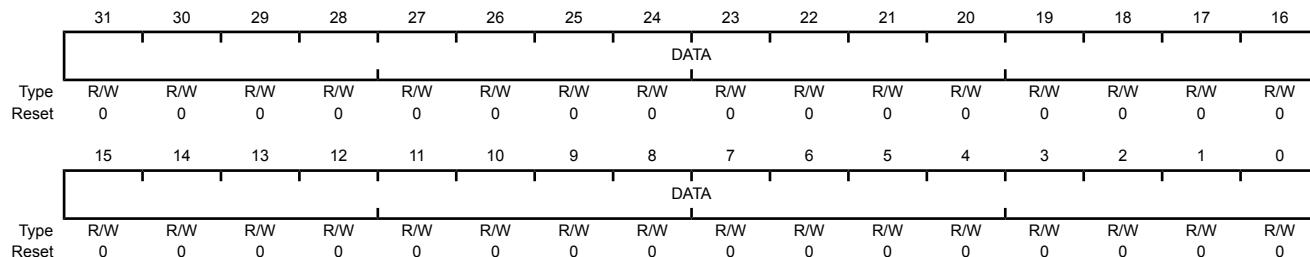
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	OFFSET	R/W	0x0	Address Offset Address offset in flash where operation is performed.

## Register 2: 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



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

## Register 3: 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 221). If the access is a write access, the data contained in the **Flash Memory Data (FMD)** register (see page 222) 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															
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 <b>WRKEY</b> 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 µs.
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.

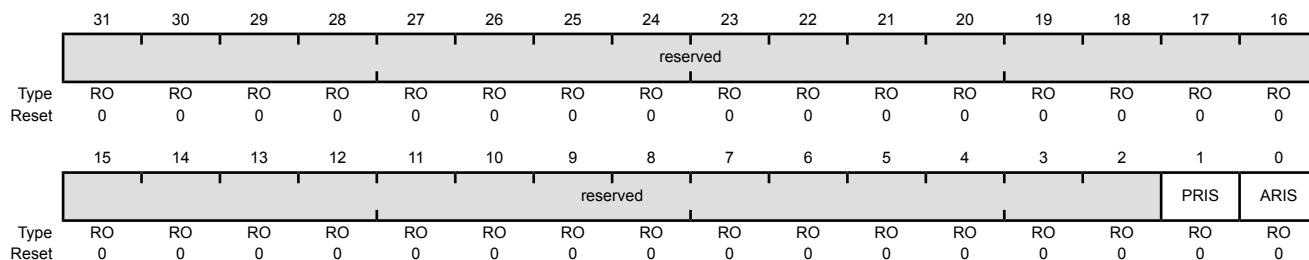
Bit/Field	Name	Type	Reset	Description
1	ERASE	R/W	0	<p>Erase a Page of Flash Memory</p> <p>If this bit is set, the page of flash main memory as specified by the contents of <b>FMA</b> is erased. A write of 0 has no effect on the state of this bit.</p> <p>If read, the state of the previous erase access 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 4: 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



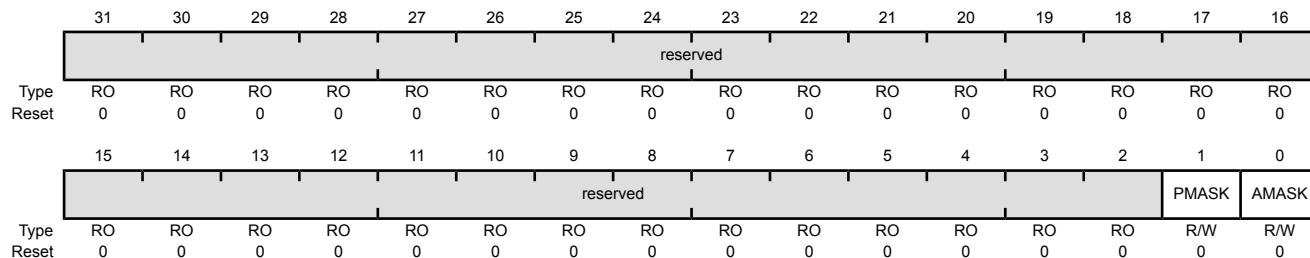
Bit/Field	Name	Type	Reset	Description						
31:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
1	PRIS	RO	0	<p>Programming Raw Interrupt Status</p> <p>This bit provides status on programming cycles which are write or erase actions generated through the <b>FMC</b> register bits (see page 223).</p> <table border="1"> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>1</td><td>The programming cycle has completed.</td></tr> <tr> <td>0</td><td>The programming cycle has not completed.</td></tr> </table> <p>This status is sent to the interrupt controller when the <b>PMASK</b> bit in the <b>FCIM</b> register is set.</p> <p>This bit is cleared by writing a 1 to the <b>PMISC</b> bit in the <b>FCMISC</b> register.</p>	Value	Description	1	The programming cycle has completed.	0	The programming cycle has not completed.
Value	Description									
1	The programming cycle has completed.									
0	The programming cycle has not completed.									
0	ARIS	RO	0	<p>Access Raw Interrupt Status</p> <table border="1"> <tr> <th>Value</th><th>Description</th></tr> <tr> <td>1</td><td>A program or erase action was attempted on a block of Flash memory that contradicts the protection policy for that block as set in the <b>FMPPEn</b> registers.</td></tr> <tr> <td>0</td><td>No access has tried to improperly program or erase the Flash memory.</td></tr> </table> <p>This status is sent to the interrupt controller when the <b>AMASK</b> bit in the <b>FCIM</b> register is set.</p> <p>This bit is cleared by writing a 1 to the <b>AMISC</b> bit in the <b>FCMISC</b> register.</p>	Value	Description	1	A program or erase action was attempted on a block of Flash memory that contradicts the protection policy for that block as set in the <b>FMPPEn</b> registers.	0	No access has tried to improperly program or erase the Flash memory.
Value	Description									
1	A program or erase action was attempted on a block of Flash memory that contradicts the protection policy for that block as set in the <b>FMPPEn</b> registers.									
0	No access has tried to improperly program or erase the Flash memory.									

## Register 5: 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



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	<p>Programming Interrupt Mask</p> <p>This bit controls the reporting of the programming raw interrupt status to the interrupt controller.</p> <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the PRIS bit is set.</td> </tr> <tr> <td>0</td> <td>The PRIS interrupt is suppressed and not sent to the interrupt controller.</td> </tr> </table>	Value	Description	1	An interrupt is sent to the interrupt controller when the PRIS bit is set.	0	The PRIS interrupt is suppressed and not sent to the interrupt controller.
Value	Description									
1	An interrupt is sent to the interrupt controller when the PRIS bit is set.									
0	The PRIS interrupt is suppressed and not sent to the interrupt controller.									
0	AMASK	R/W	0	<p>Access Interrupt Mask</p> <p>This bit controls the reporting of the access raw interrupt status to the interrupt controller.</p> <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the ARIS bit is set.</td> </tr> <tr> <td>0</td> <td>The ARIS interrupt is suppressed and not sent to the interrupt controller.</td> </tr> </table>	Value	Description	1	An interrupt is sent to the interrupt controller when the ARIS bit is set.	0	The ARIS interrupt is suppressed and not sent to the interrupt controller.
Value	Description									
1	An interrupt is sent to the interrupt controller when the ARIS bit is set.									
0	The ARIS interrupt is suppressed and not sent to the interrupt controller.									

## Register 6: Flash Controller Masked Interrupt Status and Clear (FCMISC), offset 0x014

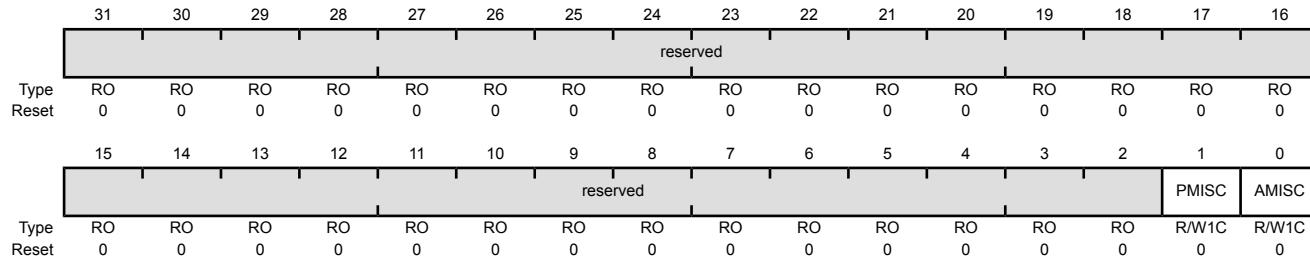
This register provides two functions. First, it reports the cause of an interrupt by indicating which interrupt source or sources are signalling the interrupt. Second, it serves as the method to clear the interrupt reporting.

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

Base 0x400FD000

Offset 0x014

Type R/W1C, reset 0x0000.0000



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  Value Description 1 When read, a 1 indicates that an unmasked interrupt was signaled because a programming cycle completed. Writing a 1 to this bit clears PMISC and also the PRIS bit in the <b>FCRIS</b> register (see page 225). 0 When read, a 0 indicates that a programming cycle complete interrupt has not occurred. A write of 0 has no effect on the state of this bit.
0	AMISC	R/W1C	0	Access Masked Interrupt Status and Clear  Value Description 1 When read, a 1 indicates that an unmasked interrupt was signaled because a program or erase action was attempted on a block of Flash memory that contradicts the protection policy for that block as set in the <b>FMPPEn</b> registers. Writing a 1 to this bit clears AMISC and also the ARIS bit in the <b>FCRIS</b> register (see page 225). 0 When read, a 0 indicates that no improper accesses have occurred. A write of 0 has no effect on the state of this bit.

## 6.6 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 7: USec Reload (USECRL), offset 0x140

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

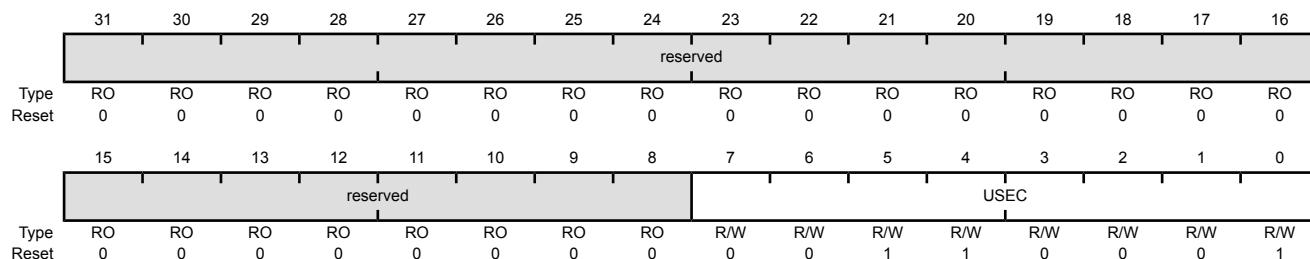
This register is provided as a means of creating a 1- $\mu$ s tick divider reload value for the flash controller. The internal flash has specific minimum and maximum requirements on the length of time the high voltage write pulse can be applied. It is required that this register contain the operating 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



## Register 8: Flash Memory Protection Read Enable (FMPRE), offset 0x130

**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 (see the **FMPPE** registers for the execute-only protection bits). This register is loaded during the power-on reset sequence. The factory settings 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. However, this register is R/W; 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 (FMPRE)

Base 0x400F.E000

Offset 0x130

Type R/W, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
READ_ENABLE																
Type	R/W															
Reset	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
READ_ENABLE																
Type	R/W															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Bit/Field	Name	Type	Reset	Description
31:30	DBG	R/W	0x2	User Controlled Debug Enable Each bit position maps 2 Kbytes of Flash to be read-enabled.
				Value Description 0x2 Debug access allowed
29:0	READ_ENABLE	R/W	0x3FFFFFFF	Flash Read Enable Each bit position maps 2 Kbytes of Flash to be read-enabled.
				Value Description 0x3FFFFFFF Enables 64 KB of flash.

29:0            READ\_ENABLE        R/W    0x3FFFFFFF    Flash Read Enable  
Each bit position maps 2 Kbytes of Flash to be read-enabled.

Value            Description  
0x3FFFFFFF    Enables 64 KB of flash.

## Register 9: Flash Memory Protection Program Enable (FMPPE), offset 0x134

**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 (see the **FMPRE** registers for the read-only protection bits). This register is loaded during the power-on reset sequence. The factory settings 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. However, this register is R/W; 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 (FMPPE)

Base 0x400F.E000

Offset 0x134

Type R/W, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PROG_ENABLE																
Type	R/W															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
PROG_ENABLE																
Type	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	PROG_ENABLE	R/W	0xFFFFFFFF	Flash Programming Enable Each bit position maps 2 Kbytes of Flash to be write-enabled.
				Value Description
				0xFFFFFFFF Enables 64 KB of flash.

## 7 General-Purpose Input/Outputs (GPIOs)

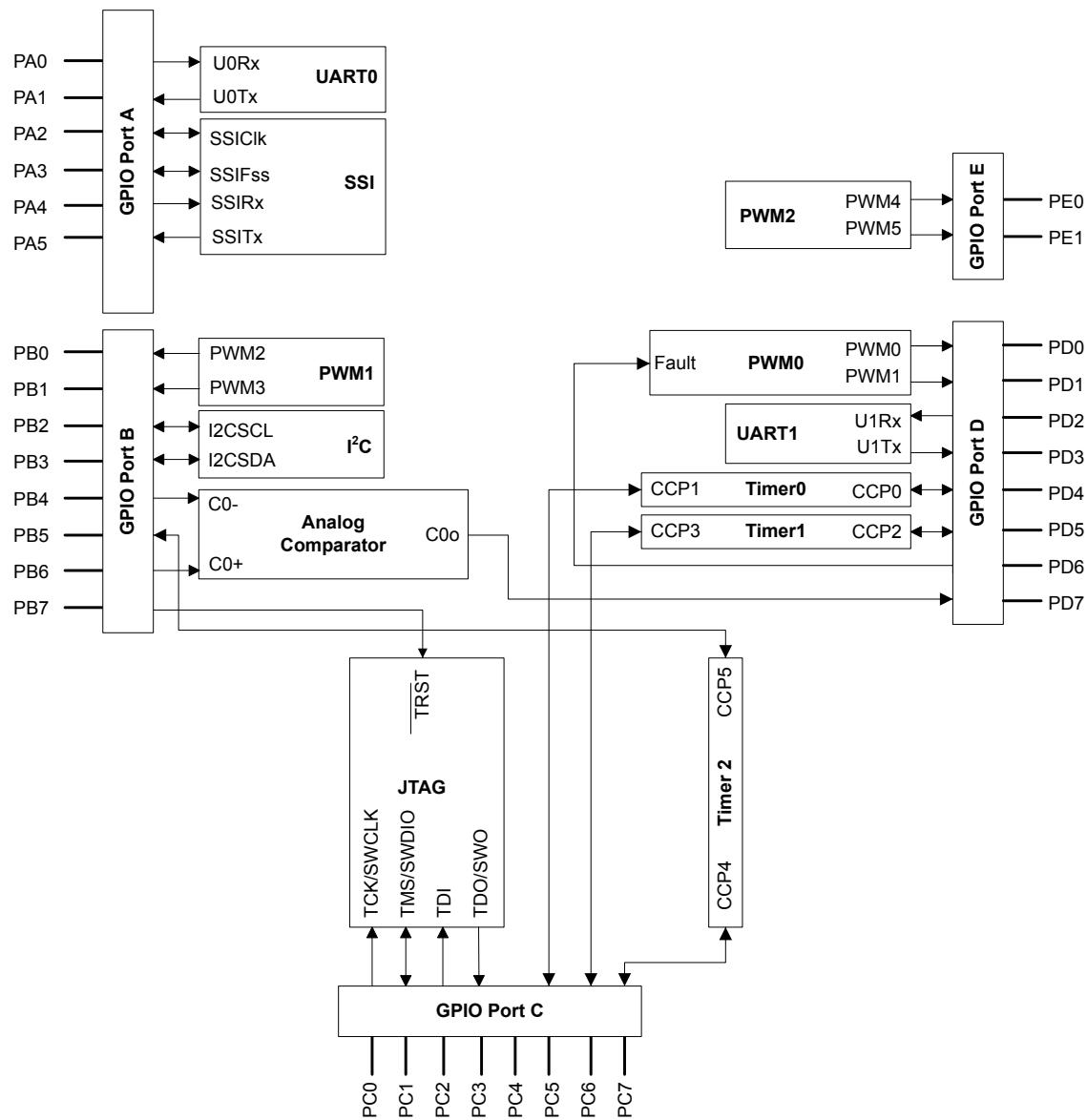
The GPIO module is composed of five physical GPIO blocks, each corresponding to an individual GPIO port (Port A, Port B, Port C, Port D, Port E). The GPIO module supports 1-32 programmable input/output pins, depending on the peripherals being used.

The GPIO module has the following features:

- 1-32 GPIOs, depending on configuration
- 5-V-tolerant in input configuration
- Fast toggle capable of a change every two clock cycles
- Programmable control for GPIO interrupts
  - Interrupt generation masking
  - Edge-triggered on rising, falling, or both
  - Level-sensitive on High or Low values
- 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
  - Slew rate control for the 8-mA drive
  - Open drain enables
  - Digital input enables

## 7.1 Block Diagram

Figure 7-1. GPIO Module Block Diagram



## 7.2 Signal Description

GPIO signals have alternate hardware functions. Table 7-4 on page 236 and Table 7-5 on page 236 list the GPIO pins and their analog and digital alternate functions. The **A<sub>IN</sub>x** analog signals are not 5-V tolerant and go through an isolation circuit before reaching their circuitry. These signals are configured by clearing the corresponding **DEN** bit in the **GPIO Digital Enable (GPIODEN)** register. Other analog signals are 5-V tolerant and are connected directly to their circuitry (**C<sub>0</sub>-**, **C<sub>0</sub>+**). These signals are configured by clearing the **DEN** bit in the **GPIO Digital Enable (GPIODEN)** register. The digital alternate hardware functions are enabled by setting the appropriate bit in the **GPIO Alternate Function Select (GPIOAFSEL)** and **GPIODEN** registers and configuring the **PMC<sub>x</sub>** bit field in the

**GPIO Port Control (GPIOPCTL)** register to the numeric encoding shown in the table below. Note that each pin must be programmed individually; no type of grouping is implied by the columns in the table.

**Important:** All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL=0**, **GPIODEN=0**, **GPIOPDR=0**, **GPIOPUR=0**, and **GPIOPCTL=0**, with the exception of the four JTAG/SWD pins (shown in the table below). A Power-On-Reset (**POR**) or asserting **RST** puts the pins back to their default state.

**Table 7-1. GPIO Pins With Non-Zero Reset Values**

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	1	1	0	0	0x1
PA[5:2]	SSI0	1	1	0	0	0x1
PB[3:2]	I <sup>2</sup> C0	1	1	0	0	0x1
PC[3:0]	JTAG/SWD	1	1	0	1	0x3

**Table 7-2. GPIO Pins and Alternate Functions (48QFP)**

IO	Pin Number	Multiplexed Function	Multiplexed Function
PA0	17	U0Rx	
PA1	18	U0Tx	
PA2	19	SSIClk	
PA3	20	SSIIFss	
PA4	21	SSIRx	
PA5	22	SSITx	
PB0	29	PWM2	
PB1	30	PWM3	
PB2	33	I <sup>2</sup> CSCL	
PB3	34	I <sup>2</sup> CSDA	
PB4	44	C0-	
PB5	43	CCP5	
PB6	42	C0+	
PB7	41	TRST	
PC0	40	TCK	SWCLK
PC1	39	TMS	SWDIO
PC2	38	TDI	
PC3	37	TDO	SWO
PC4	14		
PC5	13	CCP1	
PC6	12	CCP3	
PC7	11	CCP4	
PD0	25	PWM0	
PD1	26	PWM1	
PD2	27	U1Rx	
PD3	28	U1Tx	

**Table 7-2. GPIO Pins and Alternate Functions (48QFP) (continued)**

IO	Pin Number	Multiplexed Function	Multiplexed Function
PD4	45	CCP0	
PD5	46	CCP2	
PD6	47	Fault	
PD7	48	C0o	
PE0	35	PWM4	
PE1	36	PWM5	

**Table 7-3. GPIO Pins and Alternate Functions (48QFN)**

IO	Pin Number	Multiplexed Function	Multiplexed Function
PA0	17	U0Rx	
PA1	18	U0Tx	
PA2	19	SSIClk	
PA3	20	SSIIfss	
PA4	21	SSIRx	
PA5	22	SSITx	
PB0	29	PWM2	
PB1	30	PWM3	
PB2	33	I2CSCL	
PB3	34	I2CSDA	
PB4	44	C0-	
PB5	43	CCP5	
PB6	42	C0+	
PB7	41	TRST	
PC0	40	TCK	SWCLK
PC1	39	TMS	SWDIO
PC2	38	TDI	
PC3	37	TDO	SWO
PC4	14		
PC5	13	CCP1	
PC6	12	CCP3	
PC7	11	CCP4	
PD0	25	PWM0	
PD1	26	PWM1	
PD2	27	U1Rx	
PD3	28	U1Tx	
PD4	45	CCP0	
PD5	46	CCP2	
PD6	47	Fault	
PD7	48	C0o	
PE0	35	PWM4	
PE1	36	PWM5	

**Table 7-4. GPIO Signals (48QFP)**

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
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.
PB0	29	I/O	TTL	GPIO port B bit 0.
PB1	30	I/O	TTL	GPIO port B bit 1.
PB2	33	I/O	TTL	GPIO port B bit 2.
PB3	34	I/O	TTL	GPIO port B bit 3.
PB4	44	I/O	TTL	GPIO port B bit 4.
PB5	43	I/O	TTL	GPIO port B bit 5.
PB6	42	I/O	TTL	GPIO port B bit 6.
PB7	41	I/O	TTL	GPIO port B bit 7.
PC0	40	I/O	TTL	GPIO port C bit 0.
PC1	39	I/O	TTL	GPIO port C bit 1.
PC2	38	I/O	TTL	GPIO port C bit 2.
PC3	37	I/O	TTL	GPIO port C bit 3.
PC4	14	I/O	TTL	GPIO port C bit 4.
PC5	13	I/O	TTL	GPIO port C bit 5.
PC6	12	I/O	TTL	GPIO port C bit 6.
PC7	11	I/O	TTL	GPIO port C bit 7.
PD0	25	I/O	TTL	GPIO port D bit 0.
PD1	26	I/O	TTL	GPIO port D bit 1.
PD2	27	I/O	TTL	GPIO port D bit 2.
PD3	28	I/O	TTL	GPIO port D bit 3.
PD4	45	I/O	TTL	GPIO port D bit 4.
PD5	46	I/O	TTL	GPIO port D bit 5.
PD6	47	I/O	TTL	GPIO port D bit 6.
PD7	48	I/O	TTL	GPIO port D bit 7.
PE0	35	I/O	TTL	GPIO port E bit 0.
PE1	36	I/O	TTL	GPIO port E bit 1.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

**Table 7-5. GPIO Signals (48QFN)**

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
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.

**Table 7-5. GPIO Signals (48QFN) (continued)**

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
PA5	22	I/O	TTL	GPIO port A bit 5.
PB0	29	I/O	TTL	GPIO port B bit 0.
PB1	30	I/O	TTL	GPIO port B bit 1.
PB2	33	I/O	TTL	GPIO port B bit 2.
PB3	34	I/O	TTL	GPIO port B bit 3.
PB4	44	I/O	TTL	GPIO port B bit 4.
PB5	43	I/O	TTL	GPIO port B bit 5.
PB6	42	I/O	TTL	GPIO port B bit 6.
PB7	41	I/O	TTL	GPIO port B bit 7.
PC0	40	I/O	TTL	GPIO port C bit 0.
PC1	39	I/O	TTL	GPIO port C bit 1.
PC2	38	I/O	TTL	GPIO port C bit 2.
PC3	37	I/O	TTL	GPIO port C bit 3.
PC4	14	I/O	TTL	GPIO port C bit 4.
PC5	13	I/O	TTL	GPIO port C bit 5.
PC6	12	I/O	TTL	GPIO port C bit 6.
PC7	11	I/O	TTL	GPIO port C bit 7.
PD0	25	I/O	TTL	GPIO port D bit 0.
PD1	26	I/O	TTL	GPIO port D bit 1.
PD2	27	I/O	TTL	GPIO port D bit 2.
PD3	28	I/O	TTL	GPIO port D bit 3.
PD4	45	I/O	TTL	GPIO port D bit 4.
PD5	46	I/O	TTL	GPIO port D bit 5.
PD6	47	I/O	TTL	GPIO port D bit 6.
PD7	48	I/O	TTL	GPIO port D bit 7.
PE0	35	I/O	TTL	GPIO port E bit 0.
PE1	36	I/O	TTL	GPIO port E bit 1.

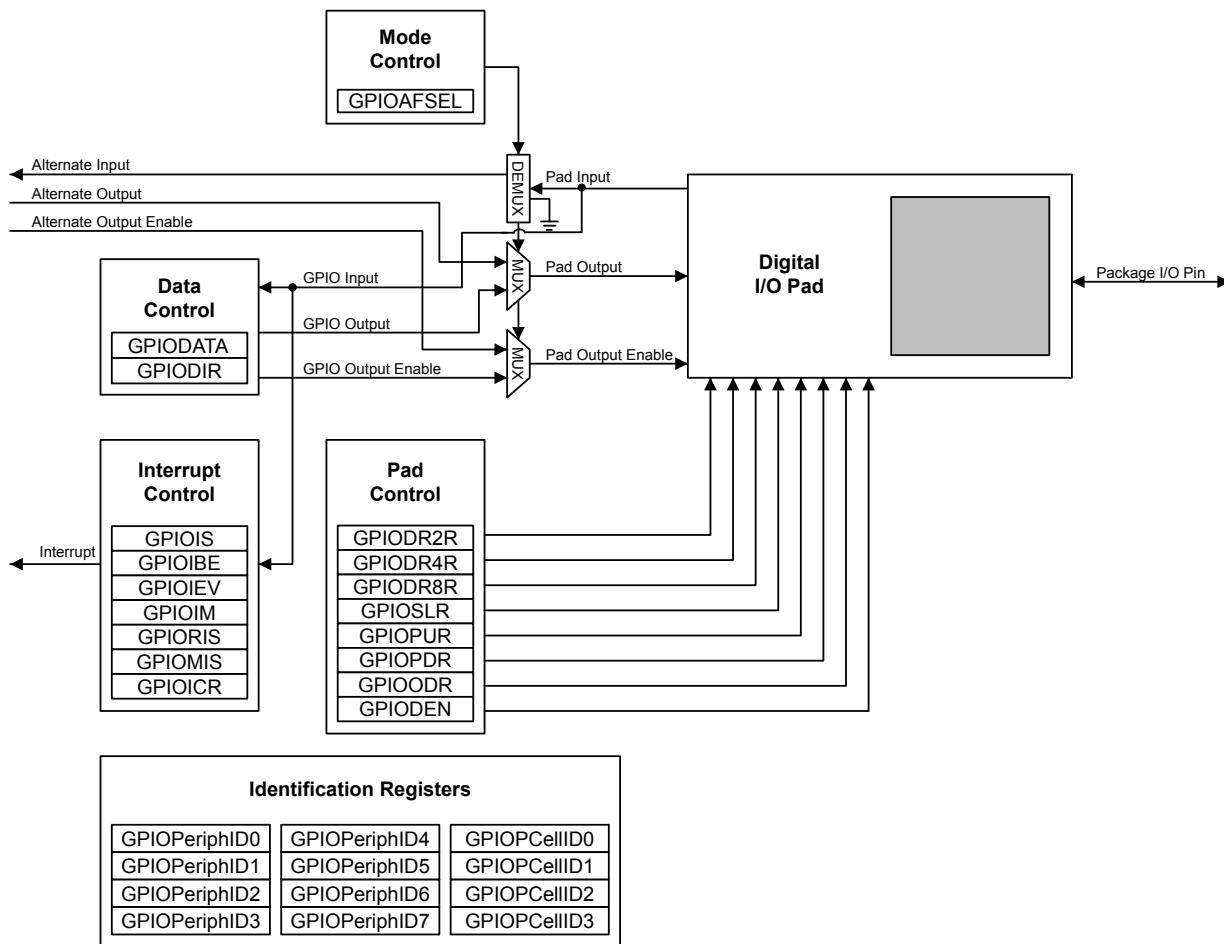
a. The TTL designation indicates the pin has TTL-compatible voltage levels.

### 7.3 Functional Description

**Important:** All GPIO pins are inputs by default (**GPIODIR=0** and **GPIOAFSEL=0**), with the exception of the five JTAG pins (**PB7** and **PC[3:0]**). The JTAG pins default to their JTAG functionality (**GPIOAFSEL=1**). A Power-On-Reset (**POR**) or asserting an external reset (**RST**) puts both groups of pins back to their default state.

While debugging systems where **PB7** is being used as a GPIO, care must be taken to ensure that a Low value is not applied to the pin when the part is reset. Because **PB7** reverts to the **TRST** function after reset, a Low value on the pin causes the JTAG controller to be reset, resulting in a loss of JTAG communication.

Each GPIO port is a separate hardware instantiation of the same physical block (see Figure 7-2 on page 238). The LM3S811 microcontroller contains five ports and thus five of these physical GPIO blocks.

**Figure 7-2. GPIO Port Block Diagram**

### 7.3.1 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.

#### 7.3.1.1 Data Direction Operation

The **GPIO Direction (GPIODIR)** register (see page 245) 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.

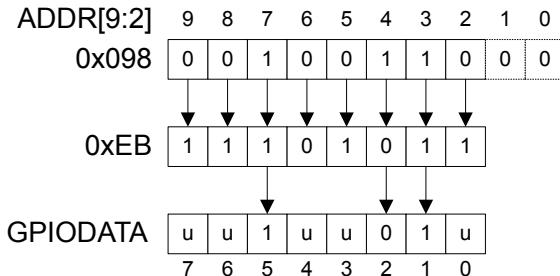
#### 7.3.1.2 Data Register Operation

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 244) 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 **GPIO DATA** register is altered. If it is cleared to 0, it is left unchanged.

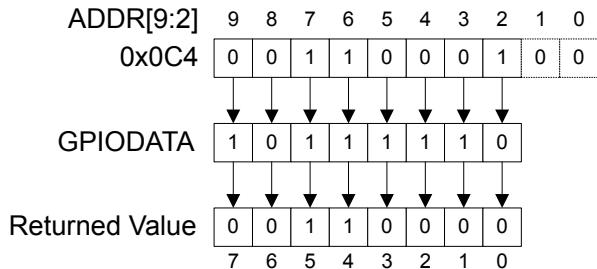
For example, writing a value of 0xEB to the address **GPIO DATA** + 0x098 would yield as shown in Figure 7-3 on page 239, where *u* is data unchanged by the write.

**Figure 7-3. GPIO DATA 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 **GPIO DATA** + 0x0C4 yields as shown in Figure 7-4 on page 239.

**Figure 7-4. GPIO DATA Read Example**



### 7.3.2 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 246)
- **GPIO Interrupt Both Edges (GPIOIBE)** register (see page 247)
- **GPIO Interrupt Event (GPIOIEV)** register (see page 248)

Interrupts are enabled/disabled via the **GPIO Interrupt Mask (GPIOIM)** register (see page 249).

When an interrupt condition occurs, the state of the interrupt signal can be viewed in two locations: the **GPIO Raw Interrupt Status (GPIORIS)** and **GPIO Masked Interrupt Status (GPIOVIS)** registers (see page 250 and page 251). As the name implies, the **GPIOVIS** register only shows interrupt

conditions that are allowed to be passed to the controller. The **GPIOISR** 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 **Interrupt 0-31 Set Enable (EN0)** 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 PB4, and wait for the ADC interrupt or the ADC interrupt must be disabled in the **EN0** register and the PortB interrupt handler must poll the ADC registers until the conversion is completed. See page 100 for more information.

Interrupts are cleared by writing a 1 to the appropriate bit of the **GPIO Interrupt Clear (GPIOICR)** register (see page 252).

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.

### 7.3.3 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 253), 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.

### 7.3.4 Pad Control

The pad control registers allow for GPIO pad configuration by software based on the application requirements. The pad control registers include the **GPIOODR2R**, **GPIOODR4R**, **GPIOODR8R**, **GPIOODR**, **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 enable.

### 7.3.5 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 **GPIOCellIID0-GPIOCellIID3** registers.

## 7.4 Initialization and Configuration

To use the GPIO, the peripheral clock must be enabled by setting the appropriate GPIO Port bit field (**GPIOOn**) in the **RCGC2** register.

On reset, all GPIO pins (except for the five JTAG pins) default to general-purpose input mode (**GPIODIR=0** and **GPIOAFSEL=0**). Table 7-6 on page 241 shows all possible configurations of the GPIO pads and the control register settings required to achieve them. Table 7-7 on page 241 shows how a rising edge interrupt would be configured for pin 2 of a GPIO port.

**Table 7-6. 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 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 (PWM)	1	X	0	1	?	?	?	?	?	?
Digital Output (Timer PWM)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (SSI)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (UART)	1	X	0	1	?	?	?	?	?	?
Analog Input (Comparator)	0	0	0	0	0	0	X	X	X	X
Digital Output (Comparator)	1	X	0	1	?	?	?	?	?	?

a. X=Ignored (don't care bit)

?=Can be either 0 or 1, depending on the configuration

**Table 7-7. 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	
GPIOEV	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)

## 7.5 Register Map

Table 7-8 on page 242 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: 0x4000.4000
- GPIO Port B: 0x4000.5000
- GPIO Port C: 0x4000.6000
- GPIO Port D: 0x4000.7000
- GPIO Port E: 0x4002.4000

Note that the GPIO module clock must be enabled before the registers can be programmed (see page 205). There must be a delay of 3 system clocks after the GPIO module clock is enabled before any GPIO module registers are accessed.

**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** register is 0x0000.0000 for all GPIO pins, with the exception of the five JTAG pins (**PB7** and **PC[3:0]**). These five pins default to JTAG functionality. Because of this, the default reset value of **GPIOAFSEL** for GPIO Port B is 0x0000.0080 while the default reset value for Port C is 0x0000.000F.

**Table 7-8. GPIO Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	GPIODATA	R/W	0x0000.0000	GPIO Data	244
0x400	GPIODIR	R/W	0x0000.0000	GPIO Direction	245
0x404	GPIOIS	R/W	0x0000.0000	GPIO Interrupt Sense	246
0x408	GPIOIBE	R/W	0x0000.0000	GPIO Interrupt Both Edges	247
0x40C	GPIOIEV	R/W	0x0000.0000	GPIO Interrupt Event	248
0x410	GPIOIM	R/W	0x0000.0000	GPIO Interrupt Mask	249
0x414	GPIOIRS	RO	0x0000.0000	GPIO Raw Interrupt Status	250
0x418	GIOMIS	RO	0x0000.0000	GPIO Masked Interrupt Status	251
0x41C	GPIOICR	W1C	0x0000.0000	GPIO Interrupt Clear	252
0x420	GPIOAFSEL	R/W	-	GPIO Alternate Function Select	253
0x500	GPIODR2R	R/W	0x0000.00FF	GPIO 2-mA Drive Select	255
0x504	GPIODR4R	R/W	0x0000.0000	GPIO 4-mA Drive Select	256
0x508	GPIODR8R	R/W	0x0000.0000	GPIO 8-mA Drive Select	257
0x50C	GPIOODR	R/W	0x0000.0000	GPIO Open Drain Select	258
0x510	GPIOPUR	R/W	0x0000.00FF	GPIO Pull-Up Select	259
0x514	GPIOPDR	R/W	0x0000.0000	GPIO Pull-Down Select	260
0x518	GPIOSLR	R/W	0x0000.0000	GPIO Slew Rate Control Select	261
0x51C	GPIODEN	R/W	0x0000.00FF	GPIO Digital Enable	262
0xFD0	GPIOPeriphID4	RO	0x0000.0000	GPIO Peripheral Identification 4	263

**Table 7-8. GPIO Register Map (continued)**

Offset	Name	Type	Reset	Description	See page
0xFD4	GPIOPeriphID5	RO	0x0000.0000	GPIO Peripheral Identification 5	264
0xFD8	GPIOPeriphID6	RO	0x0000.0000	GPIO Peripheral Identification 6	265
0xFDC	GPIOPeriphID7	RO	0x0000.0000	GPIO Peripheral Identification 7	266
0xFE0	GPIOPeriphID0	RO	0x0000.0061	GPIO Peripheral Identification 0	267
0xFE4	GPIOPeriphID1	RO	0x0000.0000	GPIO Peripheral Identification 1	268
0xFE8	GPIOPeriphID2	RO	0x0000.0018	GPIO Peripheral Identification 2	269
0xFEC	GPIOPeriphID3	RO	0x0000.0001	GPIO Peripheral Identification 3	270
0xFF0	GPIOPCellID0	RO	0x0000.000D	GPIO PrimeCell Identification 0	271
0xFF4	GPIOPCellID1	RO	0x0000.00F0	GPIO PrimeCell Identification 1	272
0xFF8	GPIOPCellID2	RO	0x0000.0005	GPIO PrimeCell Identification 2	273
0xFFC	GPIOPCellID3	RO	0x0000.00B1	GPIO PrimeCell Identification 3	274

## 7.6 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 245).

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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

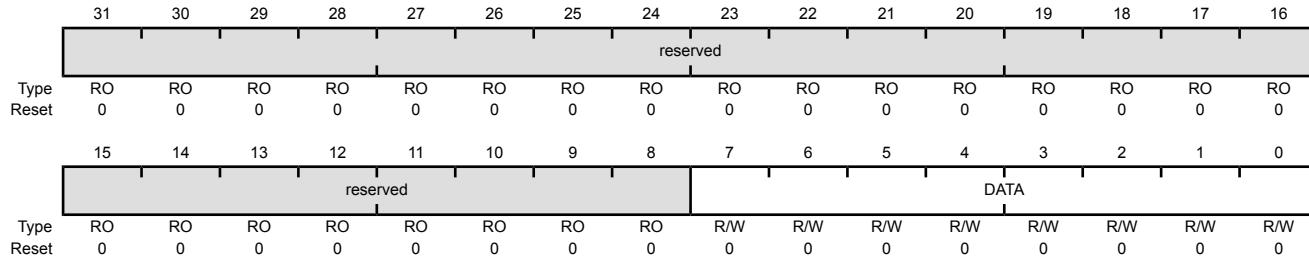
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

Offset 0x000

Type R/W, reset 0x0000.0000



## 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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

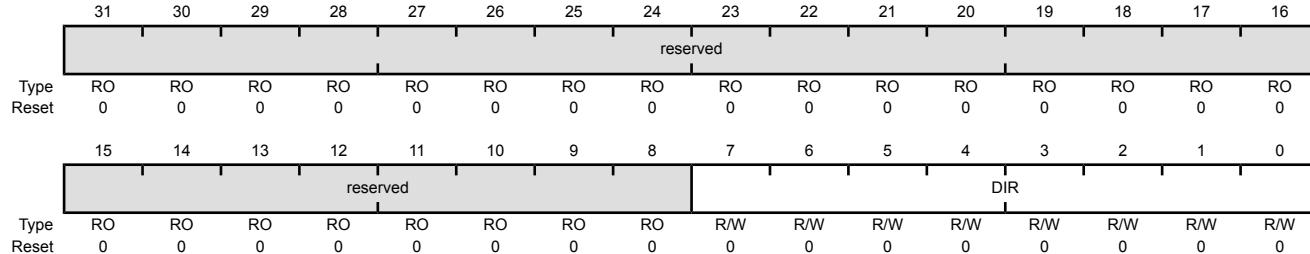
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

Offset 0x400

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      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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

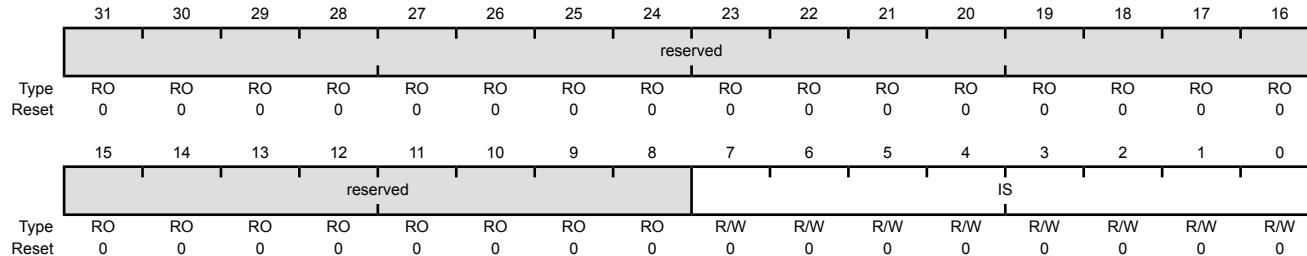
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

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 246) 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 248). 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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

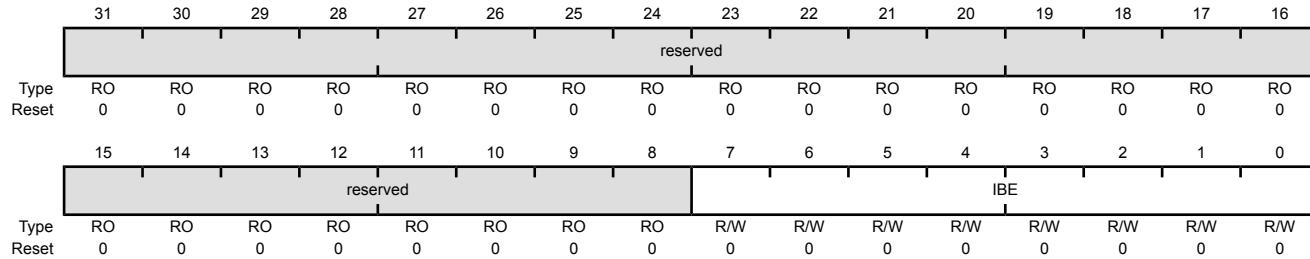
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

Offset 0x408

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	IBE	R/W	0x00	GPIO Interrupt Both Edges The IBE values are defined as follows:

Value	Description
0	Interrupt generation is controlled by the <b>GPIO Interrupt Event (GPIOIEV)</b> register (see page 248).
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 246). 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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

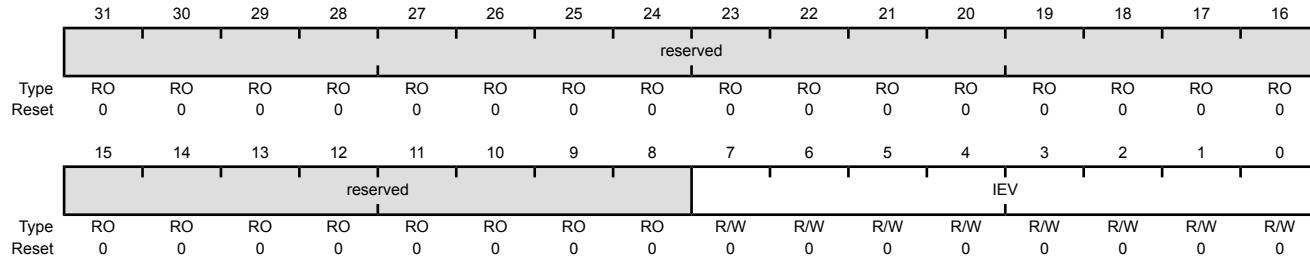
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

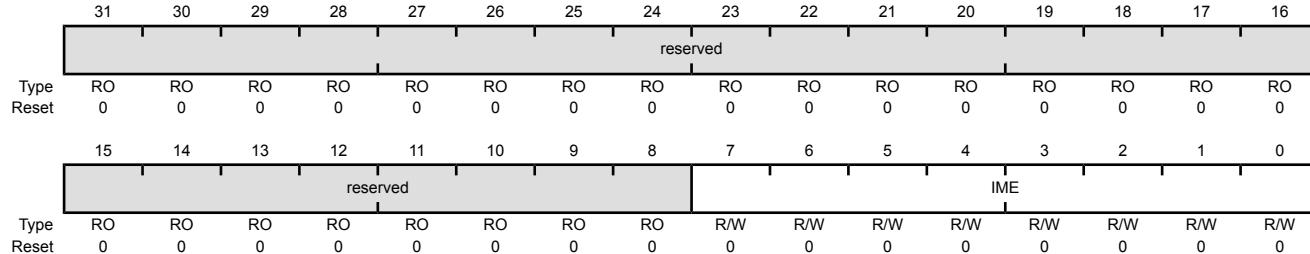
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

Offset 0x410

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      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 249). 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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

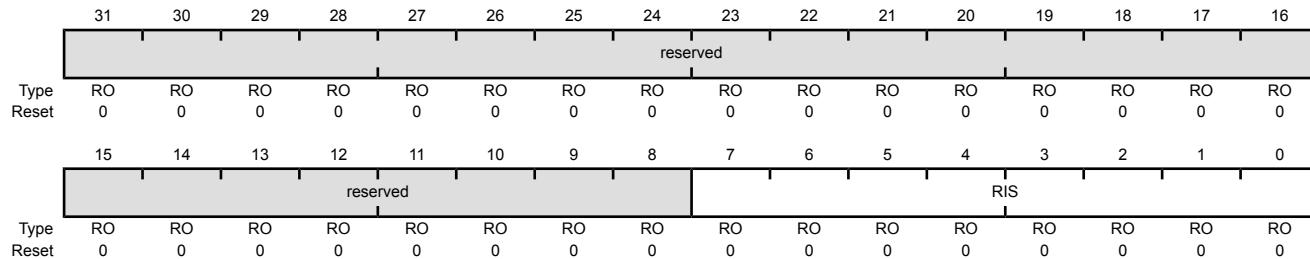
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

Offset 0x414

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	RIS	RO	0x00	<p>GPIO Interrupt Raw Status</p> <p>Reflects the status of interrupt trigger condition detection on pins (raw, prior to masking).</p> <p>The RIS values are defined as follows:</p>

#### Value Description

- 0 Corresponding pin interrupt requirements not met.
- 1 Corresponding pin interrupt has met requirements.

## Register 8: GPIO Masked Interrupt Status (GPIO MIS), offset 0x418

The **GPIO MIS** register is the masked interrupt status register. Bits read High in **GPIO MIS** 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 **Interrupt 0-31 Set Enable (EN0)** 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 PB4, and wait for the ADC interrupt or the ADC interrupt must be disabled in the **EN0** register and the PortB interrupt handler must poll the ADC registers until the conversion is completed. See page 100 for more information.

**GPIO MIS** is the state of the interrupt after masking.

### GPIO Masked Interrupt Status (GPIO MIS)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

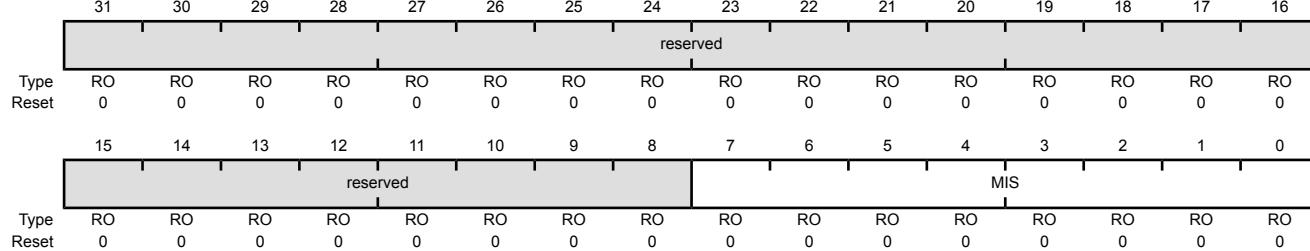
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

Offset 0x418

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	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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

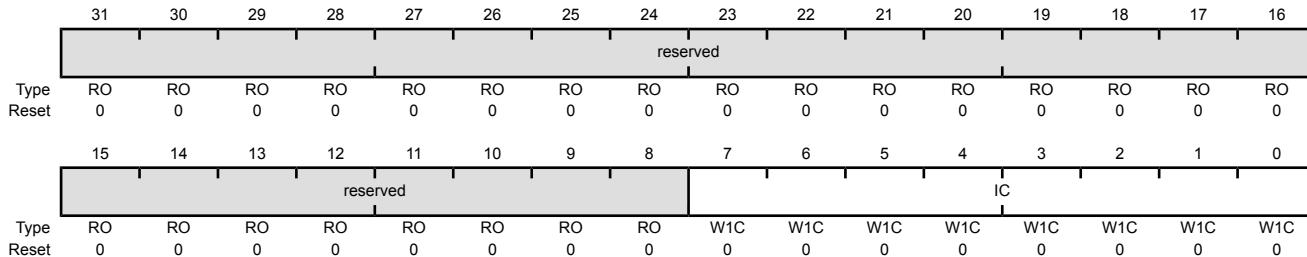
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

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: <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Corresponding interrupt is unaffected.</td> </tr> <tr> <td>1</td> <td>Corresponding interrupt is cleared.</td> </tr> </tbody> </table>	Value	Description	0	Corresponding interrupt is unaffected.	1	Corresponding interrupt is cleared.
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.

**Important:** All GPIO pins are inputs by default (**GPIODIR**=0 and **GPIOAFSEL**=0), with the exception of the five JTAG pins (**PB7** and **PC[3:0]**). The JTAG pins default to their JTAG functionality (**GPIOAFSEL**=1). A Power-On-Reset (**POR**) or asserting an external reset (**RST**) puts both groups of pins back to their default state.

While debugging systems where **PB7** is being used as a GPIO, care must be taken to ensure that a Low value is not applied to the pin when the part is reset. Because **PB7** reverts to the **TRST** function after reset, a Low value on the pin causes the JTAG controller to be reset, resulting in a loss of JTAG communication.

**Caution – If the JTAG pins are used as GPIOs in a design, **PB7** and **PC2** cannot have external pull-down resistors connected to both of them at the same time. If both pins are pulled Low during reset, the controller has unpredictable behavior. If this happens, remove one or both of the pull-down resistors, and apply **RST** or power-cycle the part.**

**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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

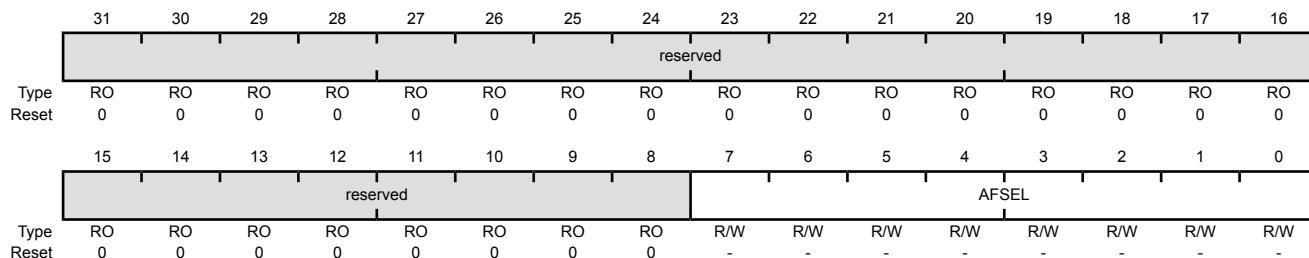
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

Offset 0x420

Type R/W, reset -



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> register is 0x0000.0000 for all GPIO pins, with the exception of the five JTAG pins ( <b>PB7</b> and <b>PC[3:0]</b> ). These five pins default to JTAG functionality. Because of this, the default reset value of <b>GPIOAFSEL</b> for GPIO Port B is 0x0000.0080 while the default reset value 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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

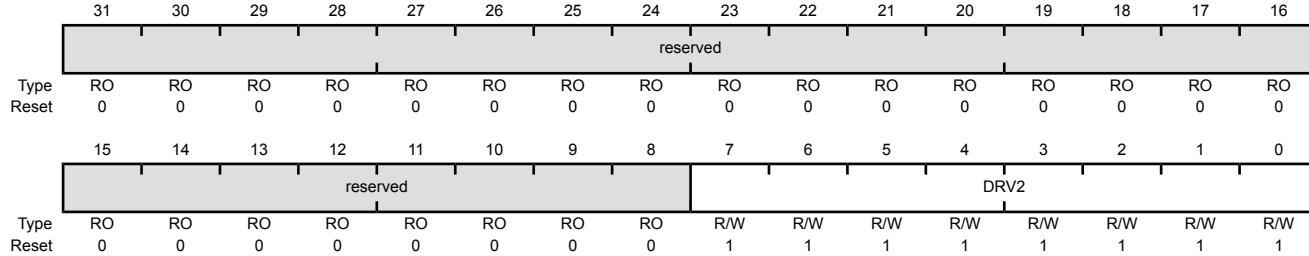
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

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.

## 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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

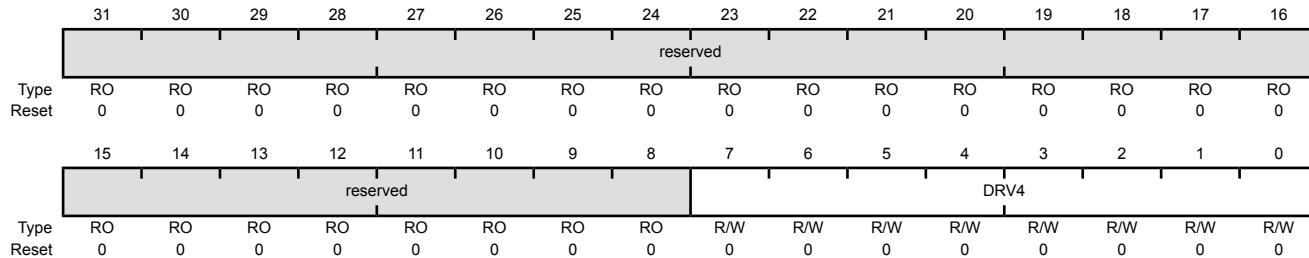
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

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.

## 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.

### GPIO 8-mA Drive Select (GPIODR8R)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

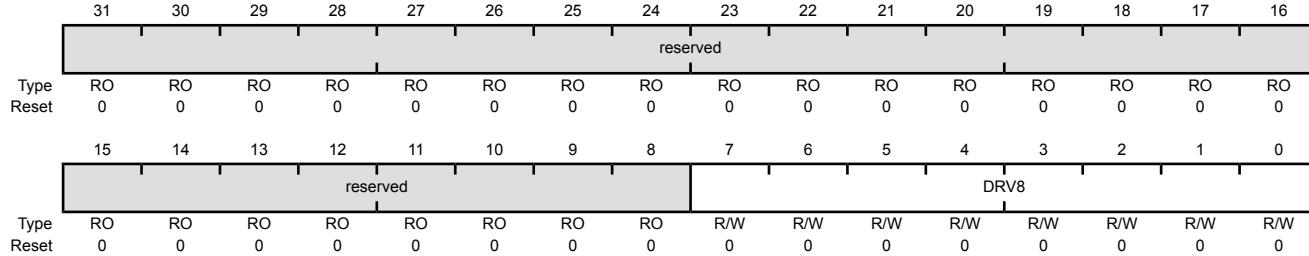
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

Offset 0x508

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	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.

## 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 Enable (GPIODEN)** register (see page 262). 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 cleared. If open drain is selected while the GPIO is configured as an input, the GPIO will remain an input and the open-drain selection has no effect until the GPIO is changed to an output.

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 bits for the I<sup>2</sup>C clock and data pins should be set to 1 (see examples in “Initialization and Configuration” on page 240).

### GPIO Open Drain Select (GPIOODR)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

Offset 0x50C

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								ODE							
Type	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

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 260).

### GPIO Pull-Up Select (GPIOPUR)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

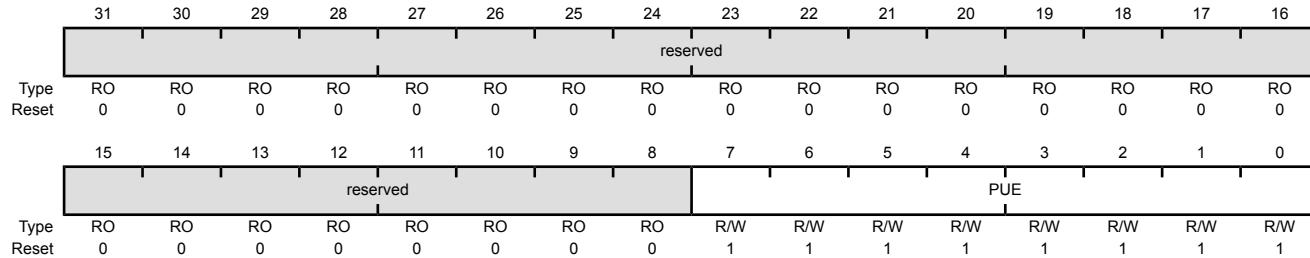
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

Offset 0x510

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      PUE      R/W      0xFF      Pad Weak Pull-Up Enable

#### Value      Description

0      The corresponding pin's weak pull-up resistor is disabled.

1      The corresponding pin's weak pull-up resistor is enabled.

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.

## Register 16: GPIO Pull-Down Select (GPIOOPDR), offset 0x514

The **GPIOOPDR** 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 **GPIOOPDR** automatically clears the corresponding bit in the **GPIO Pull-Up Select (GPIOPUR)** register (see page 259).

### GPIO Pull-Down Select (GPIOOPDR)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

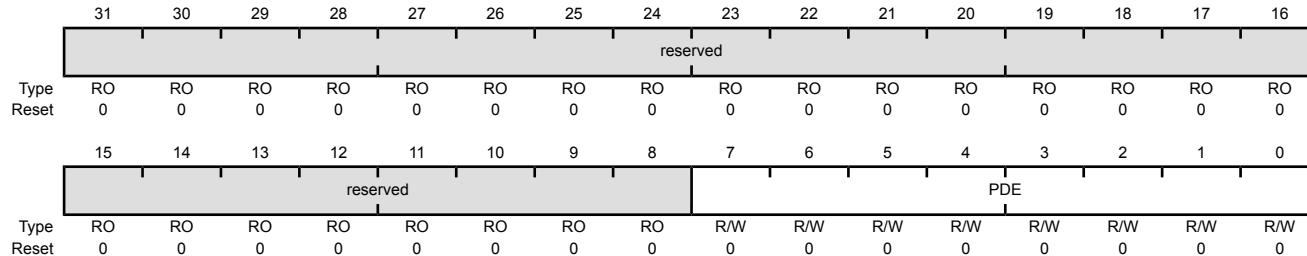
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

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

#### Value      Description

- 0      The corresponding pin's weak pull-down resistor is disabled.
- 1      The corresponding pin's weak pull-down resistor is enabled.

A write of 1 to **GPIOPUR[n]** clears the corresponding **GPIOOPDR[n]** 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 (GPIODR8R)** register (see page 257).

### GPIO Slew Rate Control Select (GPIOSLR)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

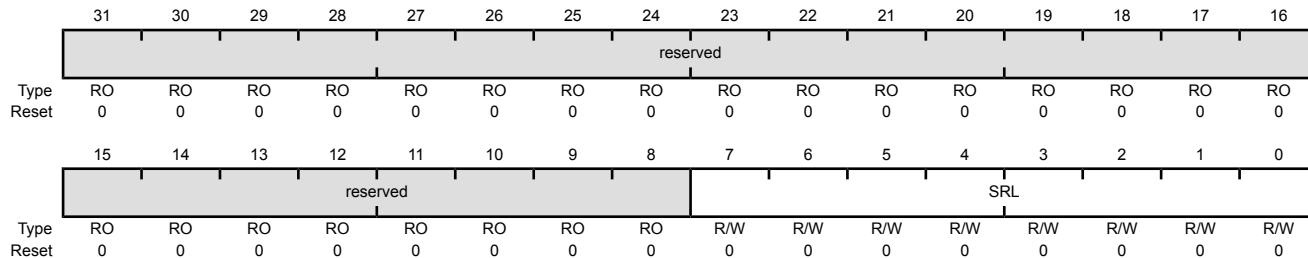
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

Offset 0x518

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      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, all GPIO signals are configured as digital inputs at reset. If a pin is being used as a GPIO or its Alternate Hardware Function, it should be configured as a digital input. The only time that a pin should not be configured as a digital input is when the GPIO pin is configured to be one of the analog input signals for the analog comparators.

### GPIO Digital Enable (GPIODEN)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

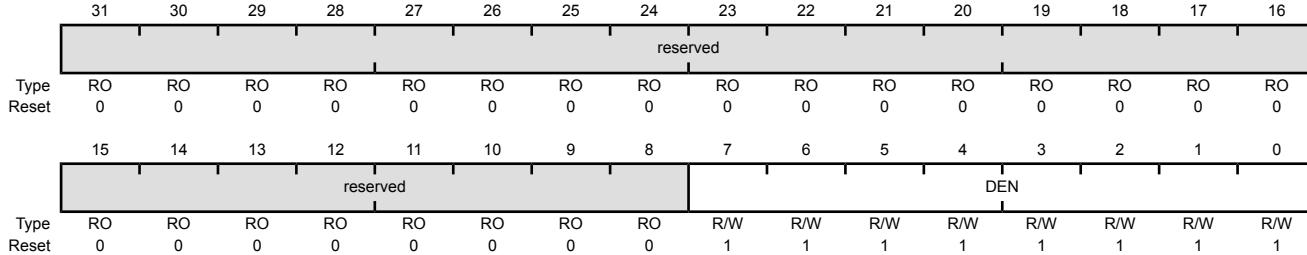
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

Offset 0x51C

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	DEN	R/W	0xFF	<p>Digital Enable</p> <p>The DEN values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Digital functions disabled.</td> </tr> <tr> <td>1</td> <td>Digital functions enabled.</td> </tr> </tbody> </table>	Value	Description	0	Digital functions disabled.	1	Digital functions enabled.
Value	Description									
0	Digital functions disabled.									
1	Digital functions enabled.									

## Register 19: 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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

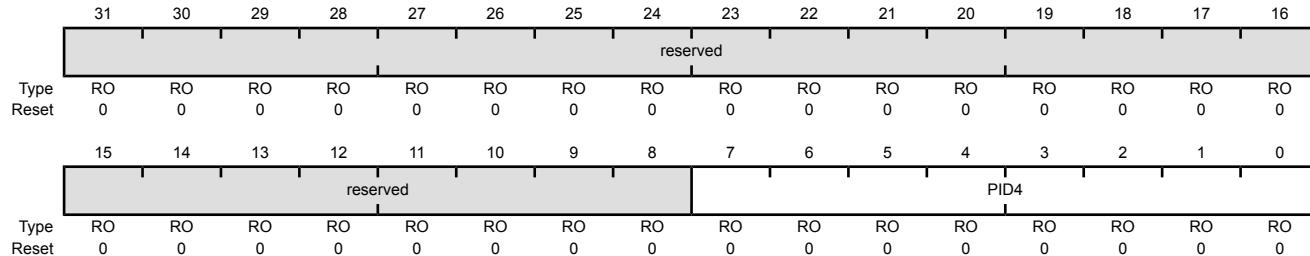
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

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 20: 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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

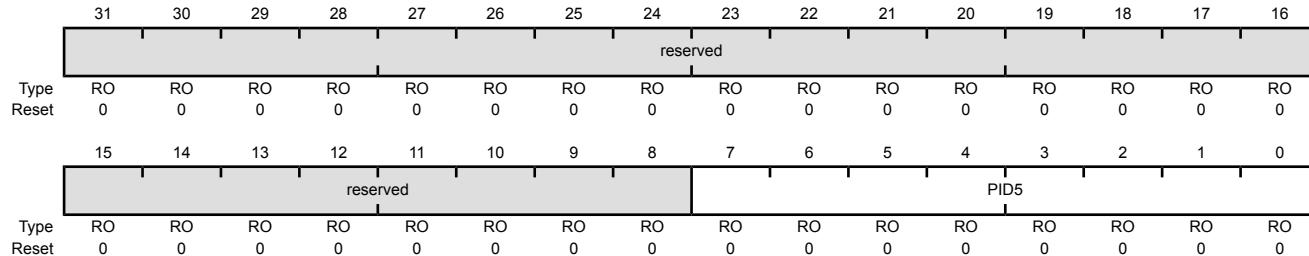
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

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      GPIO Peripheral ID Register[15:8]

## Register 21: 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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

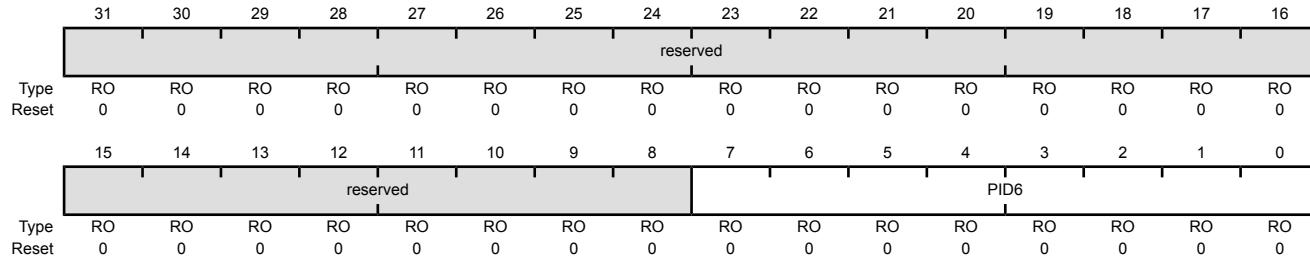
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

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 22: 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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

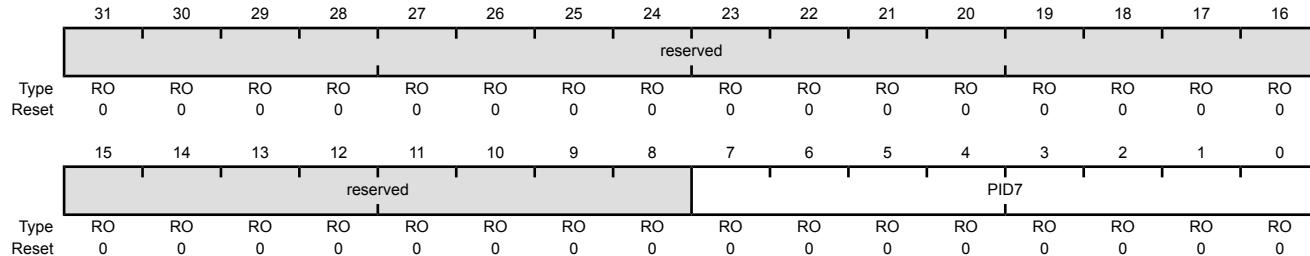
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

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	GPIO Peripheral ID Register[31:24]

## Register 23: 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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

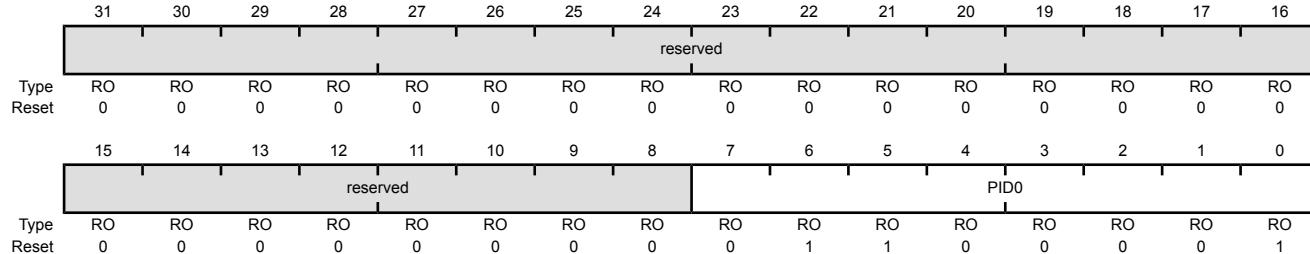
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

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 24: 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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

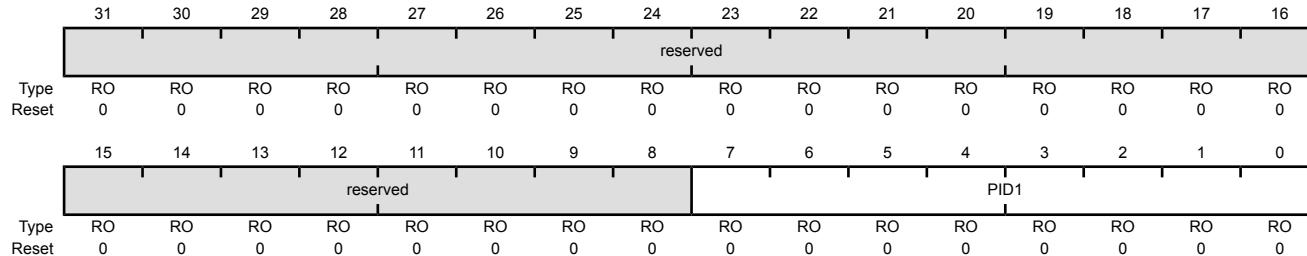
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

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	GPIO Peripheral ID Register[15:8] Can be used by software to identify the presence of this peripheral.

## Register 25: 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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

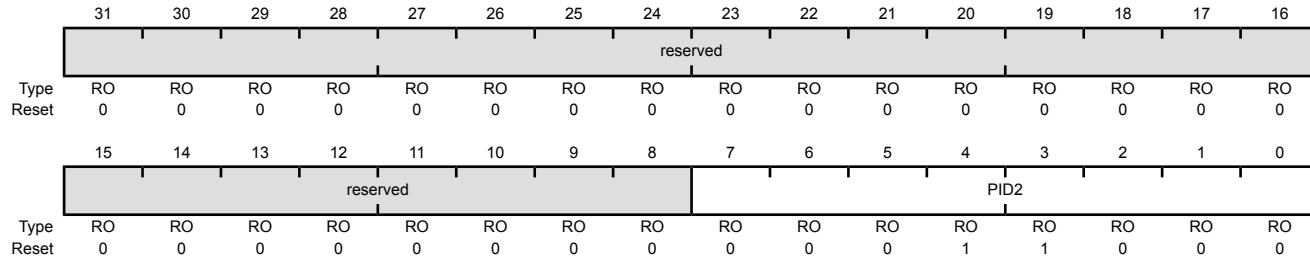
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

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 26: 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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

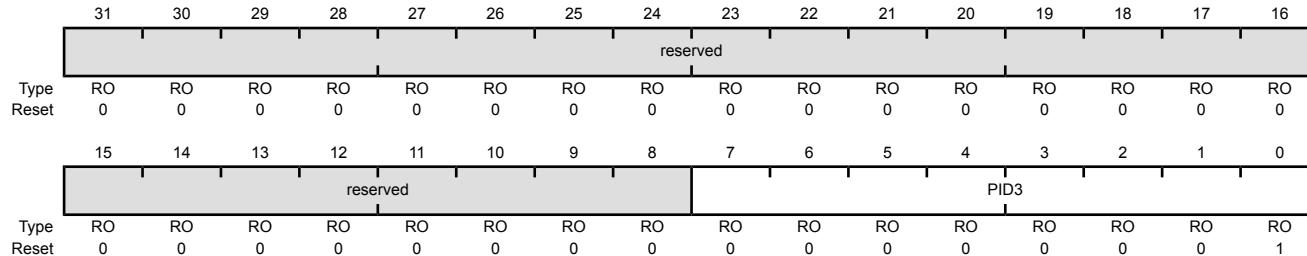
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

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	GPIO Peripheral ID Register[31:24] Can be used by software to identify the presence of this peripheral.

## Register 27: 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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

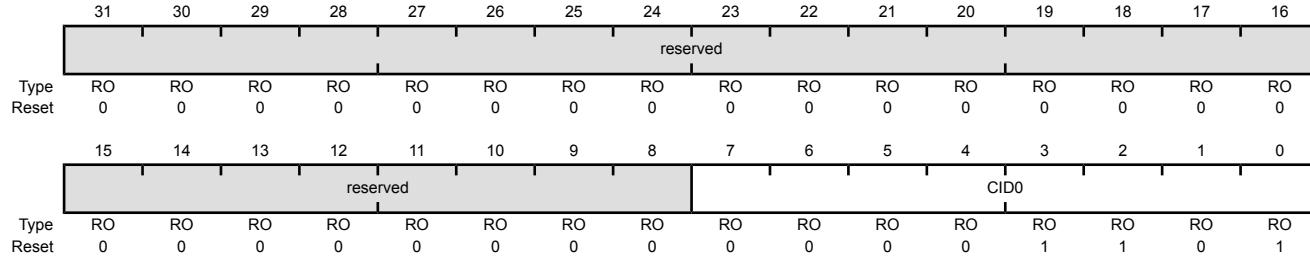
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

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	GPIO PrimeCell ID Register[7:0] Provides software a standard cross-peripheral identification system.

## Register 28: 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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

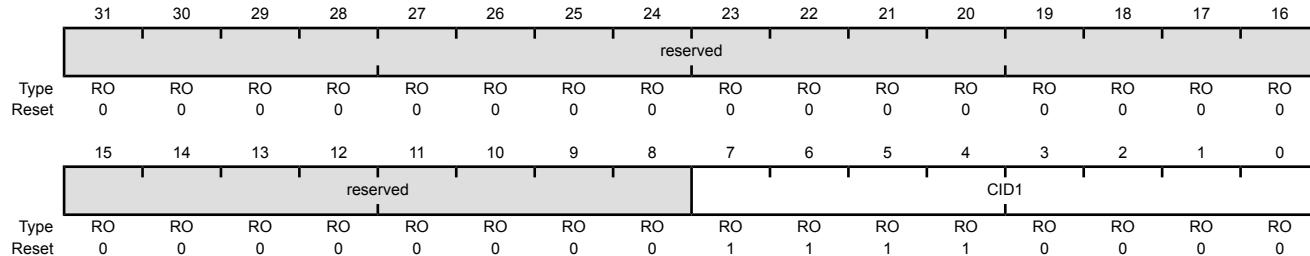
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

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	GPIO PrimeCell ID Register[15:8] Provides software a standard cross-peripheral identification system.

## Register 29: 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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

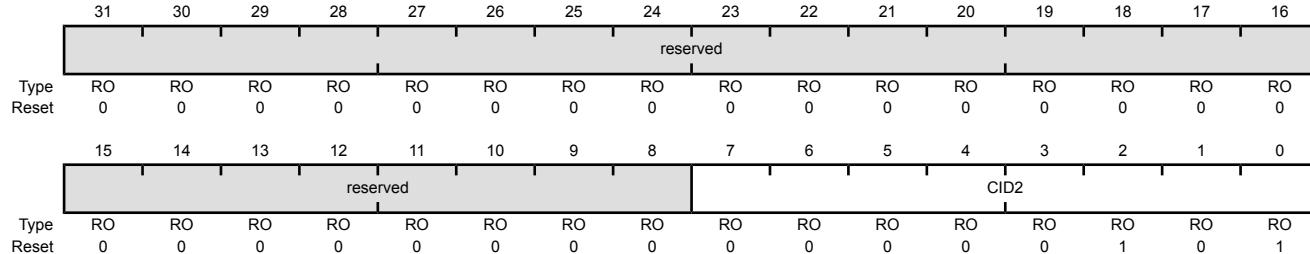
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

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 30: 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 base: 0x4000.4000

GPIO Port B base: 0x4000.5000

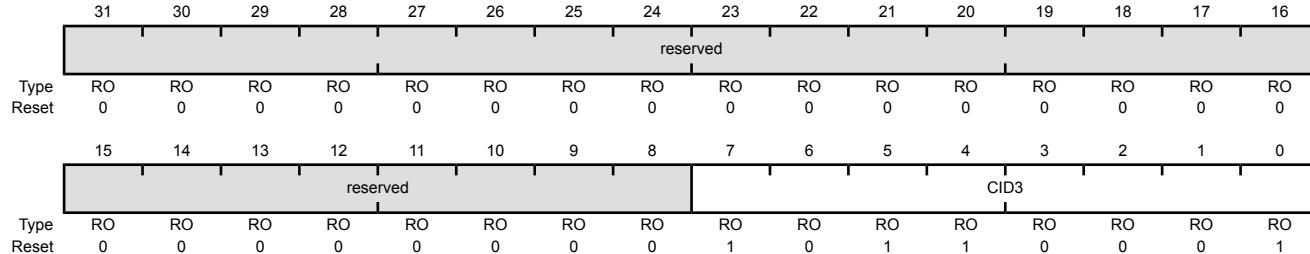
GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

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	GPIO PrimeCell ID Register[31:24] Provides software a standard cross-peripheral identification system.

## 8 General-Purpose Timers

Programmable timers can be used to count or time external events that drive the Timer input pins. The Stellaris® General-Purpose Timer Module (GPTM) contains three GPTM blocks (Timer0, Timer1, and Timer 2). 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).

In addition, timers can be used to trigger analog-to-digital conversions (ADC). The ADC 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 GPT Module is one timing resource available on the Stellaris microcontrollers. Other timer resources include the System Timer (SysTick) (see 86) and the PWM timer in the PWM module (see “PWM Timer” on page 503).

The General-Purpose Timers provide the following features:

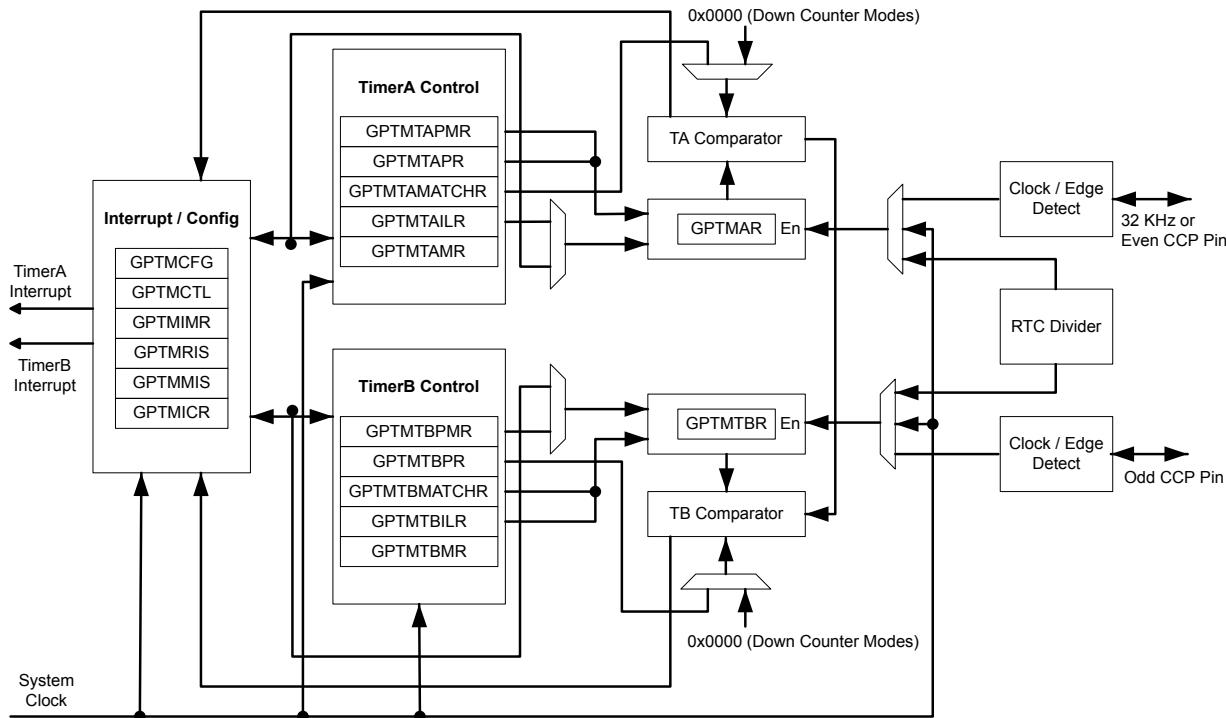
- Three General-Purpose Timer Modules (GPTM), each of which provides two 16-bit timers/counters. Each GPTM can be configured to operate independently:
  - As a single 32-bit timer
  - As one 32-bit Real-Time Clock (RTC) to event capture
  - For Pulse Width Modulation (PWM)
  - To trigger analog-to-digital 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 when the controller asserts CPU Halt flag during debug
  - ADC event trigger
- 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
  - 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

## 8.1 Block Diagram

**Note:** In Figure 8-1 on page 276, the specific CCP pins available depend on the Stellaris device. See Table 8-1 on page 276 for the available CCPs.

**Figure 8-1. GPTM Module Block Diagram**



**Table 8-1. 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

## 8.2 Signal Description

Table 8-2 on page 277 and Table 8-3 on page 277 list the external signals of the GP Timer module and describe the function of each. The GP Timer signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Assignment"

lists the possible GPIO pin placements for these GP Timer signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 253) should be set to choose the GP Timer function. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 232.

**Table 8-2. General-Purpose Timers Signals (48QFP)**

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
CCP0	45	I/O	TTL	Capture/Compare/PWM 0.
CCP1	13	I/O	TTL	Capture/Compare/PWM 1.
CCP2	46	I/O	TTL	Capture/Compare/PWM 2.
CCP3	12	I/O	TTL	Capture/Compare/PWM 3.
CCP4	11	I/O	TTL	Capture/Compare/PWM 4.
CCP5	43	I/O	TTL	Capture/Compare/PWM 5.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

**Table 8-3. General-Purpose Timers Signals (48QFN)**

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
CCP0	45	I/O	TTL	Capture/Compare/PWM 0.
CCP1	13	I/O	TTL	Capture/Compare/PWM 1.
CCP2	46	I/O	TTL	Capture/Compare/PWM 2.
CCP3	12	I/O	TTL	Capture/Compare/PWM 3.
CCP4	11	I/O	TTL	Capture/Compare/PWM 4.
CCP5	43	I/O	TTL	Capture/Compare/PWM 5.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

## 8.3 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, two prescaler 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 288), the **GPTM TimerA Mode (GPTMTAMR)** register (see page 289), and the **GPTM TimerB Mode (GPTMTBMR)** register (see page 291). 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.

### 8.3.1 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 302) and the **GPTM TimerB Interval Load (GPTMTBILR)** register (see page 303). The prescale counters are initialized to 0x00: the **GPTM TimerA Prescale (GPTMTAPR)** register (see page 306) and the **GPTM TimerB Prescale (GPTMTBPR)** register (see page 307).

### 8.3.2 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 302
- **GPTM TimerB Interval Load (GPTMTBILR)** register [15:0], see page 303
- **GPTM TimerA (GPTMTAR)** register [15:0], see page 310
- **GPTM TimerB (GPTMTBR)** register [15:0], see page 311

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]`

#### 8.3.2.1 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 289), 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 293), 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 298), and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register (see page 300). If the time-out interrupt is enabled in the **GPTM Interrupt Mask (GPTMIMR)** register (see page 296), the GPTM also sets the TATOMIS bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register (see page 299). The ADC trigger is enabled by setting the TAOTE bit in **GPTMCTL**.

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 set, the timer freezes counting while the processor is halted by the debugger. The timer resumes counting when the processor resumes execution.

#### 8.3.2.2 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 304) by the controller.

The input clock on an even CCP input 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 **RTCRIS** bit in **GPTMRIS**. If the RTC interrupt is enabled in **GPTMIMR**, the GPTM also sets the **RTCMIS** bit in **GPTMMIS** 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**.

### 8.3.3 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 288). 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.

#### 8.3.3.1 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 **GPTMIMR**, the GPTM also sets the **TnTOMIS** bit in **GPTMISR** and generates a controller interrupt. The ADC trigger is enabled by setting the **TnOTE** bit in the **GPTMCTL** register.

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 **TnSTALL** bit in the **GPTMCTL** register is set, the timer freezes counting while the processor is halted by the debugger. The timer resumes counting when the processor resumes execution.

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 8-4. 16-Bit Timer With Prescaler Configurations**

Prescale	#Clock ( $T_c$ ) <sup>a</sup>	Max Time	Units
00000000	1	1.3107	mS

**Table 8-4. 16-Bit Timer With Prescaler Configurations (continued)**

Prescale	#Clock (T c) <sup>a</sup>	Max Time	Units
00000001	2	2.6214	μs
00000010	3	3.9322	μs
-----	--	--	--
11111101	254	332.9229	μs
11111110	255	334.2336	μs
11111111	256	335.5443	μs

a. Tc is the clock period.

### 8.3.3.2 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 TnCMR bit of the **GPTMTnMR** register must be set to 0. The type of edge that the timer counts is determined by the TnEVENT 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.

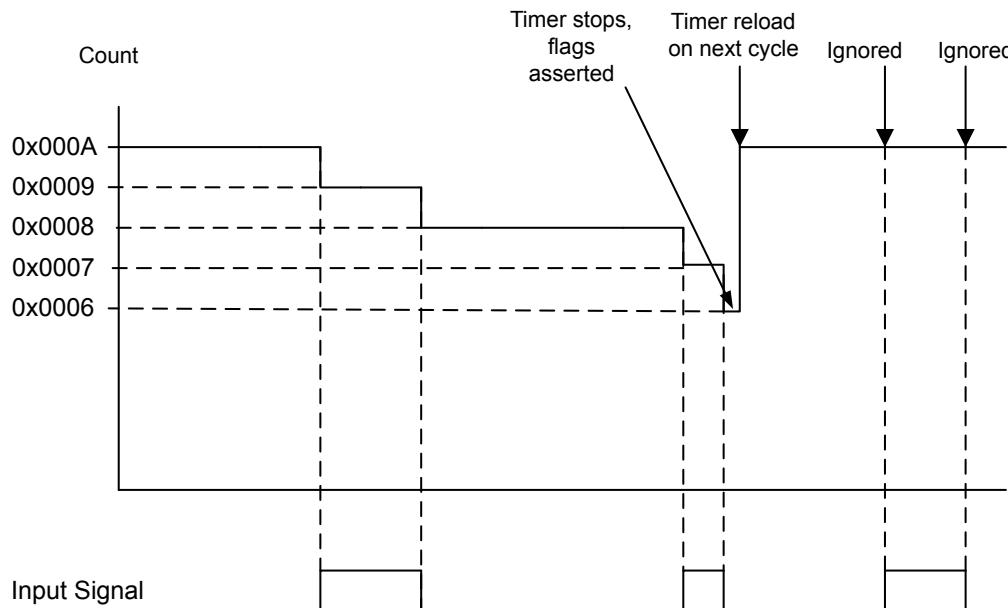
When software writes the TnEN bit in the **GPTM Control (GPTMCTL)** register, the timer is enabled for event capture. Each input event on the CCP pin decrements the counter by 1 until the event count matches **GPTMTnMATCHR**. When the counts match, the GPTM asserts the CnMRIS bit in the **GPTMRIS** register (and the CnMMIS 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 TnEN bit in the **GPTMCTL** register. Once the event count has been reached, all further events are ignored until TnEN is re-enabled by software.

Figure 8-2 on page 281 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 TnEN bit after the current count matches the value in the **GPTMTnMATCHR** register.

**Figure 8-2. 16-Bit Input Edge Count Mode Example**



### 8.3.3.3 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). The timer is capable of capturing three types of events: rising edge, falling edge, or 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 **GPTMCTL** register.

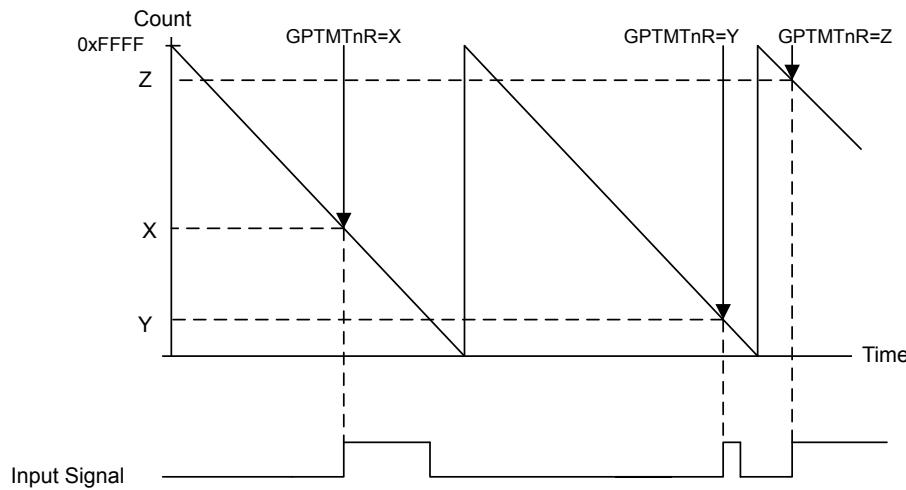
When software writes the **TnEN** bit in the **GPTMCTL** register, the timer is enabled for event capture. When the selected input event is detected, the current **Tn** counter value is captured in the **GPTMTnR** register and is available to be read by the controller. The GPTM then asserts the **CnERIS** 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 **GPTMTnILR** register.

Figure 8-3 on page 282 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 8-3. 16-Bit Input Edge Time Mode Example



#### 8.3.3.4 16-Bit PWM Mode

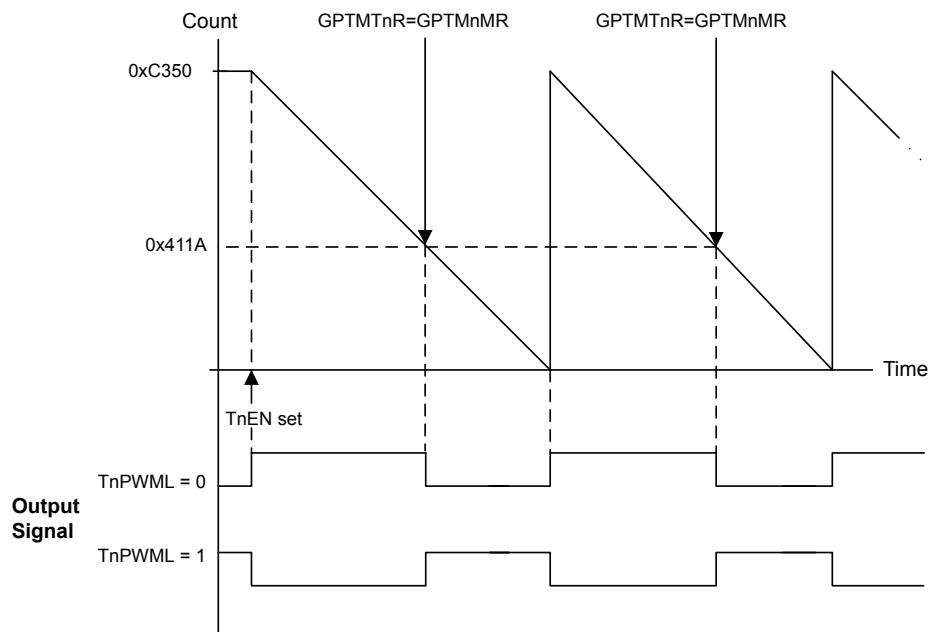
**Note:** The prescaler is not available in 16-Bit PWM mode.

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**. In this mode, the PWM frequency and period are synchronous events and therefore guaranteed to be glitch free. PWM mode is enabled with the **GPTMTnMR** register by setting the **TnAMS** bit to 0x1, the **TnCMR** bit to 0x0, and the **TnMR** field to 0x2.

When software writes the **TnEN** 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 **TnEN** 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 Timer Match Register (GPTMTnMATCHR)**. Software has the capability of inverting the output PWM signal by setting the **TnPWML** bit in the **GPTMCTL** register.

Figure 8-4 on page 283 shows how to generate an output PWM with a 1-ms period and a 66% duty cycle assuming a 50-MHz input clock and **TnPWML**=0 (duty cycle would be 33% for the **TnPWML**=1 configuration). For this example, the start value is **GPTMTnILR**=0xC350 and the match value is **GPTMTnMATCHR**=0x411A.

**Figure 8-4. 16-Bit PWM Mode Example**

## 8.4 Initialization and Configuration

To use the general-purpose timers, the peripheral clock must be enabled by setting the `TIMER0`, `TIMER1`, and `TIMER2` bits in the **RCGC1** register.

This section shows module initialization and configuration examples for each of the supported timer modes.

### 8.4.1 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 284. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode does not stop counting after it times out.

#### 8.4.2 32-Bit Real-Time Clock (RTC) Mode

To use the RTC mode, the timer must have a 32.768-KHz input signal on an even CCP input. 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 RTCCEN 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 GPTM asserts the RTCRIS bit in the **GPTMRIS** register and continues counting until Timer A is disabled or a hardware reset. The interrupt is cleared by writing the RTCCINT bit in the **GPTMICR** register.

#### 8.4.3 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 Timer 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 284. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode does not stop counting after it times out.

#### 8.4.4 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 285 through step 9 on page 285.

#### 8.4.5 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.

#### 8.4.6 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 `TnPWL` 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.

### 8.5 Register Map

Table 8-5 on page 286 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

Note that the Timer module clock must be enabled before the registers can be programmed (see page 199). There must be a delay of 3 system clocks after the Timer module clock is enabled before any Timer module registers are accessed.

**Table 8-5. Timers Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	GPTMCFG	R/W	0x0000.0000	GPTM Configuration	288
0x004	GPTMTAMR	R/W	0x0000.0000	GPTM TimerA Mode	289
0x008	GPTMTBMR	R/W	0x0000.0000	GPTM TimerB Mode	291
0x00C	GPTMCTL	R/W	0x0000.0000	GPTM Control	293

**Table 8-5. Timers Register Map (continued)**

Offset	Name	Type	Reset	Description	See page
0x018	GPTMIMR	R/W	0x0000.0000	GPTM Interrupt Mask	296
0x01C	GPTMRIS	RO	0x0000.0000	GPTM Raw Interrupt Status	298
0x020	GPTMMIS	RO	0x0000.0000	GPTM Masked Interrupt Status	299
0x024	GPTMICR	W1C	0x0000.0000	GPTM Interrupt Clear	300
0x028	GPTMTAILR	R/W	0xFFFF.FFFF	GPTM TimerA Interval Load	302
0x02C	GPTMTBILR	R/W	0x0000.FFFF	GPTM TimerB Interval Load	303
0x030	GPTMTAMATCHR	R/W	0xFFFF.FFFF	GPTM TimerA Match	304
0x034	GPTMTBMATCHR	R/W	0x0000.FFFF	GPTM TimerB Match	305
0x038	GPTMTAPR	R/W	0x0000.0000	GPTM TimerA Prescale	306
0x03C	GPTMTBPR	R/W	0x0000.0000	GPTM TimerB Prescale	307
0x040	GPTMTAPMR	R/W	0x0000.0000	GPTM TimerA Prescale Match	308
0x044	GPTMTBPMR	R/W	0x0000.0000	GPTM TimerB Prescale Match	309
0x048	GPTMTAR	RO	0xFFFF.FFFF	GPTM TimerA	310
0x04C	GPTMTBR	RO	0x0000.FFFF	GPTM TimerB	311

## 8.6 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)

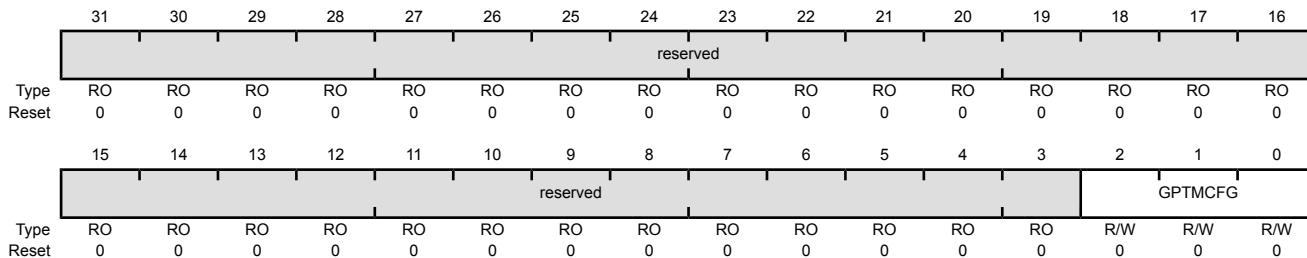
Timer0 base: 0x4003.0000

Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

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	<p>GPTM Configuration</p> <p>The GPTMCFG values are defined as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>32-bit timer configuration.</td> </tr> <tr> <td>0x1</td> <td>32-bit real-time clock (RTC) counter configuration.</td> </tr> <tr> <td>0x2</td> <td>Reserved</td> </tr> <tr> <td>0x3</td> <td>Reserved</td> </tr> <tr> <td>0x4-0x7</td> <td>16-bit timer configuration, function is controlled by bits 1:0 of <b>GPTMTAMR</b> and <b>GPTMTBMR</b>.</td> </tr> </tbody> </table>	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> .
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

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															
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	TAAMS	R/W	0	GPTM TimerA Alternate Mode Select The TAAMS values are defined as follows:

#### Value Description

- 0 Capture mode is enabled.
- 1 PWM mode is enabled.

**Note:** To enable PWM mode, you must also clear the **TACMR** bit and set the **TAMR** field to 0x2.

2	TACMR	R/W	0	GPTM TimerA Capture Mode
				The TACMR values are defined as follows:

#### Value Description

- 0 Edge-Count mode
- 1 Edge-Time mode

Bit/Field	Name	Type	Reset	Description										
1:0	TAMR	R/W	0x0	GPTM TimerA Mode The TAMR values are defined as follows:  <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> The Timer mode is based on the timer configuration defined by bits 2:0 in the <b>GPTMCFG</b> register (16-or 32-bit). In 16-bit timer configuration, TAMR controls the 16-bit timer modes for TimerA. In 32-bit timer configuration, this register controls the mode and the contents of <b>GPTMTBMR</b> are ignored.	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)

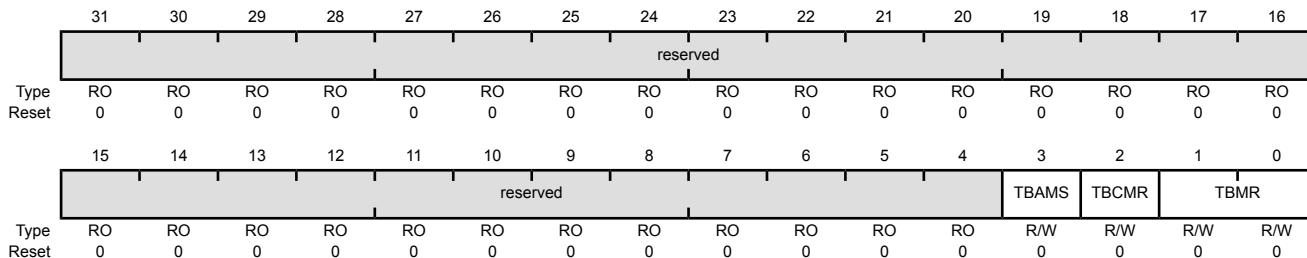
Timer0 base: 0x4003.0000

Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

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:

#### Value Description

- 0 Capture mode is enabled.
- 1 PWM mode is enabled.

**Note:** To enable PWM mode, you must also clear the **TBCMR** bit and set the **TBMR** field to 0x2.

2	TBCMR	R/W	0	GPTM TimerB Capture Mode
				The <b>TBCMR</b> values are defined as follows:

#### Value Description

- 0 Edge-Count mode
- 1 Edge-Time mode

Bit/Field	Name	Type	Reset	Description										
1:0	TBMR	R/W	0x0	GPTM TimerB Mode The TBMR values are defined as follows:  <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> The timer mode is based on the timer configuration defined by bits 2:0 in the <b>GPTMCFG</b> register. In 16-bit timer configuration, these bits control the 16-bit timer modes for TimerB. In 32-bit timer configuration, this register's contents are ignored and <b>GPTMTAMR</b> is used.	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

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 ADC trigger is disabled. 1 The output TimerB ADC trigger is enabled.  In addition, the ADC must be enabled and the timer selected as a trigger source with the <b>EMn</b> bit in the <b>ADCEMUX</b> register (see page 351).
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 <b>TBEVENT</b> values are defined as follows:</p> <table> <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 Timer B Stall Enable</p> <p>The <b>TBSTALL</b> values are defined as follows:</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Timer B continues counting while the processor is halted by the debugger.</td></tr> <tr> <td>1</td><td>Timer B freezes counting while the processor is halted by the debugger.</td></tr> </tbody> </table> <p>If the processor is executing normally, the <b>TBSTALL</b> bit is ignored.</p>	Value	Description	0	Timer B continues counting while the processor is halted by the debugger.	1	Timer B freezes counting while the processor is halted by the debugger.				
Value	Description													
0	Timer B continues counting while the processor is halted by the debugger.													
1	Timer B freezes counting while the processor is halted by the debugger.													
8	TBEN	R/W	0	<p>GPTM TimerB Enable</p> <p>The <b>TBEN</b> values are defined as follows:</p> <table> <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 <b>TAPWML</b> values are defined as follows:</p> <table> <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 <b>TAOTE</b> values are defined as follows:</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The output TimerA ADC trigger is disabled.</td></tr> <tr> <td>1</td><td>The output TimerA ADC trigger is enabled.</td></tr> </tbody> </table> <p>In addition, the ADC must be enabled and the timer selected as a trigger source with the <b>EMn</b> bit in the <b>ADCEMUX</b> register (see page 351).</p>	Value	Description	0	The output TimerA ADC trigger is disabled.	1	The output TimerA ADC trigger is enabled.				
Value	Description													
0	The output TimerA ADC trigger is disabled.													
1	The output TimerA ADC 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> <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> <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 Timer A Stall Enable</p> <p>The <code>TASTALL</code> values are defined as follows:</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Timer A continues counting while the processor is halted by the debugger.</td></tr> <tr> <td>1</td><td>Timer A freezes counting while the processor is halted by the debugger.</td></tr> </tbody> </table> <p>If the processor is executing normally, the <code>TASTALL</code> bit is ignored.</p>	Value	Description	0	Timer A continues counting while the processor is halted by the debugger.	1	Timer A freezes counting while the processor is halted by the debugger.				
Value	Description													
0	Timer A continues counting while the processor is halted by the debugger.													
1	Timer A freezes counting while the processor is halted by the debugger.													
0	TAEN	R/W	0	<p>GPTM TimerA Enable</p> <p>The <code>TAEN</code> values are defined as follows:</p> <table> <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 <b>GPTMCFG</b> 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 <b>GPTMCFG</b> register.				
Value	Description													
0	TimerA is disabled.													
1	TimerA is enabled and begins counting or the capture logic is enabled based on the <b>GPTMCFG</b> 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

Offset 0x018

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					CBEIM	CBMIM	TBTOIM	reserved				RTCIM	CAEIM	CAMIM	TATOIM
Type	RO	RO	RO	RO	RO	R/W	R/W	R/W	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
31:11	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	GPTM RTC Interrupt Mask The RTCIM values are defined as follows:  Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
2	CAEIM	R/W	0	GPTM CaptureA Event Interrupt Mask The CAEIM values are defined as follows:  Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
1	CAMIM	R/W	0	GPTM CaptureA Match Interrupt Mask The CAMIM values are defined as follows:  Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
0	TATOIM	R/W	0	GPTM TimerA Time-Out Interrupt Mask The TATOIM values are defined as follows:  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 **GPTMMR** 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

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				CBERIS	CBMRIS	TBTORIS	reserved				RTCRIS	CAERIS	CAMRIS	TATORIS	
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	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 is the TimerA time-out interrupt status prior to masking.

## Register 7: GPTM Masked Interrupt Status (GPTMMIS), offset 0x020

This register shows 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)

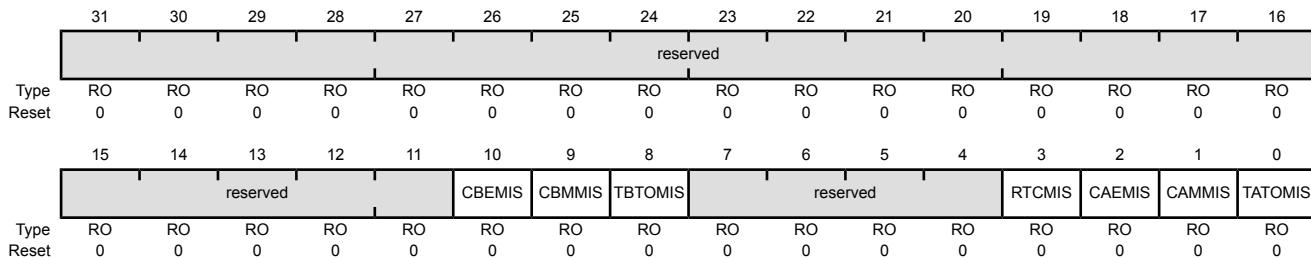
Timer0 base: 0x4003.0000

Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Offset 0x020

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	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

Offset 0x024

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					CBECINT	CBMCINT	TBTIOCINT	reserved				RTCCINT	CAECINT	CAMCINT	TATOCINT
Type	RO	RO	RO	RO	RO	W1C	W1C	W1C	RO	RO	RO	RO	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:11	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 CBECINT 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 CBMCINT values are defined as follows:  Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
8	TBTIOCINT	W1C	0	GPTM TimerB Time-Out Interrupt Clear The TBTIOCINT 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	GPTM RTC Interrupt Clear The RTCCINT values are defined as follows:  Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
2	CAECINT	W1C	0	GPTM CaptureA Event Interrupt Clear The CAECINT values are defined as follows:  Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
1	CAMCINT	W1C	0	GPTM CaptureA Match Interrupt Clear The CAMCINT values are defined as follows:  Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
0	TATOCINT	W1C	0	GPTM TimerA Time-Out Interrupt Clear The TATOCINT values are defined as follows:  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  
 Offset 0x028  
 Type R/W, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TAILRH															
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
	TAILRL															
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	TAILRH	R/W	0xFFFF	<p>GPTM TimerA Interval Load Register High</p> <p>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>.</p> <p>In 16-bit mode, this field reads as 0 and does not have an effect on the state of <b>GPTMTBILR</b>.</p>
15:0	TAILRL	R/W	0xFFFF	<p>GPTM TimerA Interval Load Register Low</p> <p>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>.</p>

## 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)

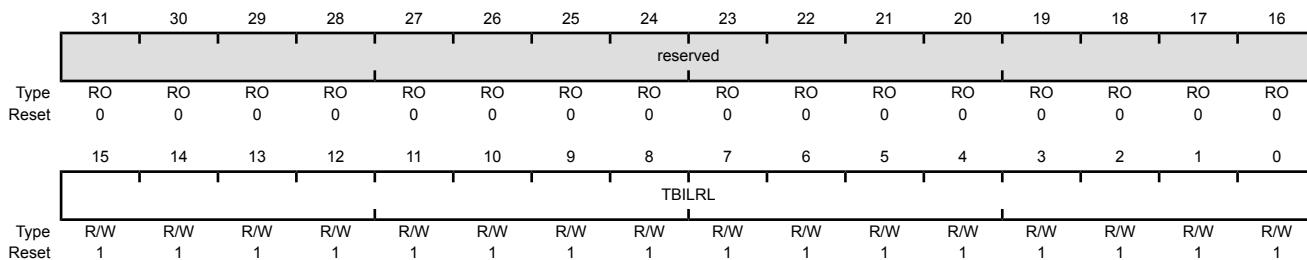
Timer0 base: 0x4003.0000

Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Offset 0x02C

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	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 <b>GPTMTBILR</b> . In 32-bit mode, writes are ignored, and reads return the current value of <b>GPTMTBILR</b> .

## 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)

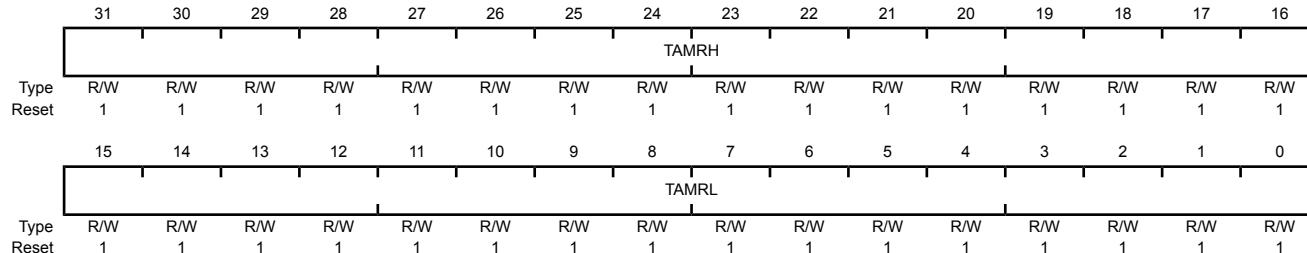
Timer0 base: 0x4003.0000

Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Offset 0x030

Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:16	TAMRH	R/W	0xFFFF	<p>GPTM TimerA Match Register High</p> <p>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.</p> <p>In 16-bit mode, this field reads as 0 and does not have an effect on the state of <b>GPTMTBMATCHR</b>.</p>
15:0	TAMRL	R/W	0xFFFF	<p>GPTM TimerA Match Register Low</p> <p>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.</p> <p>When configured for PWM mode, this value along with <b>GPTMTAILR</b>, determines the duty cycle of the output PWM signal.</p> <p>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.</p>

## 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

Offset 0x034

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
	TBMRL															
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	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)

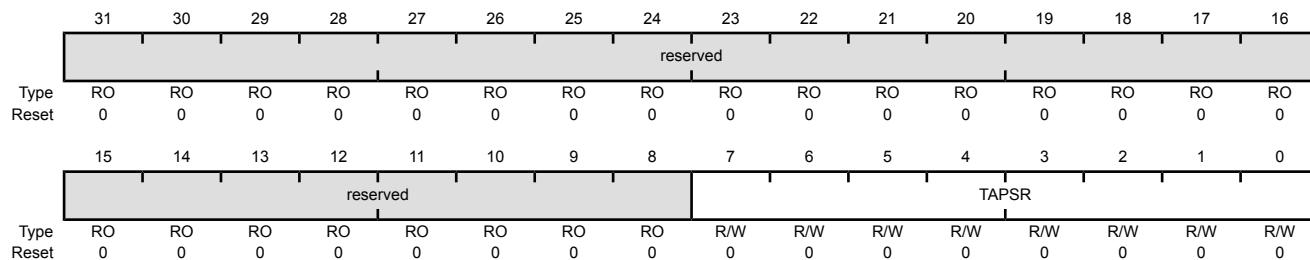
Timer0 base: 0x4003.0000

Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

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 8-4 on page 279 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

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	<p>GPTM TimerB Prescale</p> <p>The register loads this value on a write. A read returns the current value of this register.</p> <p>Refer to Table 8-4 on page 279 for more details and an example.</p>

## Register 15: GPTM TimerA Prescale Match (GPTMTAPMR), offset 0x040

This register effectively extends the range of **GPTMTAMATCHR** to 24 bits when operating in 16-bit one-shot or periodic mode.

### GPTM TimerA Prescale Match (GPTMTAPMR)

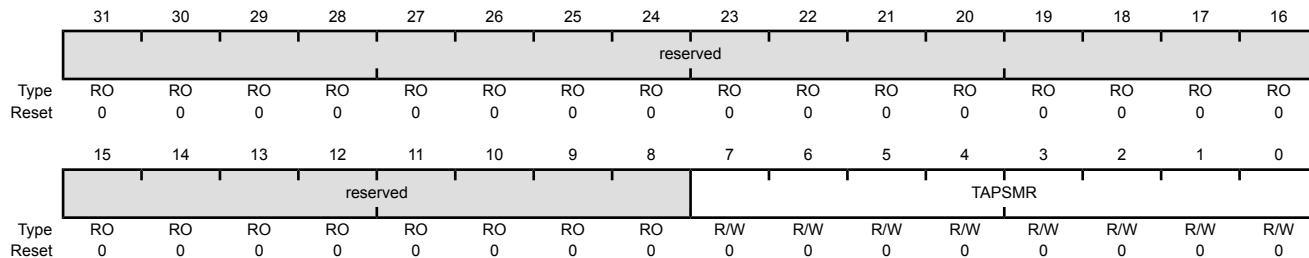
Timer0 base: 0x4003.0000

Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Offset 0x040

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	TAPSMR	R/W	0x00	GPTM TimerA Prescale Match This value is used alongside <b>GPTMTAMATCHR</b> to detect timer match events while using a prescaler.

## Register 16: GPTM TimerB Prescale Match (GPTMTBPMR), offset 0x044

This register effectively extends the range of **GPTMTBMR** to 24 bits when operating in 16-bit one-shot or periodic mode.

### GPTM TimerB Prescale Match (GPTMTBPMR)

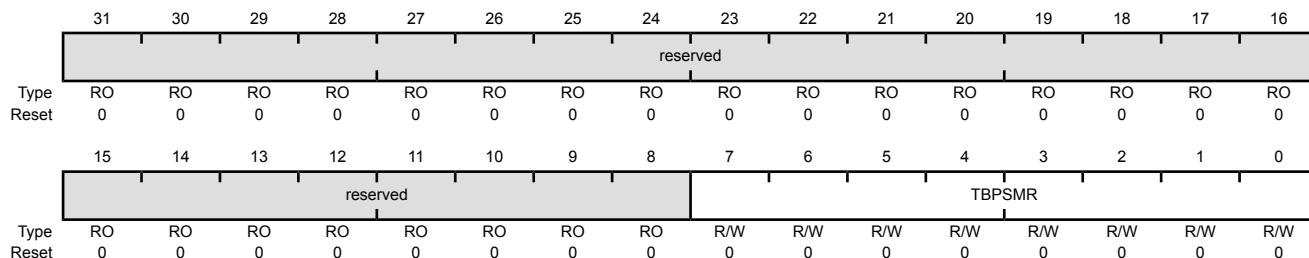
Timer0 base: 0x4003.0000

Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Offset 0x044

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	TBPSMR	R/W	0x00	GPTM TimerB Prescale Match This value is used alongside <b>GPTMTBMR</b> to detect timer match events while using a prescaler.

## Register 17: 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 number of edges that have occurred.

### GPTM TimerA (GPTMTAR)

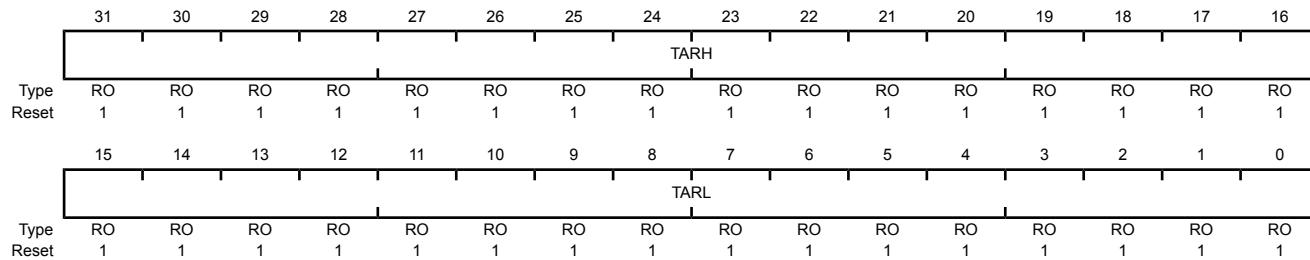
Timer0 base: 0x4003.0000

Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Offset 0x048

Type RO, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:16	TARH	RO	0xFFFF	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 number of edges that have occurred.

## Register 18: 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 number of edges that have occurred.

### GPTM TimerB (GPTMTBR)

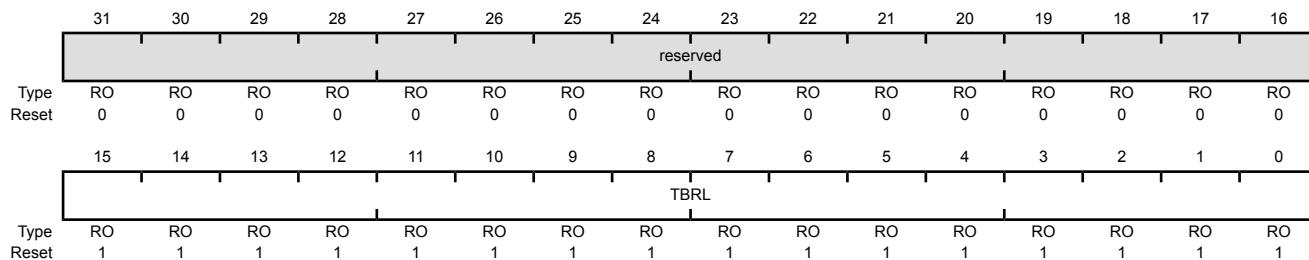
Timer0 base: 0x4003.0000

Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Offset 0x04C

Type RO, 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	TBRL	RO	0xFFFF	GPTM TimerB A read returns the current value of the <b>GPTM TimerB Count Register</b> , except in Input Edge-Count mode, when it returns the number of edges that have occurred.

## 9 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.

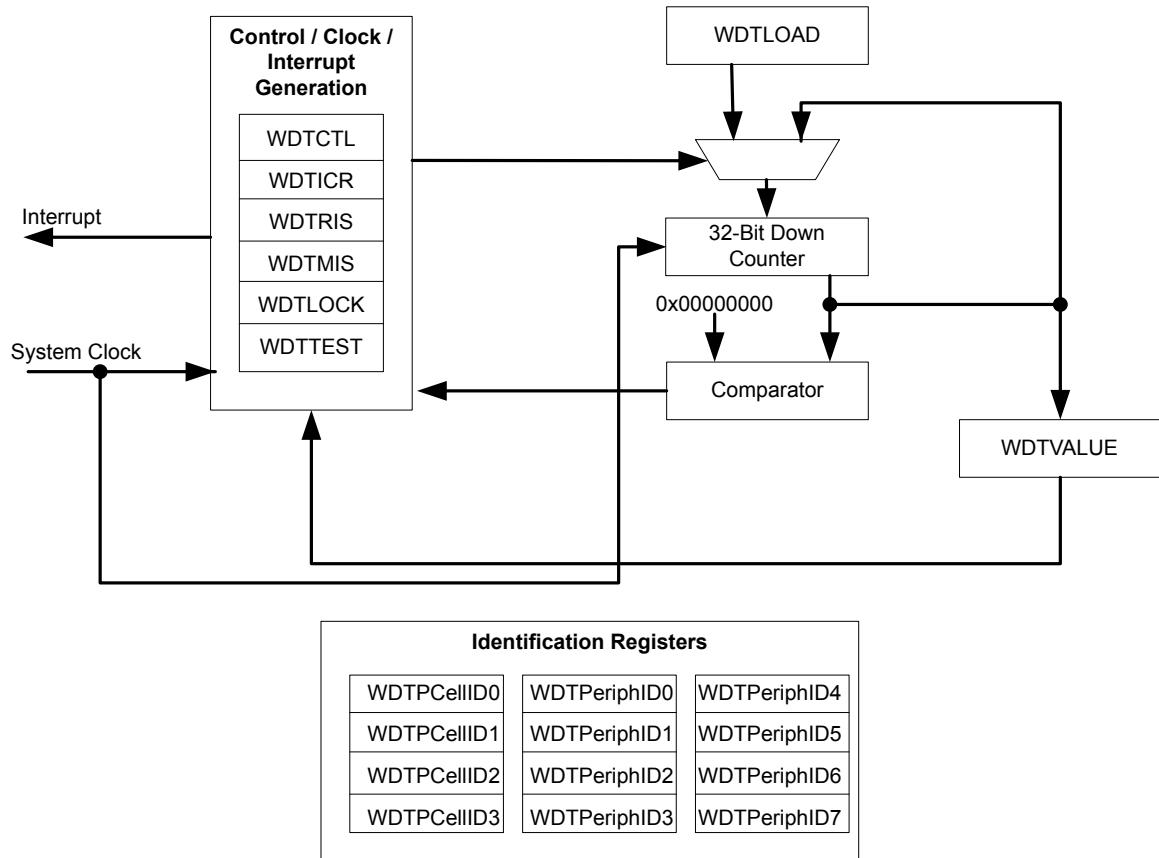
The Stellaris® Watchdog Timer module has the following features:

- 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

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.

## 9.1 Block Diagram

Figure 9-1. WDT Module Block Diagram



## 9.2 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.

## 9.3 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.

## 9.4 Register Map

Table 9-1 on page 314 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 9-1. Watchdog Timer Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	WDTLOAD	R/W	0xFFFF.FFFF	Watchdog Load	316
0x004	WDTVALUE	RO	0xFFFF.FFFF	Watchdog Value	317
0x008	WDTCTL	R/W	0x0000.0000	Watchdog Control	318
0x00C	WDTICR	WO	-	Watchdog Interrupt Clear	319
0x010	WDTRIS	RO	0x0000.0000	Watchdog Raw Interrupt Status	320
0x014	WDTMIS	RO	0x0000.0000	Watchdog Masked Interrupt Status	321
0x418	WDTTEST	R/W	0x0000.0000	Watchdog Test	322
0xC00	WDTLOCK	R/W	0x0000.0000	Watchdog Lock	323
0xFD0	WDTPeriphID4	RO	0x0000.0000	Watchdog Peripheral Identification 4	324
0xFD4	WDTPeriphID5	RO	0x0000.0000	Watchdog Peripheral Identification 5	325
0xFD8	WDTPeriphID6	RO	0x0000.0000	Watchdog Peripheral Identification 6	326
0xFDC	WDTPeriphID7	RO	0x0000.0000	Watchdog Peripheral Identification 7	327
0xFE0	WDTPeriphID0	RO	0x0000.0005	Watchdog Peripheral Identification 0	328
0xFE4	WDTPeriphID1	RO	0x0000.0018	Watchdog Peripheral Identification 1	329
0xFE8	WDTPeriphID2	RO	0x0000.0018	Watchdog Peripheral Identification 2	330

**Table 9-1. Watchdog Timer Register Map (*continued*)**

Offset	Name	Type	Reset	Description	See page
0xFEC	WDTPeriphID3	RO	0x0000.0001	Watchdog Peripheral Identification 3	331
0xFF0	WDTPCellID0	RO	0x0000.000D	Watchdog PrimeCell Identification 0	332
0xFF4	WDTPCellID1	RO	0x0000.00F0	Watchdog PrimeCell Identification 1	333
0xFF8	WDTPCellID2	RO	0x0000.0005	Watchdog PrimeCell Identification 2	334
0xFFC	WDTPCellID3	RO	0x0000.00B1	Watchdog PrimeCell Identification 3	335

## 9.5 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

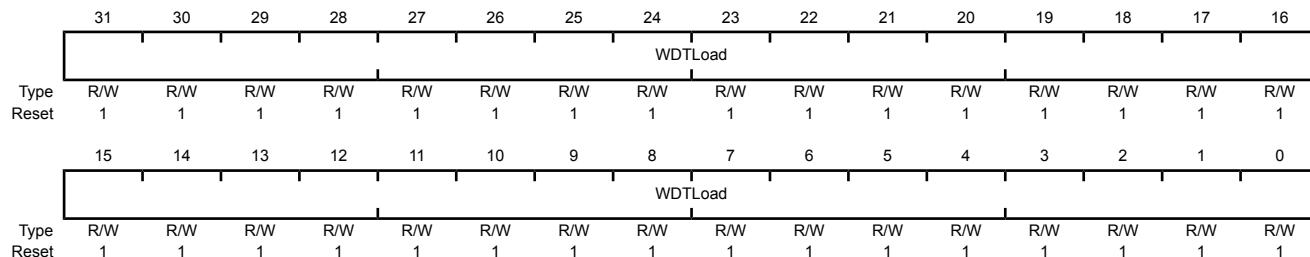
This register is the 32-bit interval value used by the 32-bit counter. When this register is written, the value is immediately loaded and the counter restarts counting down from the new value. If the **WDTLOAD** register is loaded with 0x0000.0000, an interrupt is immediately generated.

### Watchdog Load (WDTLOAD)

Base 0x4000.0000

Offset 0x000

Type R/W, reset 0xFFFF.FFFF



Bit/Field                  Name                  Type                  Reset                  Description

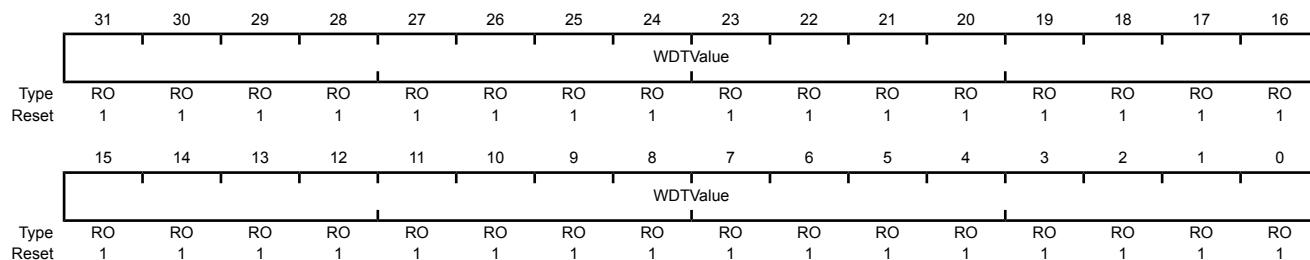
31:0                  WDTLoad                  R/W                  0xFFFF.FFFF                  Watchdog Load Value

## 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.

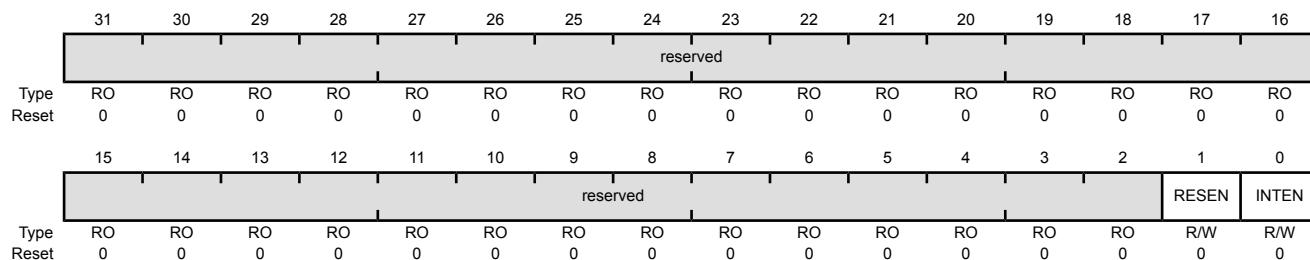
### Register 3: Watchdog Control (WDTCTL), offset 0x008

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



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:  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:  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

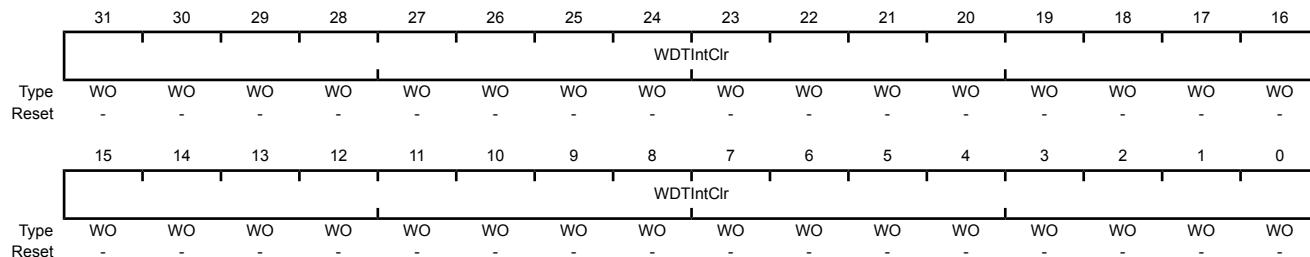
This register is the interrupt clear register. A write of any value to this register clears the Watchdog interrupt and reloads the 32-bit counter from the **WDTLOAD** register. Value for a read or reset is indeterminate.

Watchdog Interrupt Clear (WDTICR)

Base 0x4000.0000

Offset 0x00C

Type WO, reset -



Bit/Field                  Name                  Type                  Reset                  Description

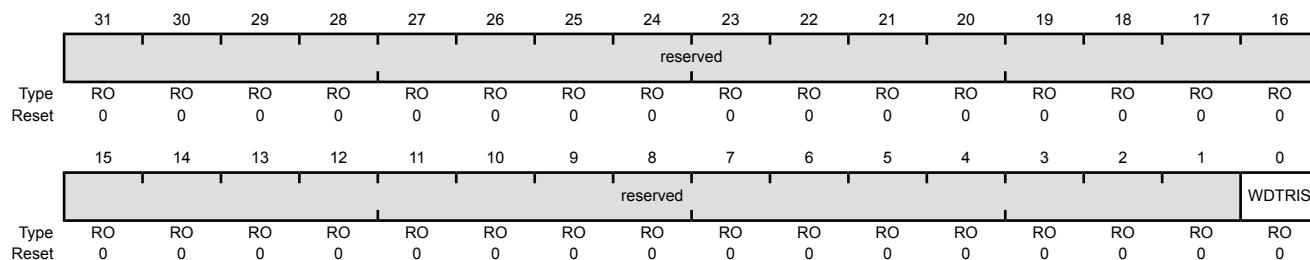
31:0                  WDTIntClr                  WO                  -                  Watchdog Interrupt Clear

## 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



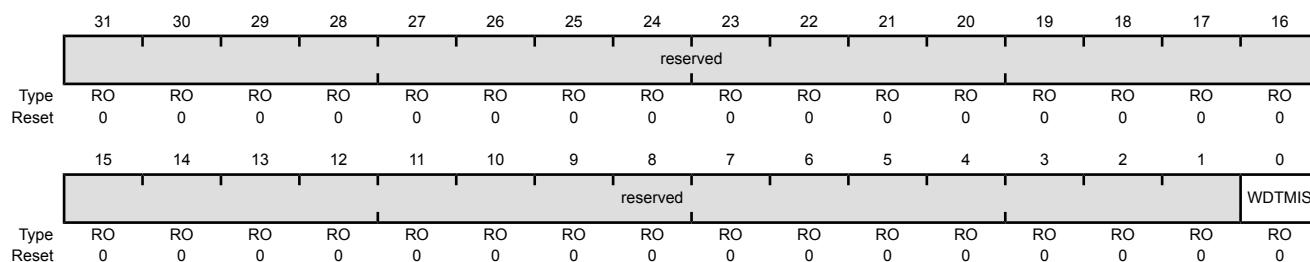
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	WDTRIS	RO	0	Watchdog Raw Interrupt Status Gives the 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



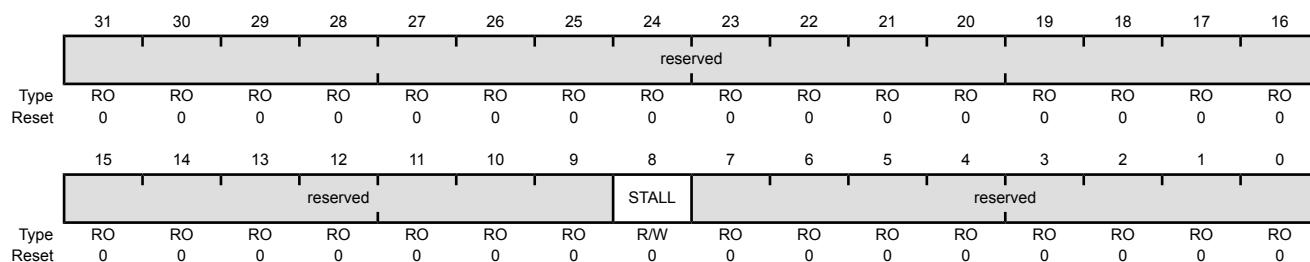
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	WDTMIS	RO	0	Watchdog Masked Interrupt Status Gives 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



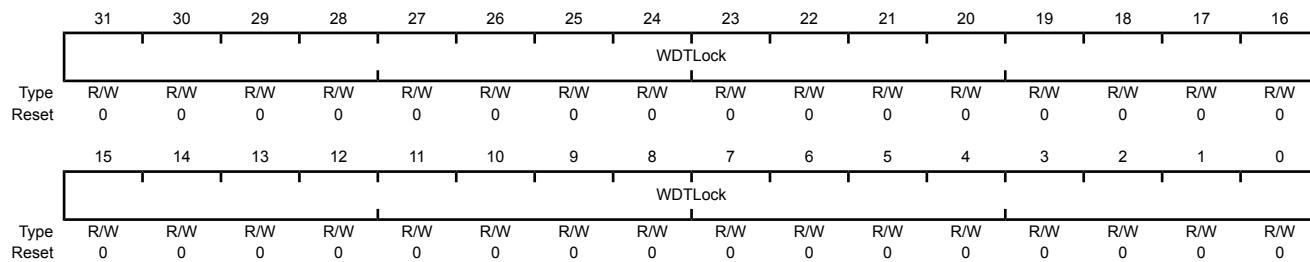
Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	STALL	R/W	0	Watchdog Stall Enable When set to 1, if the Stellaris microcontroller is stopped with a debugger, the watchdog timer stops counting. Once the microcontroller is restarted, the watchdog timer resumes counting.
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



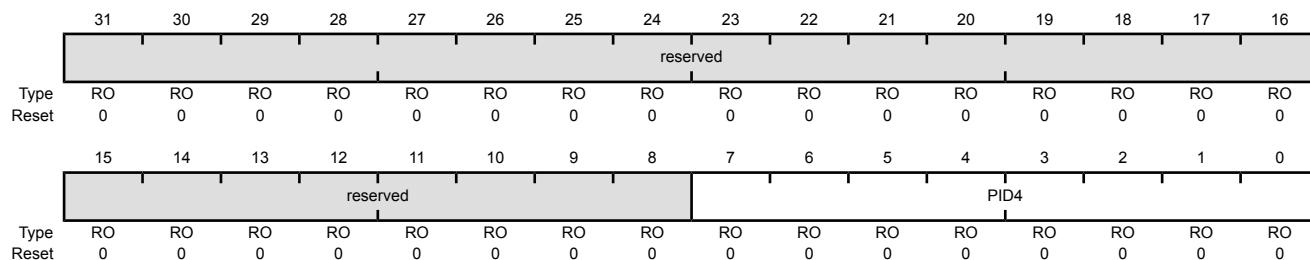
Bit/Field	Name	Type	Reset	Description						
31:0	WDTLock	R/W	0x0000	Watchdog Lock						
A write of the value 0x1ACC.E551 unlocks the watchdog registers for write access. A write of any other value reapplies the lock, preventing any register updates.										
A read of this register returns the following values:										
<table> <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



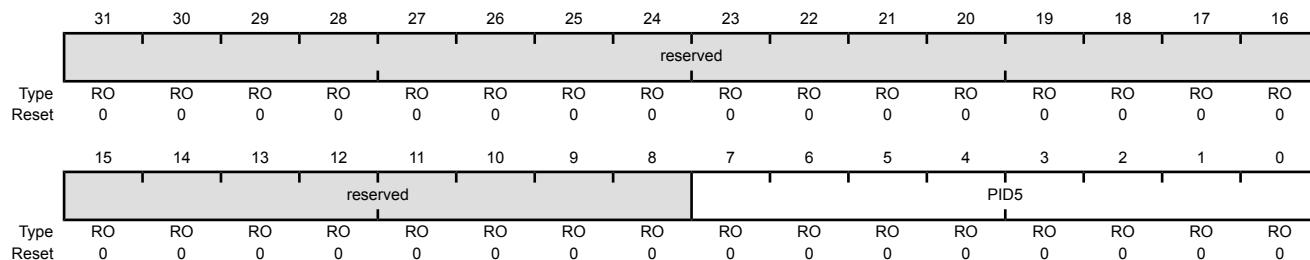
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	WDT Peripheral ID Register[7:0]

## 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

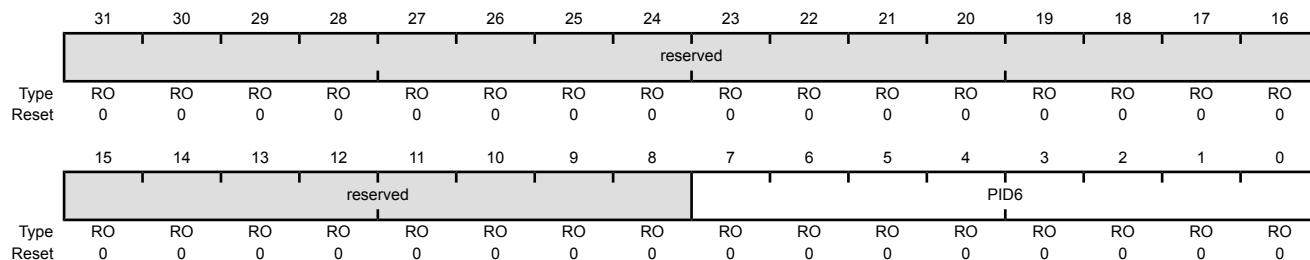
The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### Watchdog Peripheral Identification 6 (WDTPeriphID6)

Base 0x4000.0000

Offset 0xFD8

Type RO, reset 0x0000.0000



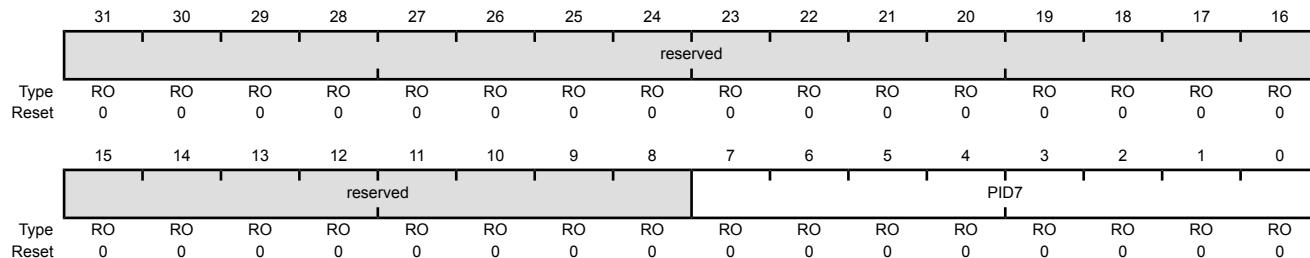
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	WDT Peripheral ID Register[23:16]

## 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



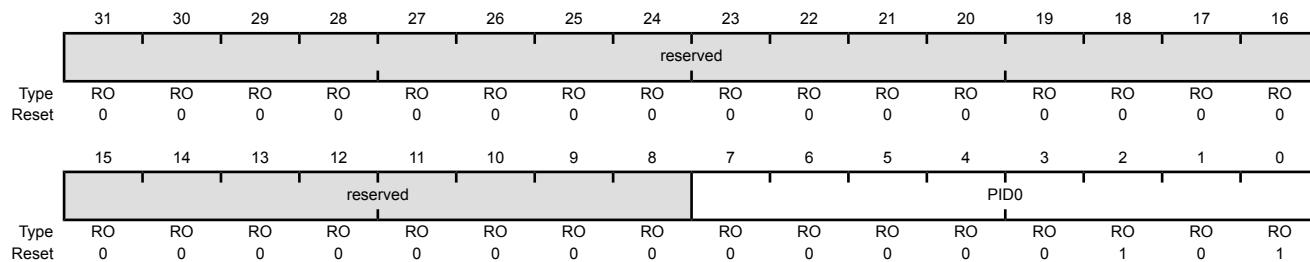
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	WDT Peripheral ID Register[31:24]

## 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



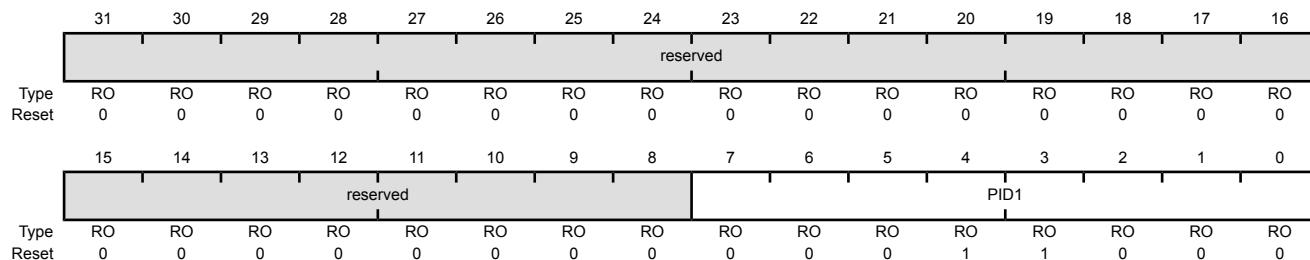
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x05	Watchdog Peripheral ID Register[7:0]

## 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



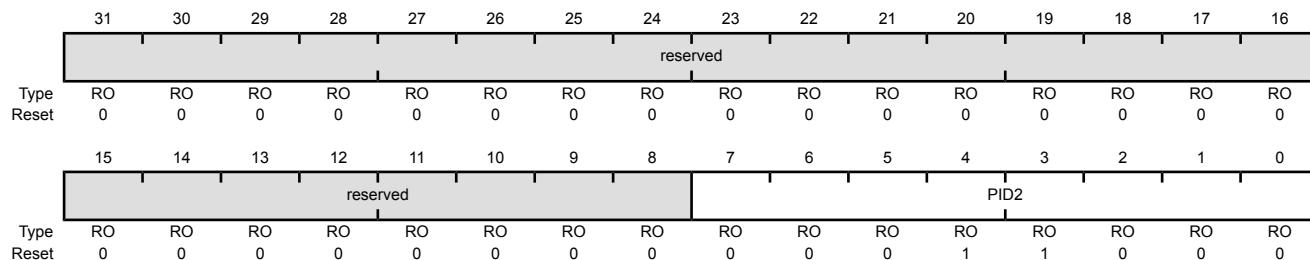
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x18	Watchdog Peripheral ID Register[15:8]

## 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



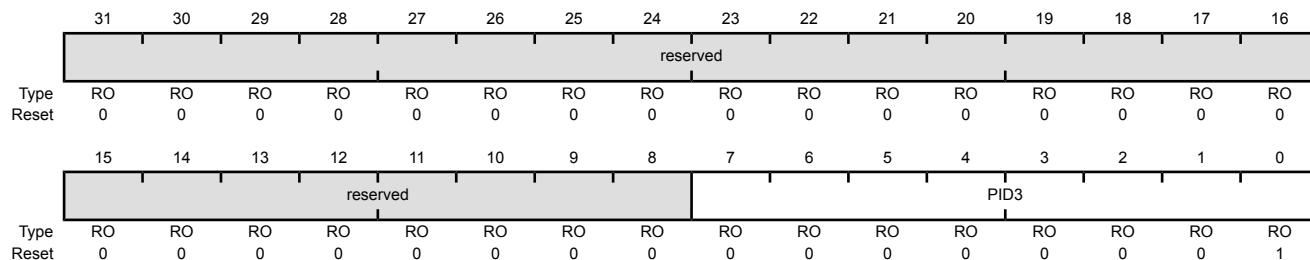
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	Watchdog Peripheral ID Register[23:16]

## 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



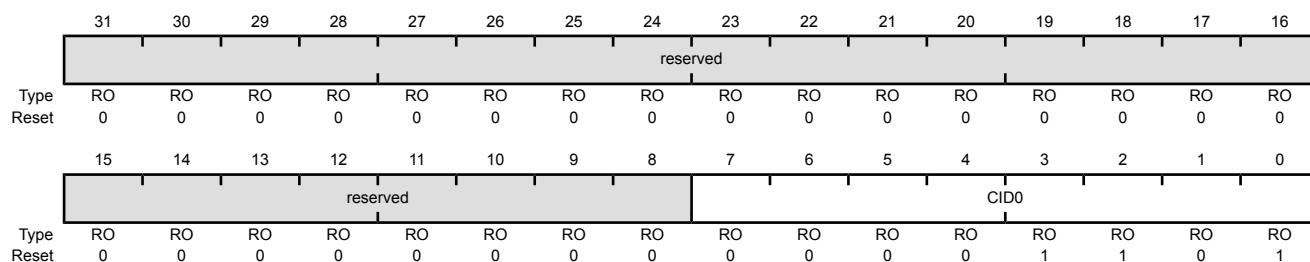
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	Watchdog Peripheral ID Register[31:24]

**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 (WDT\_PCellID0)

Base 0x4000.0000  
Offset 0xFF0  
Type RO, reset 0x0000.000D



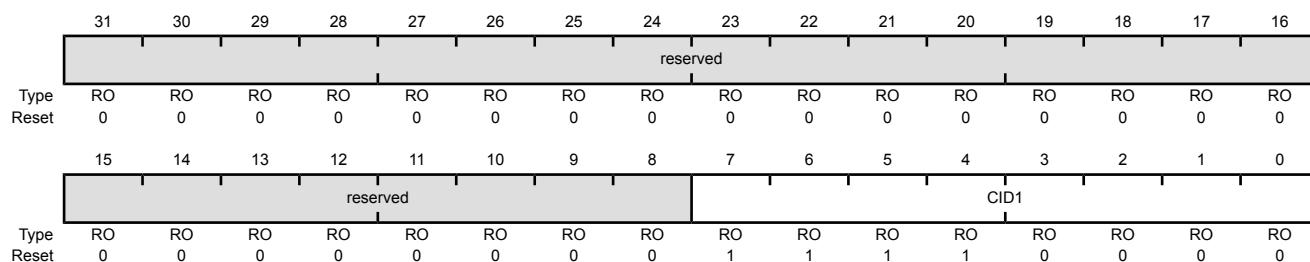
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	Watchdog PrimeCell ID Register[7:0]

## 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



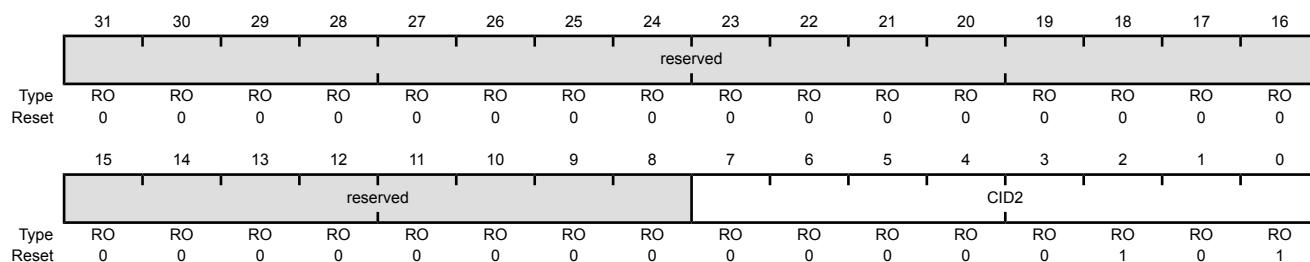
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	Watchdog PrimeCell ID Register[15:8]

## 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



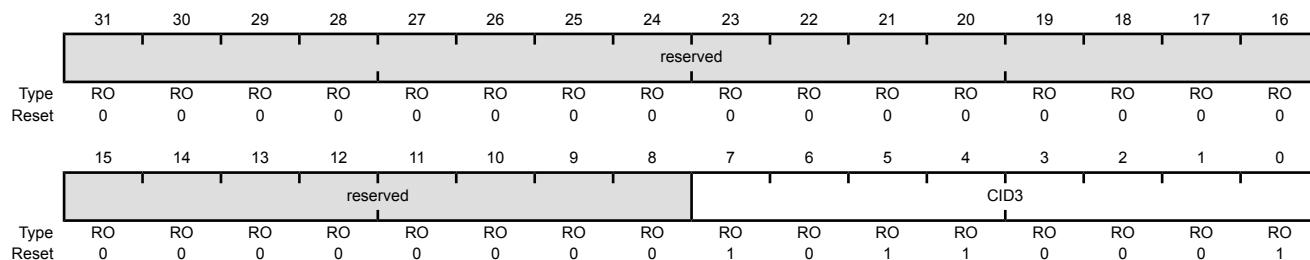
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	Watchdog PrimeCell ID Register[23:16]

## 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]

## 10 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.

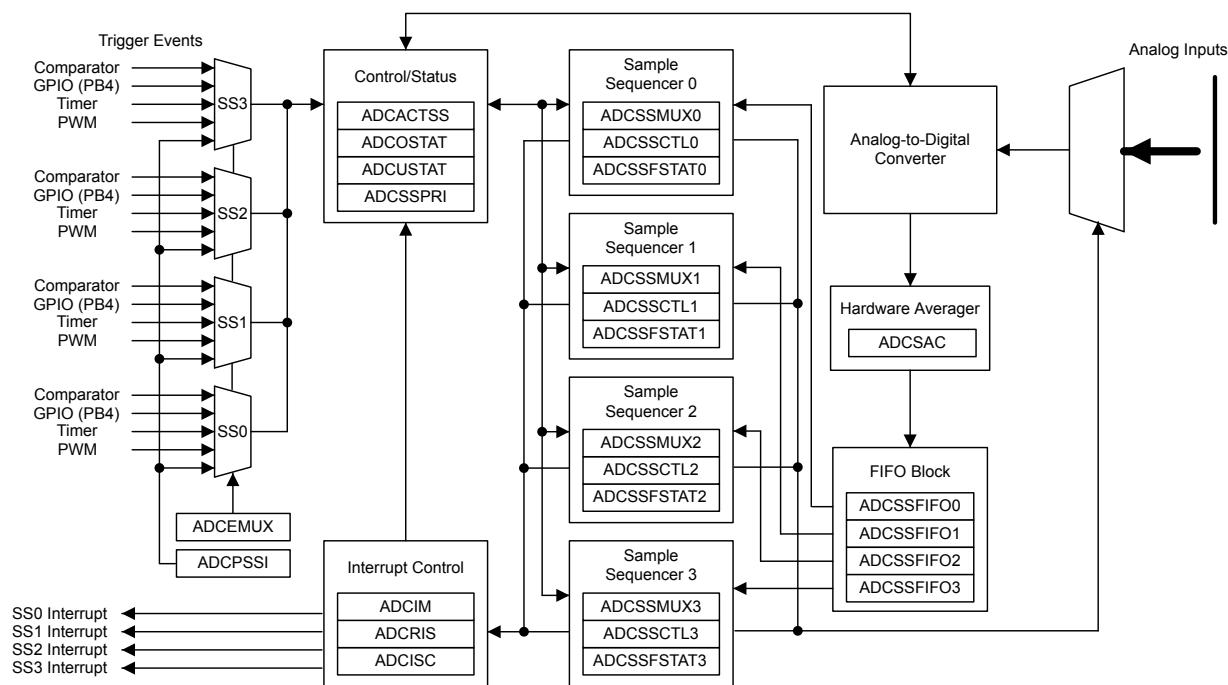
The Stellaris® ADC module features 10-bit conversion resolution and supports four input channels, plus an internal temperature sensor. The ADC module contains four programmable sequencer which allows for the sampling of multiple analog input sources without controller intervention. Each sample sequence provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequence priority.

The Stellaris ADC module provides the following features:

- Four analog input channels
- Single-ended and differential-input configurations
- On-chip internal temperature sensor
- 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
- Flexible trigger control
  - Controller (software)
  - Timers
  - Analog Comparators
  - PWM
  - GPIO
- Hardware averaging of up to 64 samples for improved accuracy
- Converter uses an internal 3-V reference

### 10.1 Block Diagram

Figure 10-1 on page 337 provides details on the internal configuration of the ADC controls and data registers.

**Figure 10-1. ADC Module Block Diagram**

## 10.2 Signal Description

The signals are analog functions for some GPIO signals. The column in the table below titled "Pin Assignment" lists the GPIO pin placement for the ADC signals. The  $A_{INx}$  analog signals are not 5-V tolerant and go through an isolation circuit before reaching their circuitry. These signals are configured by clearing the corresponding DEN bit in the **GPIO Digital Enable (GPIODEN)** register. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 232.

**Table 10-1. ADC Signals (48QFP)**

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
ADC0	1	I	Analog	Analog-to-digital converter input 0.
ADC1	2	I	Analog	Analog-to-digital converter input 1.
ADC2	3	I	Analog	Analog-to-digital converter input 2.
ADC3	4	I	Analog	Analog-to-digital converter input 3.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

**Table 10-2. ADC Signals (48QFN)**

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
ADC0	1	I	Analog	Analog-to-digital converter input 0.
ADC1	2	I	Analog	Analog-to-digital converter input 1.
ADC2	3	I	Analog	Analog-to-digital converter input 2.
ADC3	4	I	Analog	Analog-to-digital converter input 3.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

## 10.3 Functional Description

The Stellaris ADC collects sample data by using a programmable sequence-based approach instead of the traditional single or double-sampling approaches 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.

### 10.3.1 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 10-3 on page 338 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 10-3. 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, and should 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 **IEn** 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 (ADCSSFIFOOn)** 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 **ADCOSTAT** and **ADCUSTAT** registers.

### 10.3.2 Module Control

Outside of the sample sequencers, the remainder of the control logic is responsible for tasks such as:

- Interrupt generation
- Sequence prioritization
- 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 devices.

#### 10.3.2.1 Interrupts

The register configurations of the sample sequencers dictate which events generate raw interrupts, but do not have control over whether the interrupt is actually sent to the interrupt controller. The ADC module's interrupt signals are 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 the various interrupt signals, and the **ADC Interrupt Status and Clear (ADCISC)** register, which shows active interrupts that are enabled by the **ADCIM** register. Sequencer interrupts are cleared by writing a 1 to the corresponding IN bit in **ADCISC**.

#### 10.3.2.2 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.

#### 10.3.2.3 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 family member, but all devices share the "Controller" and "Always" triggers. Software can initiate sampling by setting the SS<sub>x</sub> bits in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register.

Care must be taken when using the "Always" trigger. If a sequence's priority is too high, it is possible to starve other lower priority sequences.

#### 10.3.3 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 359). There is a single averaging circuit and all input channels receive the same amount of averaging whether they are single-ended or differential.

#### 10.3.4 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.

### 10.3.5 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 `Dn` bit in the **ADCSSCTL0n** register in a step's configuration nibble.

When a sequence step is configured for differential sampling, its corresponding value in the **ADCSSMUXn** 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 10-4 on page 340). 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 10-4 on page 340).

**Table 10-4. Differential Sampling Pairs**

Differential Pair	Analog Inputs
0	0 and 1
1	2 and 3

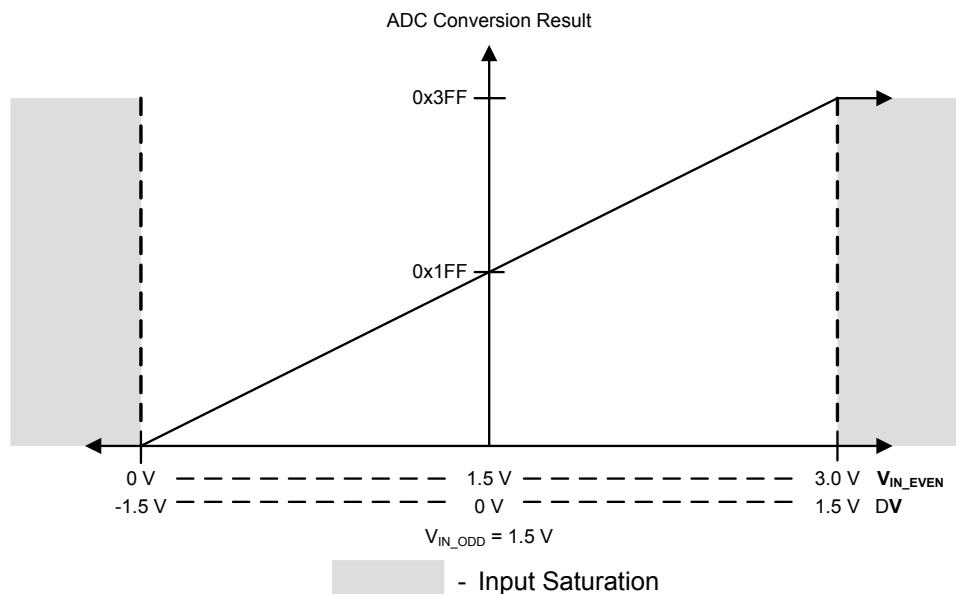
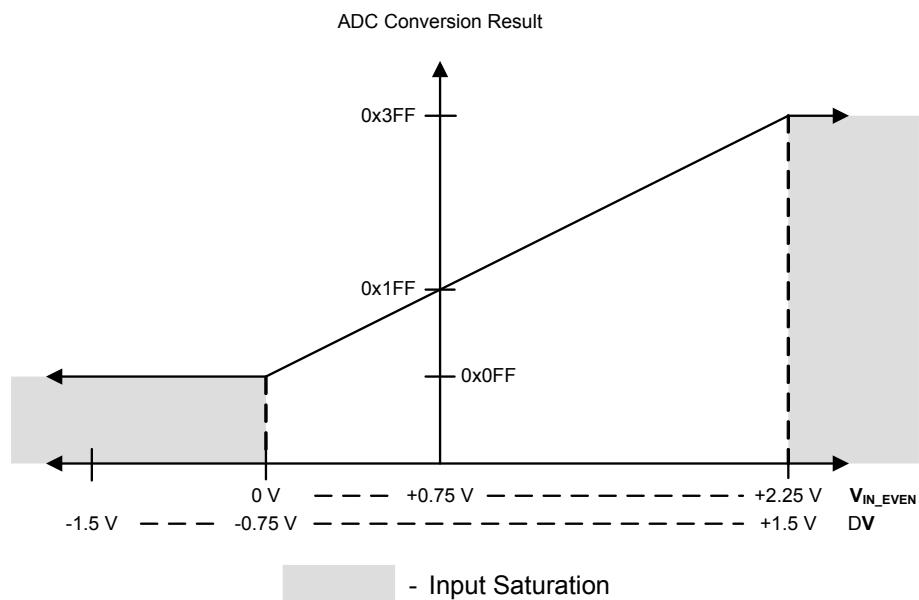
The voltage sampled in differential mode is the difference between the odd and even channels:

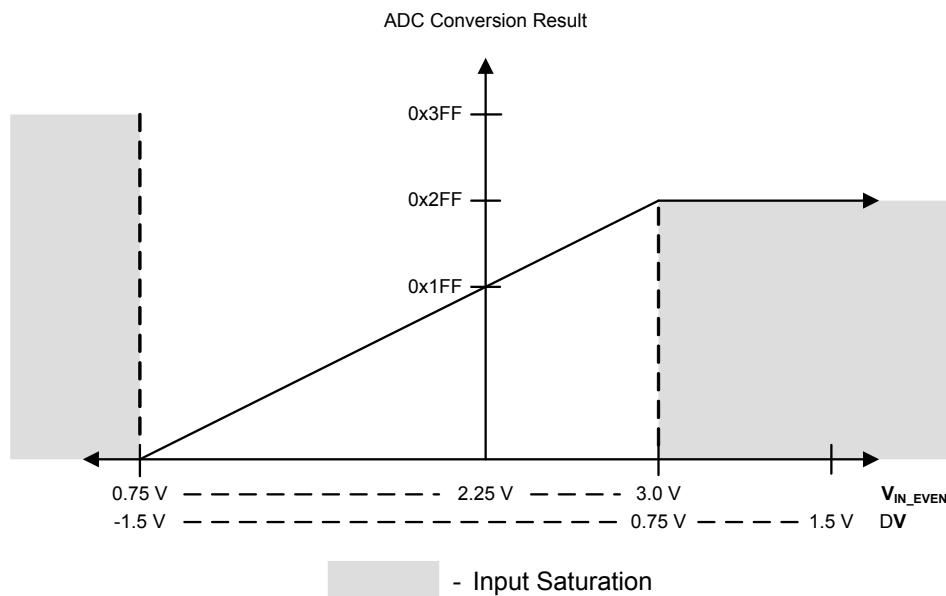
$$\Delta V \text{ (differential voltage)} = V_{IN\_EVEN} \text{ (even channels)} - V_{IN\_ODD} \text{ (odd channels)}, \text{ therefore:}$$

- 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 10-2 on page 341 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 10-3 on page 341 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 10-4 on page 342 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 10-2. Differential Sampling Range,  $V_{IN\_ODD} = 1.5\text{ V}$** **Figure 10-3. Differential Sampling Range,  $V_{IN\_ODD} = 0.75\text{ V}$** 

**Figure 10-4. Differential Sampling Range,  $V_{IN\_ODD} = 2.25\text{ V}$** 

### 10.3.6 Test Modes

There is a user-available test mode that allows for loopback operation within the digital portion of the ADC module. This can be useful for debugging software without having to provide actual analog stimulus. This mode is available through the **ADC Test Mode Loopback (ADCTMLB)** register (see page 372).

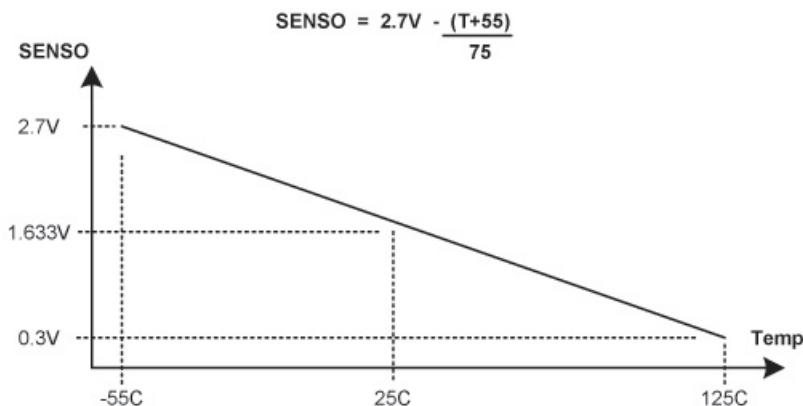
### 10.3.7 Internal Temperature Sensor

The temperature sensor does not have a separate enable, since it also contains the bandgap reference and must always be enabled. The reference is supplied to other analog modules; not just the ADC.

The internal temperature sensor provides an analog temperature reading as well as a reference voltage. The voltage at the output terminal SENSO is given by the following equation:

$$SENDO = 2.7 - ((T + 55) / 75)$$

This relation is shown in Figure 10-5 on page 343.

**Figure 10-5. Internal Temperature Sensor Characteristic**

## 10.4 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.

### 10.4.1 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 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 **RCGC0** register (see page 193).
2. 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.

### 10.4.2 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 **ASENn** 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 `ASENn` bit in the **ADCACTSS** register.

## 10.5 Register Map

Table 10-5 on page 344 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.

Note that the ADC module clock must be enabled before the registers can be programmed (see page 193). There must be a delay of 3 system clocks after the ADC module clock is enabled before any ADC module registers are accessed.

**Table 10-5. ADC Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	ADCACTSS	R/W	0x0000.0000	ADC Active Sample Sequencer	346
0x004	ADCRIS	RO	0x0000.0000	ADC Raw Interrupt Status	347
0x008	ADCIM	R/W	0x0000.0000	ADC Interrupt Mask	348
0x00C	ADCISC	R/W1C	0x0000.0000	ADC Interrupt Status and Clear	349
0x010	ADCOSTAT	R/W1C	0x0000.0000	ADC Overflow Status	350
0x014	ADCEMUX	R/W	0x0000.0000	ADC Event Multiplexer Select	351
0x018	ADCUSTAT	R/W1C	0x0000.0000	ADC Underflow Status	355
0x020	ADCSPPRI	R/W	0x0000.3210	ADC Sample Sequencer Priority	356
0x028	ADCPSSI	WO	-	ADC Processor Sample Sequence Initiate	358
0x030	ADCSAC	R/W	0x0000.0000	ADC Sample Averaging Control	359
0x040	ADCSSMUX0	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 0	360
0x044	ADCSSCTL0	R/W	0x0000.0000	ADC Sample Sequence Control 0	362
0x048	ADCSSFIFO0	RO	-	ADC Sample Sequence Result FIFO 0	365
0x04C	ADCSSFSTAT0	RO	0x0000.0100	ADC Sample Sequence FIFO 0 Status	366
0x060	ADCSSMUX1	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 1	367
0x064	ADCSSCTL1	R/W	0x0000.0000	ADC Sample Sequence Control 1	368
0x068	ADCSSFIFO1	RO	-	ADC Sample Sequence Result FIFO 1	365
0x06C	ADCSSFSTAT1	RO	0x0000.0100	ADC Sample Sequence FIFO 1 Status	366
0x080	ADCSSMUX2	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 2	367
0x084	ADCSSCTL2	R/W	0x0000.0000	ADC Sample Sequence Control 2	368
0x088	ADCSSFIFO2	RO	-	ADC Sample Sequence Result FIFO 2	365

**Table 10-5. ADC Register Map (*continued*)**

Offset	Name	Type	Reset	Description	See page
0x08C	ADCSSFSTAT2	RO	0x0000.0100	ADC Sample Sequence FIFO 2 Status	366
0x0A0	ADCSSMUX3	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 3	370
0x0A4	ADCSSCTL3	R/W	0x0000.0002	ADC Sample Sequence Control 3	371
0x0A8	ADCSSFIFO3	RO	-	ADC Sample Sequence Result FIFO 3	365
0x0AC	ADCSSFSTAT3	RO	0x0000.0100	ADC Sample Sequence FIFO 3 Status	366
0x100	ADCTMLB	R/W	0x0000.0000	ADC Test Mode Loopback	372

## 10.6 Register Descriptions

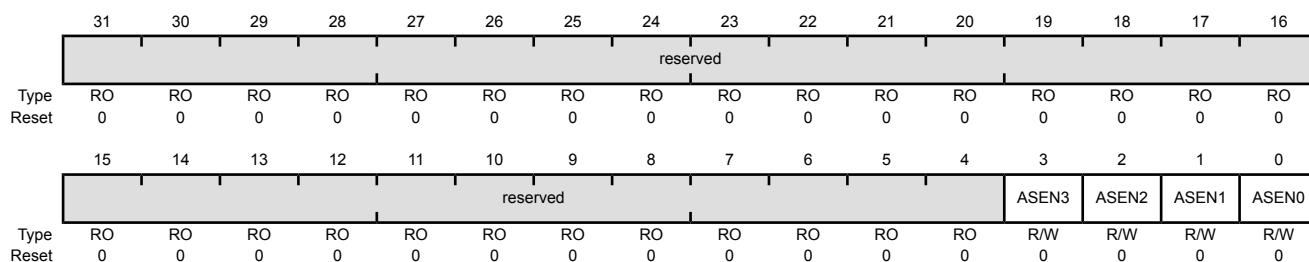
The remainder of this section lists and describes the ADC registers, in numerical order by address offset.

## Register 1: ADC Active Sample Sequencer (ADCACTSS), offset 0x000

This register controls the activation of the sample sequencers. Each sample sequencer can be enabled or disabled independently.

### ADC Active Sample Sequencer (ADCACTSS)

Base 0x4003.8000  
Offset 0x000  
Type R/W, reset 0x0000.0000



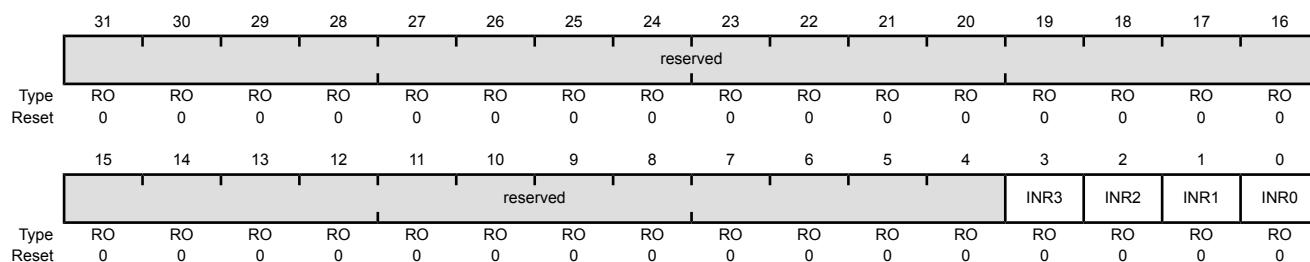
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
3	ASEN3	R/W	0	ADC SS3 Enable Specifies whether Sample Sequencer 3 is enabled. If set, the sample sequence logic for Sequencer 3 is active. Otherwise, the sequencer is inactive.										
2	ASEN2	R/W	0	ADC SS2 Enable Specifies whether Sample Sequencer 2 is enabled. If set, the sample sequence logic for Sequencer 2 is active. Otherwise, the sequencer is inactive.										
1	ASEN1	R/W	0	ADC SS1 Enable Specifies whether Sample Sequencer 1 is enabled. If set, the sample sequence logic for Sequencer 1 is active. Otherwise, the sequencer is inactive.										
0	ASEN0	R/W	0	ADC SS0 Enable Specifies whether Sample Sequencer 0 is enabled. If set, the sample sequence logic for Sequencer 0 is active. Otherwise, the sequencer is inactive.										

## 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



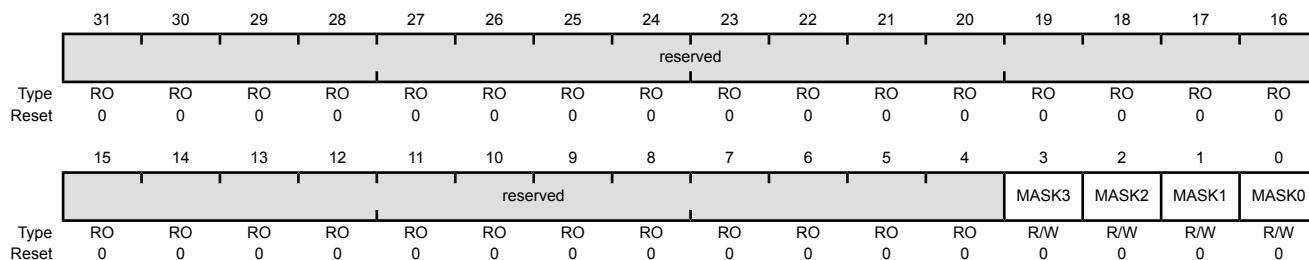
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	INR3	RO	0	SS3 Raw Interrupt Status This bit is set by hardware when a sample with its respective <b>ADCSSCTL3 IE</b> bit has completed conversion. This bit is cleared by setting the <b>IN3</b> bit in the <b>ADCISC</b> register.
2	INR2	RO	0	SS2 Raw Interrupt Status This bit is set by hardware when a sample with its respective <b>ADCSSCTL2 IE</b> bit has completed conversion. This bit is cleared by setting the <b>IN2</b> bit in the <b>ADCISC</b> register.
1	INR1	RO	0	SS1 Raw Interrupt Status This bit is set by hardware when a sample with its respective <b>ADCSSCTL1 IE</b> bit has completed conversion. This bit is cleared by setting the <b>IN1</b> bit in the <b>ADCISC</b> register.
0	INR0	RO	0	SS0 Raw Interrupt Status This bit is set by hardware when a sample with its respective <b>ADCSSCTL0 IE</b> bit has completed conversion. This bit is cleared by setting the <b>IN30</b> bit in the <b>ADCISC</b> register.

## Register 3: ADC Interrupt Mask (ADCIM), offset 0x008

This register controls whether the sample sequencer raw interrupt signals are promoted to controller interrupts. Each raw interrupt signal can be masked independently.

### ADC Interrupt Mask (ADCIM)

Base 0x4003.8000  
Offset 0x008  
Type R/W, reset 0x0000.0000



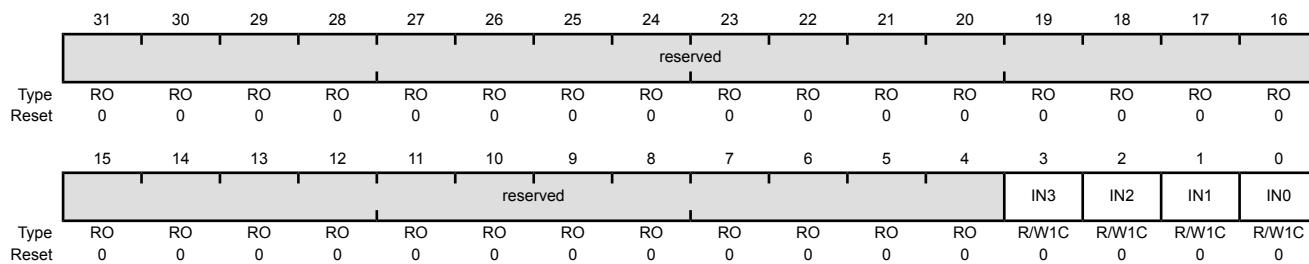
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	MASK3	R/W	0	<p>SS3 Interrupt Mask</p> <p>When set, this bit allows the raw interrupt signal from Sample Sequencer 3 (<b>ADCRIS</b> register <b>INR3</b> bit) to be promoted to a controller interrupt.</p> <p>When clear, the status of Sample Sequencer 3 does not affect the SS3 interrupt status.</p>
2	MASK2	R/W	0	<p>SS2 Interrupt Mask</p> <p>When set, this bit allows the raw interrupt signal from Sample Sequencer 2 (<b>ADCRIS</b> register <b>INR2</b> bit) to be promoted to a controller interrupt.</p> <p>When clear, the status of Sample Sequencer 2 does not affect the SS2 interrupt status.</p>
1	MASK1	R/W	0	<p>SS1 Interrupt Mask</p> <p>When set, this bit allows the raw interrupt signal from Sample Sequencer 1 (<b>ADCRIS</b> register <b>INR1</b> bit) to be promoted to a controller interrupt.</p> <p>When clear, the status of Sample Sequencer 1 does not affect the SS1 interrupt status.</p>
0	MASK0	R/W	0	<p>SS0 Interrupt Mask</p> <p>When set, this bit allows the raw interrupt signal from Sample Sequencer 0 (<b>ADCRIS</b> register <b>INR0</b> bit) to be promoted to a controller interrupt.</p> <p>When clear, the status of Sample Sequencer 0 does not affect the SS0 interrupt status.</p>

## Register 4: ADC Interrupt Status and Clear (ADCISC), offset 0x00C

This register provides the mechanism for clearing sample sequence 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. Sample sequence nterrupts are cleared by setting the corresponding bit position. If software is polling the **ADCRIS** instead of generating interrupts, the sample sequence `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



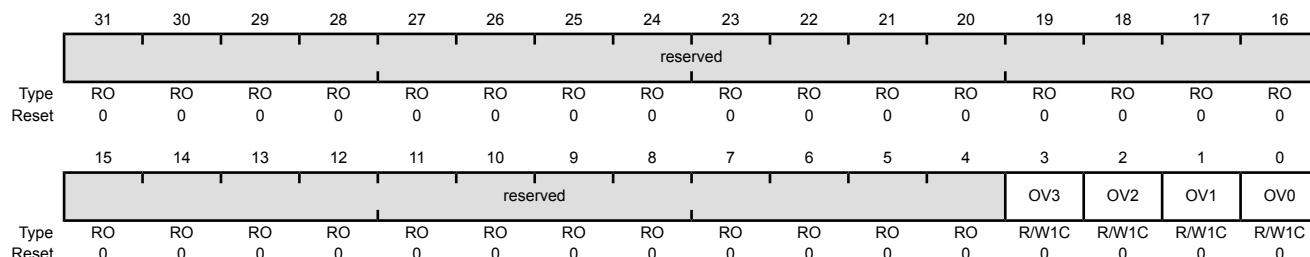
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	IN3	R/W1C	0	<p>SS3 Interrupt Status and Clear</p> <p>This bit is set when both the <code>INR3</code> bit in the <b>ADCRIS</b> register and the <code>MASK3</code> bit in the <b>ADCIM</b> register are set, providing a level-based interrupt to the controller.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the <code>INR3</code> bit.</p>
2	IN2	R/W1C	0	<p>SS2 Interrupt Status and Clear</p> <p>This bit is set when both the <code>INR2</code> bit in the <b>ADCRIS</b> register and the <code>MASK2</code> bit in the <b>ADCIM</b> register are set, providing a level-based interrupt to the controller.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the <code>INR2</code> bit.</p>
1	IN1	R/W1C	0	<p>SS1 Interrupt Status and Clear</p> <p>This bit is set when both the <code>INR1</code> bit in the <b>ADCRIS</b> register and the <code>MASK1</code> bit in the <b>ADCIM</b> register are set, providing a level-based interrupt to the controller.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the <code>INR1</code> bit.</p>
0	IN0	R/W1C	0	<p>SS0 Interrupt Status and Clear</p> <p>This bit is set when both the <code>INR0</code> bit in the <b>ADCRIS</b> register and the <code>MASK0</code> bit in the <b>ADCIM</b> register are set, providing a level-based interrupt to the controller.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the <code>INR0</code> bit.</p>

## 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



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.0000	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	OV3	R/W1C	0	SS3 FIFO Overflow  When set, 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.  This bit is cleared by writing a 1.
2	OV2	R/W1C	0	SS2 FIFO Overflow  When set, 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.  This bit is cleared by writing a 1.
1	OV1	R/W1C	0	SS1 FIFO Overflow  When set, 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.  This bit is cleared by writing a 1.
0	OV0	R/W1C	0	SS0 FIFO Overflow  When set, 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.  This bit is cleared by writing a 1.

## 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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	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
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
EM3				EM2				EM1				EM0				

Bit/Field	Name	Type	Reset	Description
31:16	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.
15:12	EM3	R/W	0x0	SS3 Trigger Select  This field selects the trigger source for Sample Sequencer 3.  The valid configurations for this field are:
				Value      Event
				0x0      Controller (default)
				0x1      Analog Comparator 0
				0x2      Reserved
				0x3      Reserved
				0x4      External (GPIO PB4)
				0x5      Timer
				In addition, the trigger must be enabled with the <b>TNOTE</b> bit in the <b>GPTMCTL</b> register (see page 293).
				0x6      PWM0
				The PWM module 0 trigger can be configured with the <b>PWM0 Interrupt and Trigger Enable (PWM0INTEN)</b> register, see page 521.
				0x7      PWM1
				The PWM module 1 trigger can be configured with the <b>PWM1INTEN</b> register, see page 521.
				0x8      PWM2
				The PWM module 2 trigger can be configured with the <b>PWM2INTEN</b> register, see page 521.
				0x9-0xE reserved
				0xF      Always (continuously sample)

Bit/Field	Name	Type	Reset	Description
11:8	EM2	R/W	0x0	SS2 Trigger Select This field selects the trigger source for Sample Sequencer 2. The valid configurations for this field are:
				Value      Event
				0x0      Controller (default)
				0x1      Analog Comparator 0
				0x2      Reserved
				0x3      Reserved
				0x4      External (GPIO PB4)
				0x5      Timer In addition, the trigger must be enabled with the TNOTE bit in the <b>GPTMCTL</b> register (see page 293).
				0x6      PWM0 The PWM module 0 trigger can be configured with the <b>PWM0 Interrupt and Trigger Enable (PWM0INTEN)</b> register, see page 521.
				0x7      PWM1 The PWM module 1 trigger can be configured with the <b>PWM1INTEN</b> register, see page 521.
				0x8      PWM2 The PWM module 2 trigger can be configured with the <b>PWM2INTEN</b> register, see page 521.
				0x9-0xE reserved
				0xF      Always (continuously sample)

Bit/Field	Name	Type	Reset	Description																								
7:4	EM1	R/W	0x0	<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> <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>Analog Comparator 0</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>PWM0</td></tr> <tr> <td>0x7</td><td>PWM1</td></tr> <tr> <td>0x8</td><td>PWM2</td></tr> <tr> <td>0x9-0xE</td><td>reserved</td></tr> <tr> <td>0xF</td><td>Always (continuously sample)</td></tr> </tbody> </table> <p>In addition, the trigger must be enabled with the TNOTE bit in the <b>GPTMCTL</b> register (see page 293).</p>	Value	Event	0x0	Controller (default)	0x1	Analog Comparator 0	0x2	Reserved	0x3	Reserved	0x4	External (GPIO PB4)	0x5	Timer	0x6	PWM0	0x7	PWM1	0x8	PWM2	0x9-0xE	reserved	0xF	Always (continuously sample)
Value	Event																											
0x0	Controller (default)																											
0x1	Analog Comparator 0																											
0x2	Reserved																											
0x3	Reserved																											
0x4	External (GPIO PB4)																											
0x5	Timer																											
0x6	PWM0																											
0x7	PWM1																											
0x8	PWM2																											
0x9-0xE	reserved																											
0xF	Always (continuously sample)																											

Bit/Field	Name	Type	Reset	Description
3:0	EM0	R/W	0x0	SS0 Trigger Select This field selects the trigger source for Sample Sequencer 0. The valid configurations for this field are:
				Value      Event
				0x0      Controller (default)
				0x1      Analog Comparator 0
				0x2      Reserved
				0x3      Reserved
				0x4      External (GPIO PB4)
				0x5      Timer In addition, the trigger must be enabled with the TNOTE bit in the <b>GPTMCTL</b> register (see page 293).
				0x6      PWM0 The PWM module 0 trigger can be configured with the <b>PWM0 Interrupt and Trigger Enable (PWM0INTEN)</b> register, see page 521.
				0x7      PWM1 The PWM module 1 trigger can be configured with the <b>PWM1INTEN</b> register, see page 521.
				0x8      PWM2 The PWM module 2 trigger can be configured with the <b>PWM2INTEN</b> register, see page 521.
				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 is 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															
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

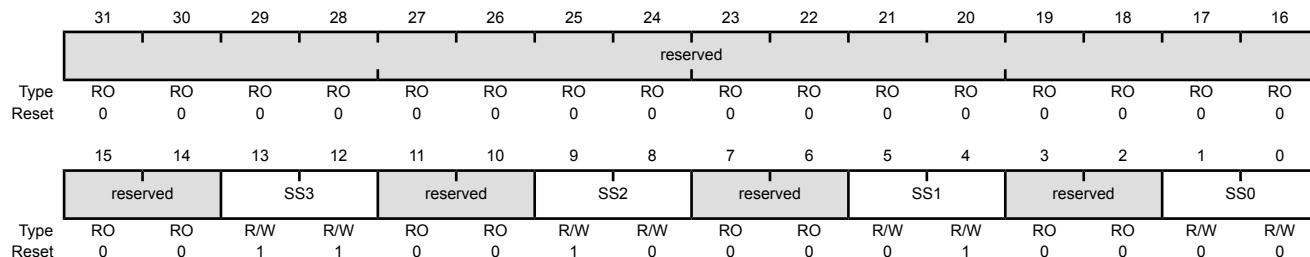
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	UV3	R/W1C	0	<p>SS3 FIFO Underflow</p> <p>When set, 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.</p> <p>This bit is cleared by writing a 1.</p>
2	UV2	R/W1C	0	<p>SS2 FIFO Underflow</p> <p>When set, 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.</p> <p>This bit is cleared by writing a 1.</p>
1	UV1	R/W1C	0	<p>SS1 FIFO Underflow</p> <p>When set, 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.</p> <p>This bit is cleared by writing a 1.</p>
0	UV0	R/W1C	0	<p>SS0 FIFO Underflow</p> <p>When set, 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.</p> <p>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 Sequencer 3 has the lowest priority. When reconfiguring sequence priorities, each sequence must have a unique priority for the ADC to operate properly.

### 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	0x0000.0	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.
13:12	SS3	R/W	0x3	SS3 Priority This 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. The ADC may not operate properly 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 This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 2. A priority encoding of 0 is highest and 3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
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 This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 1. A priority encoding of 0 is highest and 3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
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.

Bit/Field	Name	Type	Reset	Description
1:0	SS0	R/W	0x0	<p>SS0 Priority</p> <p>This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 0. A priority encoding of 0 is highest and 3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.</p>

## 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	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	WO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-

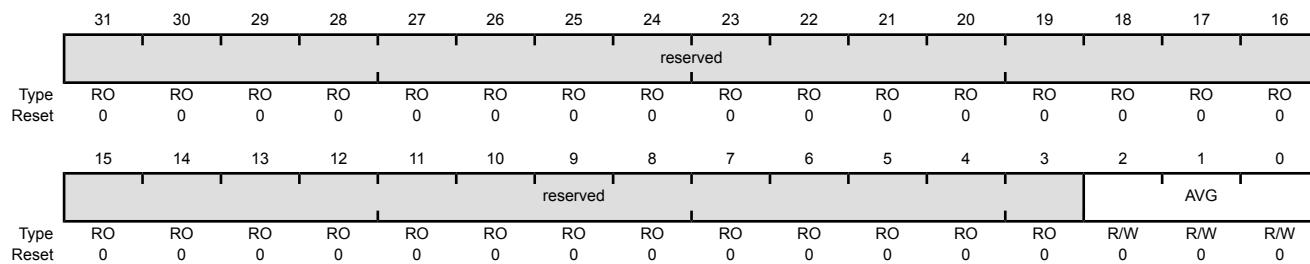
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	SS3	WO	-	SS3 Initiate When set, this bit triggers sampling on Sample Sequencer 3 if the sequencer is enabled in the <b>ADCACTSS</b> register. Only a write by software is valid; a read of this register returns no meaningful data.
2	SS2	WO	-	SS2 Initiate When set, this bit triggers sampling on Sample Sequencer 2 if the sequencer is enabled in the <b>ADCACTSS</b> register. Only a write by software is valid; a read of this register returns no meaningful data.
1	SS1	WO	-	SS1 Initiate When set, this bit triggers sampling on Sample Sequencer 1 if the sequencer is enabled in the <b>ADCACTSS</b> register. Only a write by software is valid; a read of this register returns no meaningful data.
0	SS0	WO	-	SS0 Initiate When set, this bit triggers sampling on Sample Sequencer 0 if the sequencer is enabled in the <b>ADCACTSS</b> register. Only a write by software is valid; a read of this register returns no meaningful data.

## 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  $2^{\text{AVG}}$  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	0x0000.0000	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.
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
Type	reserved	MUX7	reserved	MUX6	reserved	MUX5	reserved	MUX4								
Reset	RO 0	RO 0	R/W 0	R/W 0	RO 0	RO 0	R/W 0	R/W 0	RO 0	RO 0	R/W 0	R/W 0	RO 0	RO 0	R/W 0	R/W 0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	reserved	MUX3	reserved	MUX2	reserved	MUX1	reserved	MUX0								
Reset	RO 0	RO 0	R/W 0	R/W 0	RO 0	RO 0	R/W 0	R/W 0	RO 0	RO 0	R/W 0	R/W 0	RO 0	RO 0	R/W 0	R/W 0

Bit/Field	Name	Type	Reset	Description
31:30	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.
29:28	MUX7	R/W	0x0	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.
27:26	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.
25:24	MUX6	R/W	0x0	7th Sample Input Select The MUX6 field is used during the seventh sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
23:22	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.
21:20	MUX5	R/W	0x0	6th Sample Input Select The MUX5 field is used during the sixth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
19:18	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
17:16	MUX4	R/W	0x0	5th Sample Input Select The MUX4 field is used during the fifth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
15:14	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.
13:12	MUX3	R/W	0x0	4th Sample Input Select The MUX3 field is used during the fourth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
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	MUX2	R/W	0x0	3rd Sample Input Select The MUX72 field is used during the third sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
7:6	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.
5:4	MUX1	R/W	0x0	2nd Sample Input Select The MUX1 field is used during the second sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
3:2	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.
1:0	MUX0	R/W	0x0	1st Sample Input Select The MUX0 field is used during the first sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.

## Register 12: ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044

This register contains the configuration information for each sample for a sequence executed with a sample sequencer. 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 (ADCSSCTL0)

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
Type	TS7	IE7	END7	D7	TS6	IE6	END6	D6	TS5	IE5	END5	D5	TS4	IE4	END4	D4
Reset	R/W 0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Reset	R/W 0															

Bit/Field	Name	Type	Reset	Description
31	TS7	R/W	0	8th Sample Temp Sensor Select  This bit is used during the eighth sample of the sample sequence and specifies the input source of the sample.  When set, the temperature sensor is read.  When clear, the input pin specified by the <b>ADCSSMUX</b> register is read.
30	IE7	R/W	0	8th Sample Interrupt Enable  This bit is used during the eighth sample of the sample sequence and specifies whether the raw interrupt signal (INR0 bit) is asserted at the end of the sample's conversion. If the MASK0 bit in the <b>ADCIM</b> register is set, the interrupt is promoted to a controller-level interrupt.  When this bit is set, the raw interrupt is asserted.  When this bit is clear, the raw interrupt is not asserted.  It is legal to have multiple samples within a sequence generate interrupts.
29	END7	R/W	0	8th Sample is End of Sequence  The END7 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 END are not requested for conversion even though the fields may be non-zero. It is required that software write the END bit somewhere within the sequence. (Sample Sequencer 3, which only has a single sample in the sequence, is hardwired to have the END0 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 D7 bit indicates that the analog input is to be differentially sampled. The corresponding <b>ADCSSMUXx</b> 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 TS7 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.
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.

Bit/Field	Name	Type	Reset	Description
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.
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 (ADCSS FIFO0), offset 0x048****Register 14: ADC Sample Sequence Result FIFO 1 (ADCSS FIFO1), offset 0x068****Register 15: ADC Sample Sequence Result FIFO 2 (ADCSS FIFO2), offset 0x088****Register 16: ADC Sample Sequence Result FIFO 3 (ADCSS FIFO3), offset 0x0A8**


---

**Important:** This register is read-sensitive. See the register description for details.

---

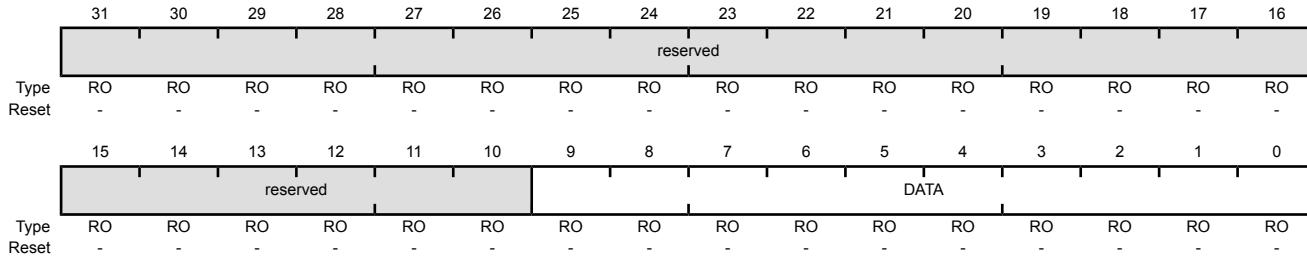
This register contains the conversion results for samples collected with the sample sequencer (the **ADCSS FIFO0** register is used for Sample Sequencer 0, **ADCSS FIFO1** for Sequencer 1, **ADCSS FIFO2** for Sequencer 2, and **ADCSS FIFO3** for Sequencer 3). Reads of this register return conversion 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 (ADCSS FIFO0)**

Base 0x4003.8000

Offset 0x048

Type RO, reset -



Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	-	Software should not rely on the value of 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	-	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**

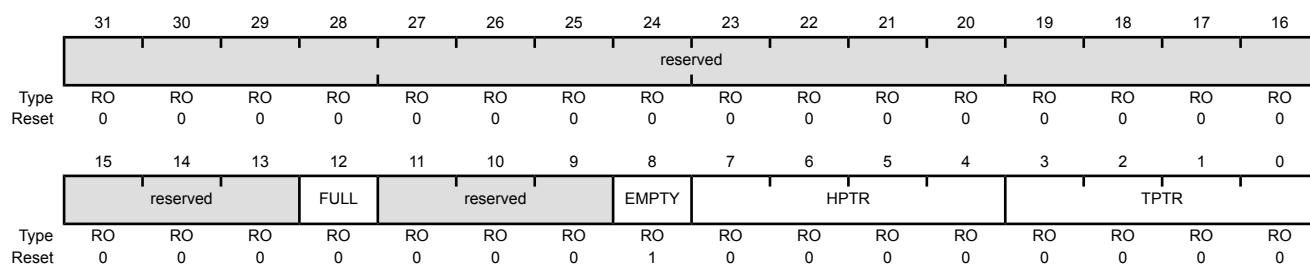
**Register 19: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C**

**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



Bit/Field	Name	Type	Reset	Description
31:13	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.
12	FULL	RO	0	FIFO Full When set, this bit indicates that the FIFO is currently full.
11:9	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.
8	EMPTY	RO	1	FIFO Empty When set, this bit indicates that the FIFO is currently empty.
7:4	HPTR	RO	0x0	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	0x0	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 360 for detailed bit descriptions. The **ADCSSMUX1** register affects Sample Sequencer 1 and the **ADCSSMUX2** register affects Sample Sequencer 2.

### ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1)

Base 0x4003.8000  
Offset 0x060  
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				MUX3		reserved		MUX2		reserved		MUX1		reserved		MUX0
Type	RO	RO	R/W	R/W	RO	RO	R/W	R/W	RO	RO	R/W	R/W	RO	RO	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	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.
13:12	MUX3	R/W	0x0	4th Sample Input Select
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	MUX2	R/W	0x0	3rd Sample Input Select
7:6	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.
5:4	MUX1	R/W	0x0	2nd Sample Input Select
3:2	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.
1:0	MUX0	R/W	0x0	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. These registers are 16-bits wide and contain information for four possible samples. See the **ADCSSCTL0** register on page 362 for detailed bit descriptions. The **ADCSSCTL1** register configures Sample Sequencer 1 and the **ADCSSCTL2** register configures Sample Sequencer 2.

## 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
Type	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Reset	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

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	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.
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.

Bit/Field	Name	Type	Reset	Description
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.
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

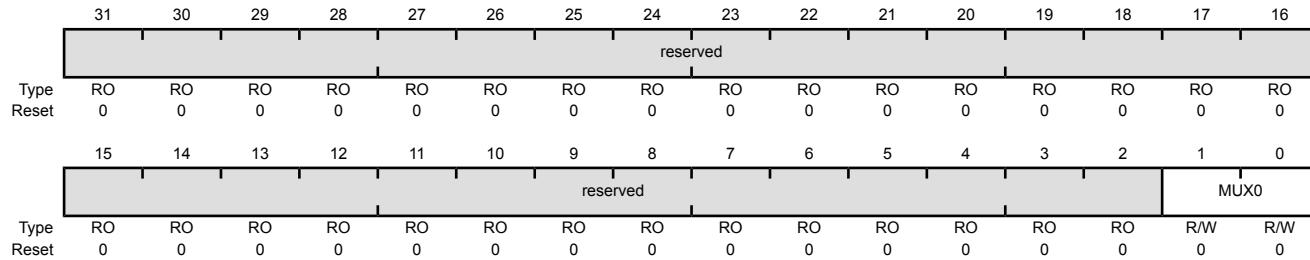
This register defines the analog input configuration for a sample executed with Sample Sequencer 3. This register is 4-bits wide and contains information for one possible sample. See the **ADCSSMUX0** register on page 360 for detailed bit descriptions.

### ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3)

Base 0x4003.8000

Offset 0x0A0

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.0000	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.
1:0	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 a sample 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 362 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															
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	1	0

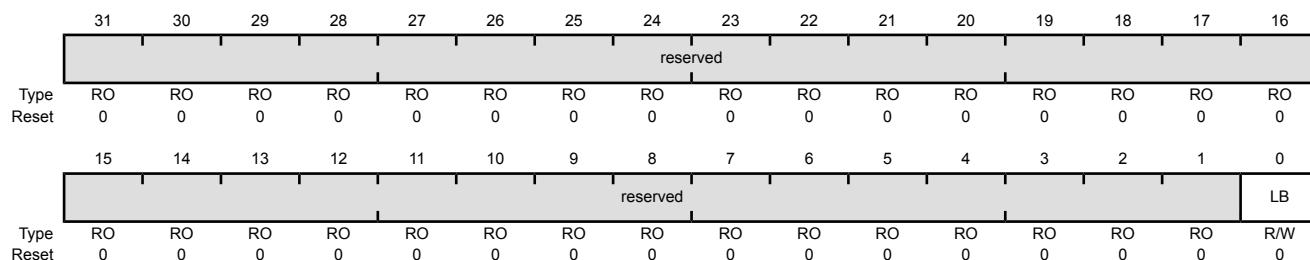
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same 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.

## Register 27: ADC Test Mode Loopback (ADCTMLB), offset 0x100

This register provides loopback operation within the digital logic of the ADC, which can be useful in debugging software without having to provide actual analog stimulus. This test mode is entered by writing a value of 0x0000.0001 to this register. When data is read from the FIFO in loopback mode, the read-only portion of this register is returned.

### ADC Test Mode Loopback (ADCTMLB)

Base 0x4003.8000  
Offset 0x100  
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of 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	LB	R/W	0	<p>Loopback Mode Enable</p> <p>When set, forces a loopback within the digital block to provide information on input and unique numbering. The <b>ADCSSFIFO</b> registers do not provide sample data, but instead provide the 10-bit loopback data as shown below.</p>
Bit/Field Name Description				
9:6	CNT	Continuous Sample Counter		Continuous sample counter that is initialized to 0 and counts each sample as it processed. This helps provide a unique value for the data received.
5	CONT	Continuation Sample Indicator		When set, indicates that this is a continuation sample. For example, if two sequencers were to run back-to-back, this indicates that the controller kept continuously sampling at full rate.
4	DIFF	Differential Sample Indicator		When set, indicates that this is a differential sample.
3	TS	Temp Sensor Sample Indicator		When set, indicates that this is a temperature sensor sample.
2:0	MUX	Analog Input Indicator		Indicates which analog input is to be sampled.

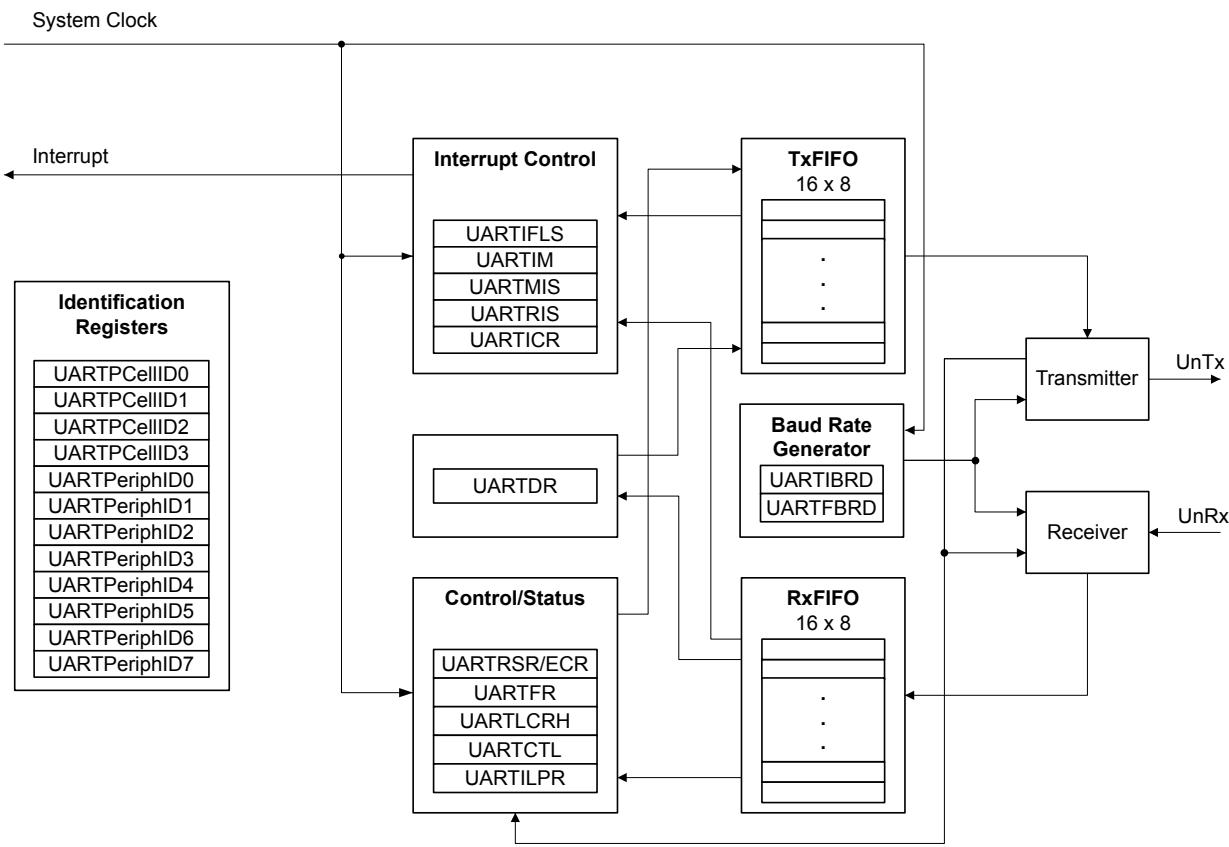
# 11 Universal Asynchronous Receivers/Transmitters (UARTs)

Each Stellaris® Universal Asynchronous Receiver/Transmitter (UART) has the following features:

- Two fully programmable 16C550-type UARTs
- Separate 16x8 transmit (TX) and 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
- 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

## 11.1 Block Diagram

Figure 11-1. UART Module Block Diagram



## 11.2 Signal Description

Table 11-1 on page 374 and Table 11-2 on page 375 list the external signals of the UART module and describe the function of each. The UART signals are alternate functions for some GPIO signals and default to be GPIO signals at reset, with the exception of the **U0Rx** and **U0Tx** pins which default to the UART function. The column in the table below titled "Pin Assignment" lists the possible GPIO pin placements for these UART signals. The **AFSEL** bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 253) should be set to choose the UART function. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 232.

Table 11-1. UART Signals (48QFP)

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
U0Rx	17	I	TTL	UART module 0 receive.
U0Tx	18	O	TTL	UART module 0 transmit.
U1Rx	27	I	TTL	UART module 1 receive.
U1Tx	28	O	TTL	UART module 1 transmit.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

**Table 11-2. UART Signals (48QFN)**

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
U0Rx	17	I	TTL	UART module 0 receive.
U0Tx	18	O	TTL	UART module 0 transmit.
U1Rx	27	I	TTL	UART module 1 receive.
U1Tx	28	O	TTL	UART module 1 transmit.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

## 11.3 Functional Description

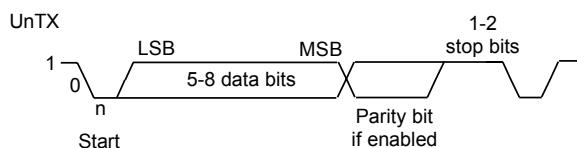
Each Stellaris UART performs the functions of parallel-to-serial and serial-to-parallel conversions. It is similar in functionality to a 16C550 UART, but is not register compatible.

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.

### 11.3.1 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 11-2 on page 375 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 11-2. UART Character Frame**

### 11.3.2 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 Divider (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.)

$$\text{BRD} = \text{BRDI} + \text{BRDF} = \text{UARTSysClk} / (16 * \text{Baud Rate})$$

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:

$$\text{UARTFBRD}[DIVFRAC] = \text{integer}(\text{BRDF} * 64 + 0.5)$$

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

### 11.3.3 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 385) 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 375).

The start bit is valid and recognized if UnRx is still low on the eighth cycle of Baud16, otherwise it is ignored. After a valid start bit is detected, 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.

### 11.3.4 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 381). 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 385) 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 1/8, 1/4, 1/2, 3/4, and 7/8. For example, if the 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 1/2 mark.

### 11.3.5 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 interrupt changes state when one of the following events occurs:

- If the FIFOs are enabled and the receive FIFO reaches the programmed trigger level, the **RXRIS** bit is set. The receive interrupt is cleared by reading data from the receive FIFO until it becomes less than the trigger level, or by clearing the interrupt by writing a 1 to the **RXIC** bit.
- If the FIFOs are disabled (have a depth of one location) and data is received thereby filling the location, the **RXRIS** bit is set. The receive interrupt is cleared by performing a single read of the receive FIFO, or by clearing the interrupt by writing a 1 to the **RXIC** bit.

The transmit interrupt changes state when one of the following events occurs:

- If the FIFOs are enabled and the transmit FIFO progresses through the programmed trigger level, the TXRIS bit is set. The transmit interrupt is based on a transition through level, therefore the FIFO must be written past the programmed trigger level otherwise no further transmit interrupts will be generated. The transmit interrupt is cleared by writing data to the transmit FIFO until it becomes greater than the trigger level, or by clearing the interrupt by writing a 1 to the TXIC bit.
- If the FIFOs are disabled (have a depth of one location) and there is no data present in the transmitters single location, the TXRIS bit is set. It is cleared by performing a single write to the transmit FIFO, or by clearing the interrupt by writing a 1 to the TXIC bit.

### 11.3.6 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.

## 11.4 Initialization and Configuration

To use the UARTs, the peripheral clock must be enabled by setting the UART0 or UART1 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 375, 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:

$$\text{UARTFBRD[DIVFRAC]} = \text{integer}(0.8507 * 64 + 0.5) = 54$$

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. Enable the UART by setting the **UARTEN** bit in the **UARTCTL** register.

## 11.5 Register Map

Table 11-3 on page 379 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

Note that the UART module clock must be enabled before the registers can be programmed (see page 199). There must be a delay of 3 system clocks after the UART module clock is enabled before any UART module registers are accessed.

**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 11-3. UART Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	UARTDR	R/W	0x0000.0000	UART Data	381
0x004	UARTRSR/UARTECR	R/W	0x0000.0000	UART Receive Status/Error Clear	383
0x018	UARTFR	RO	0x0000.0090	UART Flag	385
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	0x0000.0000	UART Raw Interrupt Status	397
0x040	UARTMIS	RO	0x0000.0000	UART Masked Interrupt Status	398
0x044	UARTICR	W1C	0x0000.0000	UART Interrupt Clear	399
0xFD0	UARTPeriphID4	RO	0x0000.0000	UART Peripheral Identification 4	401
0xFD4	UARTPeriphID5	RO	0x0000.0000	UART Peripheral Identification 5	402
0xFD8	UARTPeriphID6	RO	0x0000.0000	UART Peripheral Identification 6	403
0xFDC	UARTPeriphID7	RO	0x0000.0000	UART Peripheral Identification 7	404
0xFE0	UARTPeriphID0	RO	0x0000.0011	UART Peripheral Identification 0	405
0xFE4	UARTPeriphID1	RO	0x0000.0000	UART Peripheral Identification 1	406
0xFE8	UARTPeriphID2	RO	0x0000.0018	UART Peripheral Identification 2	407

**Table 11-3. UART Register Map (continued)**

Offset	Name	Type	Reset	Description	See page
0xFEC	UARTPeriphID3	RO	0x0000.0001	UART Peripheral Identification 3	408
0xFF0	UARTPCellID0	RO	0x0000.000D	UART PrimeCell Identification 0	409
0xFF4	UARTPCellID1	RO	0x0000.00F0	UART PrimeCell Identification 1	410
0xFF8	UARTPCellID2	RO	0x0000.0005	UART PrimeCell Identification 2	411
0xFFC	UARTPCellID3	RO	0x0000.00B1	UART PrimeCell Identification 3	412

## 11.6 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

**Important:** This register is read-sensitive. See the register description for details.

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.

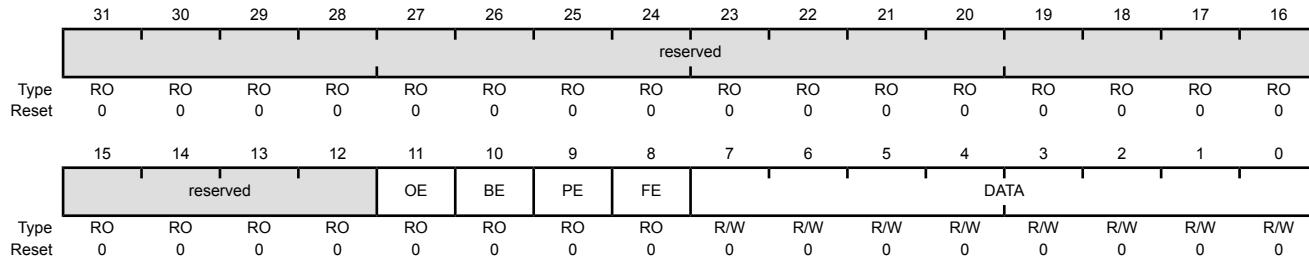
### UART Data (UARTDR)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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	<b>UART Parity Error</b> 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	<b>UART Framing Error</b> 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	<b>Data Transmitted or Received</b> 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.

### Reads

#### UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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															
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	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	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. This bit is cleared to 0 by a write to <b>UARTECR</b> .
0	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). 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.

## Writes

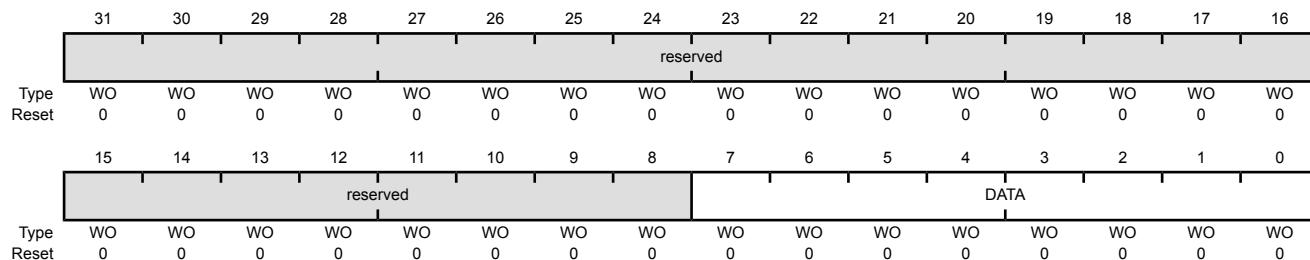
### UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

Offset 0x004

Type WO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	WO	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	DATA	WO	0	Error Clear A write to this register of any data clears the framing, parity, break, and overrun flags.

## 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

Offset 0x018

Type RO, reset 0x0000.0090

	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								TXFE	RXFF	TXFF	RXFE	BUSY	reserved		
Type	RO	RO	RO	RO	RO	RO	RO	RO	0	1	0	0	1	0	0	0
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	TXFE	RO	1	UART Transmit FIFO Empty The meaning of this bit depends on the state of the FEN bit in the <b>UARTLCRH</b> register. If the FIFO is disabled (FEN is 0), this bit is set when the transmit holding register is empty. If the FIFO is enabled (FEN 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 FEN 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 FEN 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.
4	RXFE	RO	1	UART Receive FIFO Empty The meaning of this bit depends on the state of the FEN 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.

Bit/Field	Name	Type	Reset	Description
3	BUSY	RO	0	<p>UART Busy</p> <p>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.</p> <p>This bit is set as soon as the transmit FIFO becomes non-empty (regardless of whether UART is enabled).</p>
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 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 375 for configuration details.

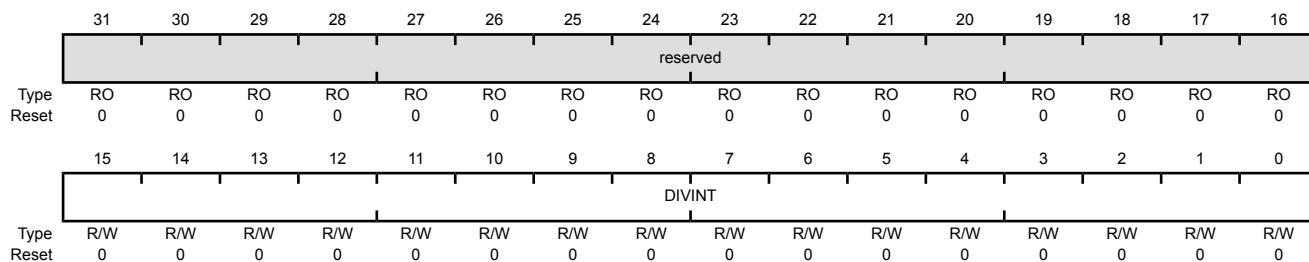
### UART Integer Baud-Rate Divisor (UARTIBRD)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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 5: 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 375 for configuration details.

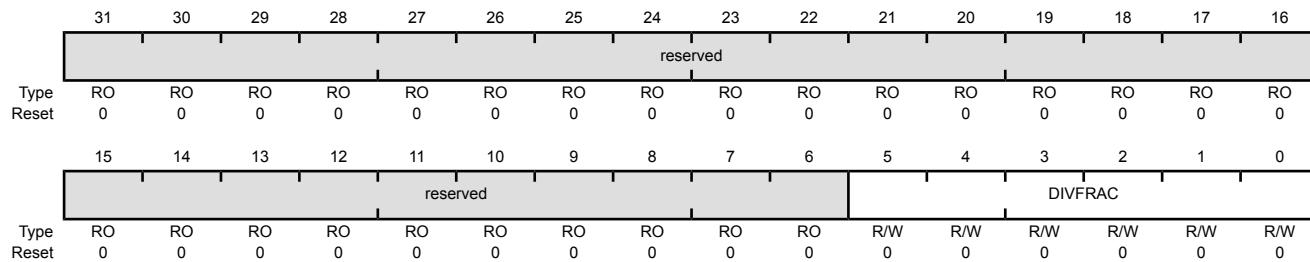
UART Fractional Baud-Rate Divisor (UARTFBRD)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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 6: 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

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	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
31:8	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	UART Stick Parity Select 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. When this bit is cleared, stick parity is disabled.
6:5	WLEN	R/W	0	UART Word Length The bits indicate the number of data bits transmitted or received in a frame as follows:  Value Description 0x3 8 bits 0x2 7 bits 0x1 6 bits 0x0 5 bits (default)
4	FEN	R/W	0	UART Enable FIFOs If this bit is set to 1, transmit and receive FIFO buffers are enabled (FIFO mode). When cleared to 0, FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers.
3	STP2	R/W	0	UART Two Stop Bits Select 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.

Bit/Field	Name	Type	Reset	Description
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>UNTX</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 7: 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

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						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
reserved																
Type	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	RO	RO	RO	RO	RO	RO	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.  <b>Note:</b> To enable reception, the <b>UARTEN</b> bit must also be set.
8	TXE	R/W	1	UART Transmit Enable 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.  <b>Note:</b> To enable transmission, the <b>UARTEN</b> bit must also be set.

Bit/Field	Name	Type	Reset	Description
7	LBE	R/W	0	UART Loop Back Enable If this bit is set to 1, the <code>UnTX</code> path is fed through the <code>UnRX</code> path.
6:1	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.
0	UARTEN	R/W	0	UART Enable 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.

## Register 8: 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.

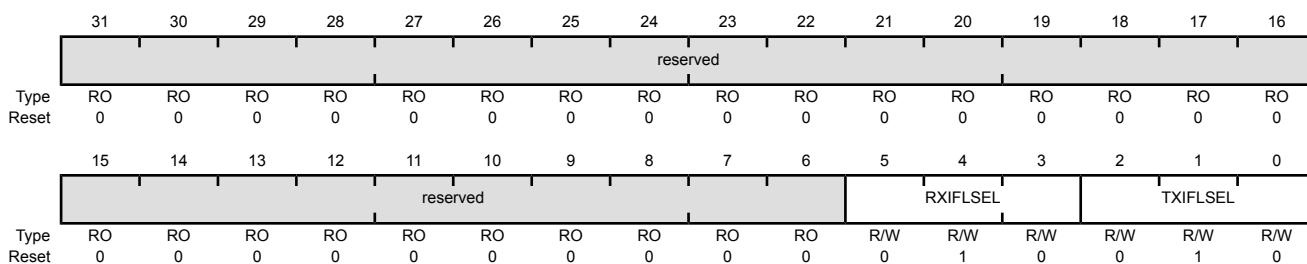
### UART Interrupt FIFO Level Select (UARTIFLS)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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 $\geq \frac{1}{8}$ full
0x1	RX FIFO $\geq \frac{1}{4}$ full
0x2	RX FIFO $\geq \frac{1}{2}$ full (default)
0x3	RX FIFO $\geq \frac{3}{4}$ full
0x4	RX FIFO $\geq \frac{7}{8}$ 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:														
				<table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0x0</td><td>TX FIFO <math>\leq \frac{1}{8}</math> empty</td></tr><tr><td>0x1</td><td>TX FIFO <math>\leq \frac{3}{4}</math> empty</td></tr><tr><td>0x2</td><td>TX FIFO <math>\leq \frac{1}{2}</math> empty (default)</td></tr><tr><td>0x3</td><td>TX FIFO <math>\leq \frac{1}{4}</math> empty</td></tr><tr><td>0x4</td><td>TX FIFO <math>\leq \frac{1}{8}</math> empty</td></tr><tr><td>0x5-0x7</td><td>Reserved</td></tr></tbody></table>	Value	Description	0x0	TX FIFO $\leq \frac{1}{8}$ empty	0x1	TX FIFO $\leq \frac{3}{4}$ empty	0x2	TX FIFO $\leq \frac{1}{2}$ empty (default)	0x3	TX FIFO $\leq \frac{1}{4}$ empty	0x4	TX FIFO $\leq \frac{1}{8}$ empty	0x5-0x7	Reserved
Value	Description																	
0x0	TX FIFO $\leq \frac{1}{8}$ empty																	
0x1	TX FIFO $\leq \frac{3}{4}$ empty																	
0x2	TX FIFO $\leq \frac{1}{2}$ empty (default)																	
0x3	TX FIFO $\leq \frac{1}{4}$ empty																	
0x4	TX FIFO $\leq \frac{1}{8}$ empty																	
0x5-0x7	Reserved																	

## Register 9: 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

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	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.
4	RXIM	R/W	0	UART Receive Interrupt Mask On a read, the current mask for the <b>RXIM</b> interrupt is returned. Setting this bit to 1 promotes the <b>RXIM</b> interrupt to the interrupt controller.

Bit/Field	Name	Type	Reset	Description
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 10: 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.

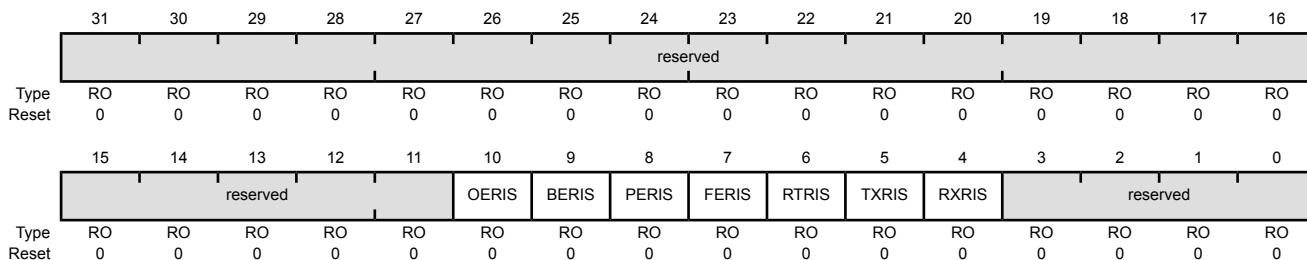
### UART Raw Interrupt Status (UARTRIS)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

Offset 0x03C

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	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	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 11: 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

Offset 0x040

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					OEMIS	BEMIS	PEMIS	FEMIS	RTMIS	TXMIS	RXMIS	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
31:11	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 12: 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.

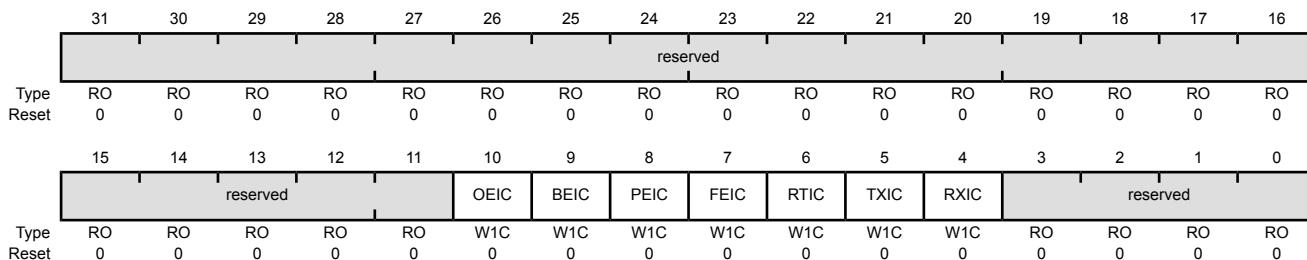
### UART Interrupt Clear (UARTICR)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

Offset 0x044

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	OEIC	W1C	0	Overrun Error Interrupt Clear The <b>OEIC</b> values are defined as follows:  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:  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:  Value Description 0 No effect on the interrupt. 1 Clears interrupt.
7	FEIC	W1C	0	Framing Error Interrupt Clear The <b>FEIC</b> values are defined as follows:  Value Description 0 No effect on the interrupt. 1 Clears interrupt.

Bit/Field	Name	Type	Reset	Description						
6	RTIC	W1C	0	<p>Receive Time-Out Interrupt Clear</p> <p>The RTIC values are defined as follows:</p> <table> <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.									
5	TXIC	W1C	0	<p>Transmit Interrupt Clear</p> <p>The TXIC values are defined as follows:</p> <table> <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.									
4	RXIC	W1C	0	<p>Receive Interrupt Clear</p> <p>The RXIC values are defined as follows:</p> <table> <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.									
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 13: UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

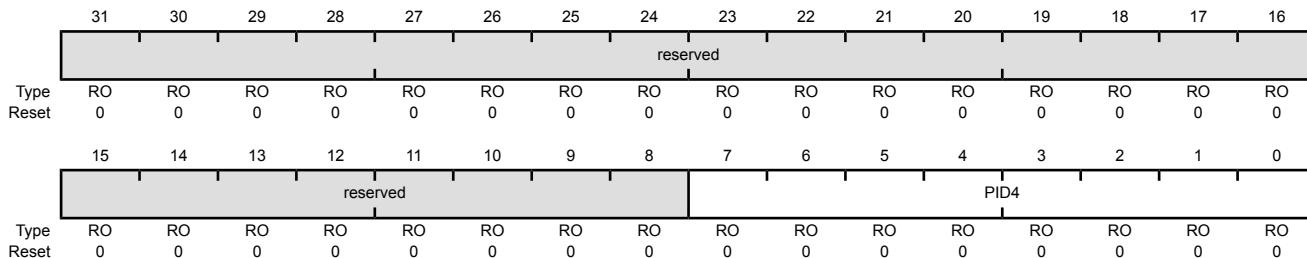
### UART Peripheral Identification 4 (UARTPeriphID4)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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 14: UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4**

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

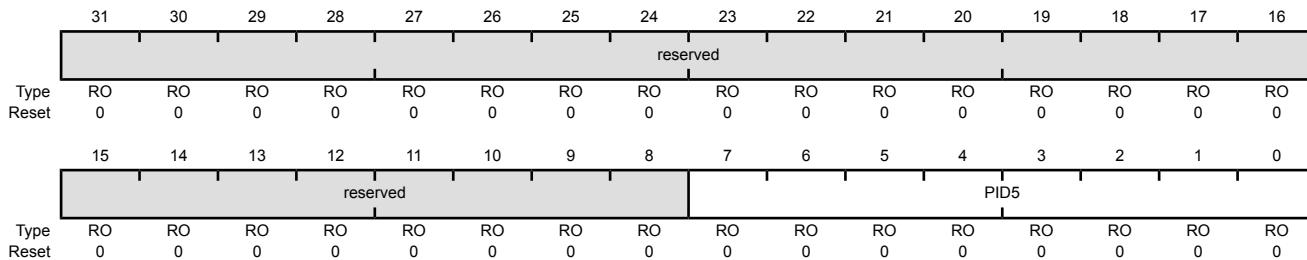
## UART Peripheral Identification 5 (UARTPeriphID5)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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	0x0000	UART Peripheral ID Register[15:8] Can be used by software to identify the presence of this peripheral.

## Register 15: UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

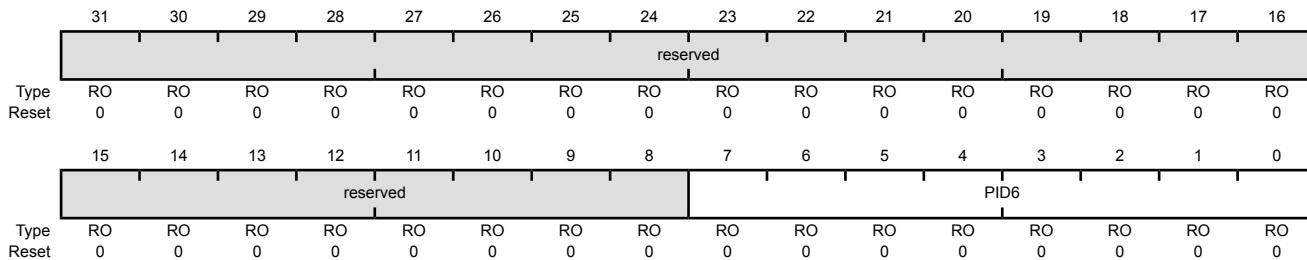
### UART Peripheral Identification 6 (UARTPeriphID6)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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 16: UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

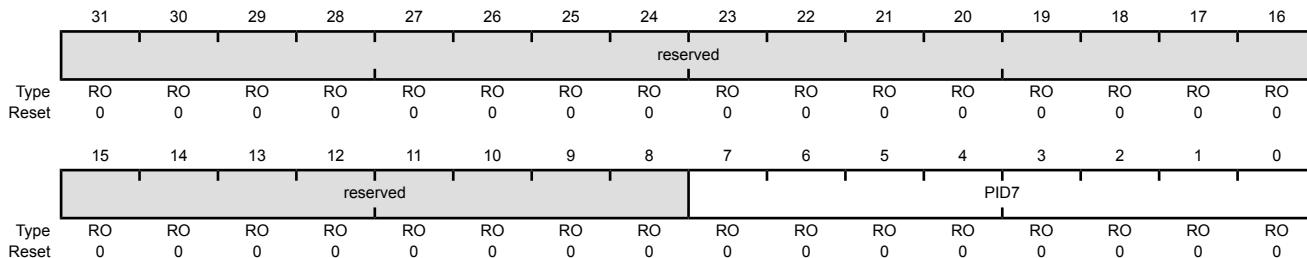
### UART Peripheral Identification 7 (UARTPeriphID7)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

Offset 0xFDC

Type RO, 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	PID7	RO	0x0000	UART Peripheral ID Register[31:24] Can be used by software to identify the presence of this peripheral.

## Register 17: UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

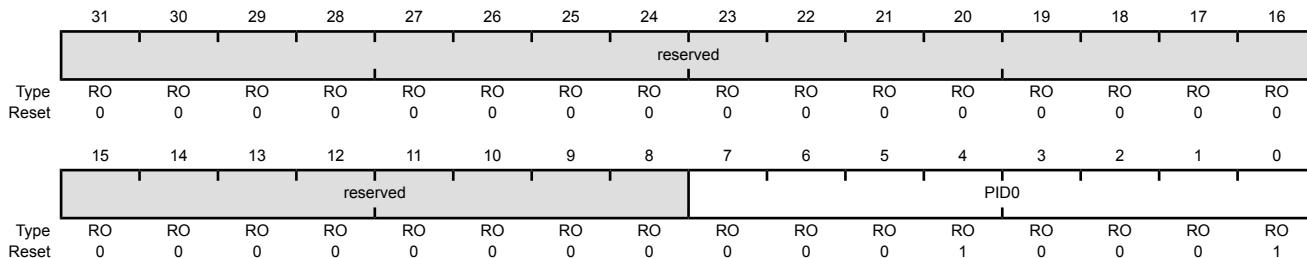
### UART Peripheral Identification 0 (UARTPeriphID0)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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 18: UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4**

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

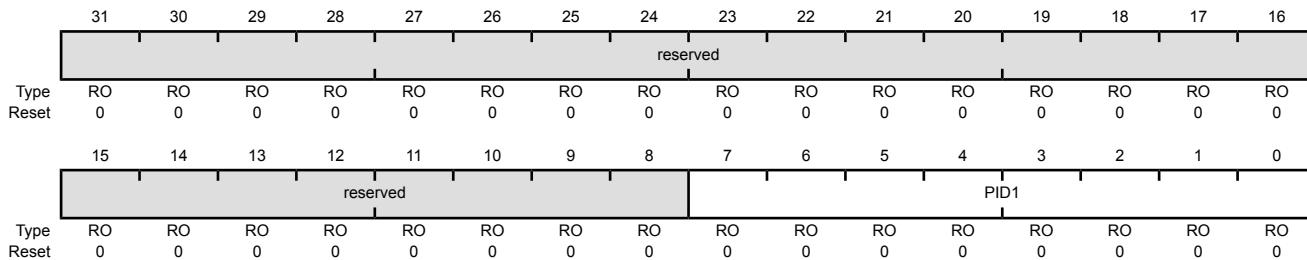
## UART Peripheral Identification 1 (UARTPeriphID1)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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	UART Peripheral ID Register[15:8] Can be used by software to identify the presence of this peripheral.

## Register 19: UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

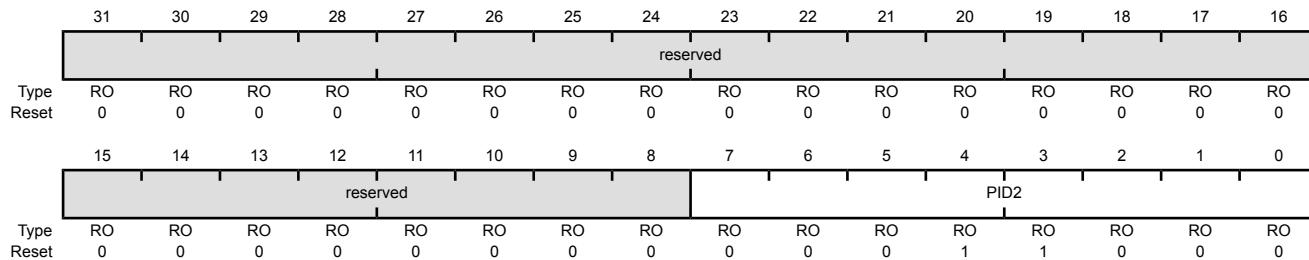
### UART Peripheral Identification 2 (UARTPeriphID2)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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 20: UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

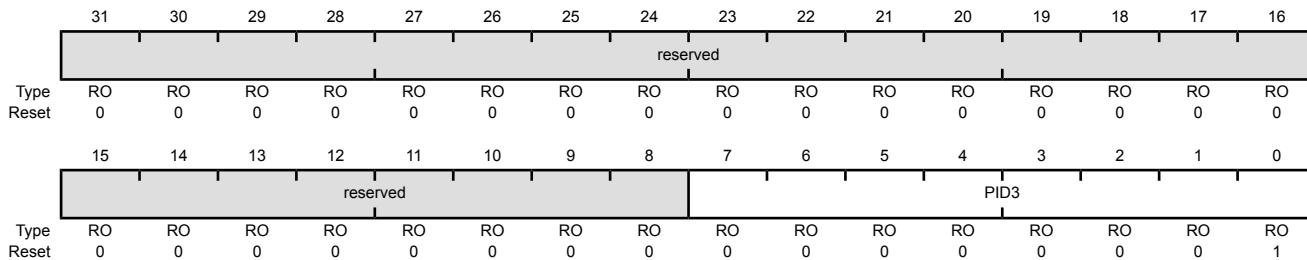
### UART Peripheral Identification 3 (UARTPeriphID3)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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	UART Peripheral ID Register[31:24] Can be used by software to identify the presence of this peripheral.

## Register 21: 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

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	UART PrimeCell ID Register[7:0] Provides software a standard cross-peripheral identification system.

## Register 22: 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

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 23: UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

### UART PrimeCell Identification 2 (UARTPCellID2)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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	UART PrimeCell ID Register[23:16] Provides software a standard cross-peripheral identification system.

## Register 24: UART PrimeCell Identification 3 (UARTPCellID3), offset 0xFFC

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

### UART PrimeCell Identification 3 (UARTPCellID3)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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.

## 12 Synchronous Serial Interface (SSI)

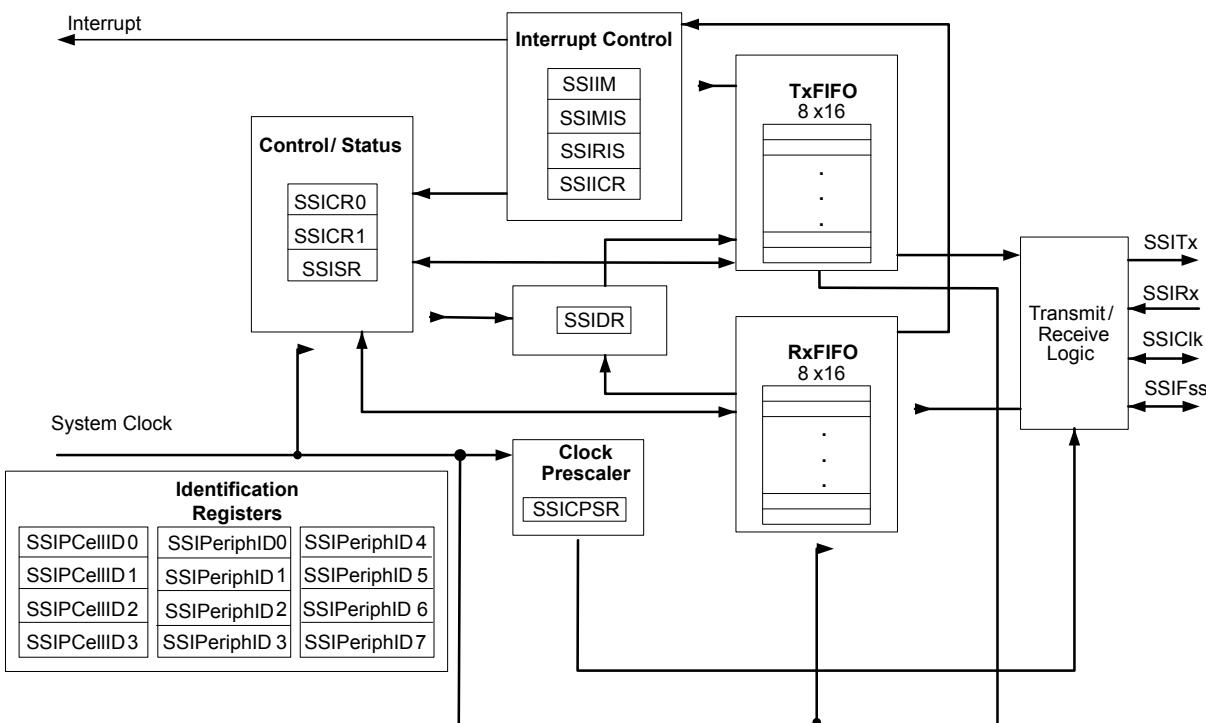
The Stellaris® Synchronous Serial Interface (SSI) is a master or slave interface for synchronous serial communication with peripheral devices that have either Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces.

The Stellaris SSI module has the following features:

- 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

### 12.1 Block Diagram

Figure 12-1. SSI Module Block Diagram



### 12.2 Signal Description

Table 12-1 on page 414 and Table 12-2 on page 414 list the external signals of the SSI module and describe the function of each. The SSI signals are alternate functions for some GPIO signals and

default to be GPIO signals at reset., with the exception of the `SSI0Clk`, `SSI0Fss`, `SSI0Rx`, and `SSI0Tx` pins which default to the SSI function. The column in the table below titled "Pin Assignment" lists the possible GPIO pin placements for the SSI signals. The `AFSEL` bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 253) should be set to choose the SSI function. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 232.

**Table 12-1. SSI Signals (48QFP)**

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
<code>SSIClk</code>	19	I/O	TTL	SSI clock.
<code>SSI0Fss</code>	20	I/O	TTL	SSI frame.
<code>SSI0Rx</code>	21	I	TTL	SSI receive.
<code>SSI0Tx</code>	22	O	TTL	SSI transmit.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

**Table 12-2. SSI Signals (48QFN)**

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
<code>SSIClk</code>	19	I/O	TTL	SSI clock.
<code>SSI0Fss</code>	20	I/O	TTL	SSI frame.
<code>SSI0Rx</code>	21	I	TTL	SSI receive.
<code>SSI0Tx</code>	22	O	TTL	SSI transmit.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

## 12.3 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.

### 12.3.1 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 1.5 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 `CPSDVSR` from 2 to 254, which is programmed in the **SSI Clock Prescale (SSICPSR)** register (see page 433). 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 426).

The frequency of the output clock `SSIClk` is defined by:

$$\text{SSIClk} = \text{FSysClk} / (\text{CPSDVSR} * (1 + \text{SCR}))$$

**Note:** 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 559 to view SSI timing parameters.

## 12.3.2 FIFO Operation

### 12.3.2.1 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 430), 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.

In slave mode, the SSI transmits data each time the master initiates a transaction. If the transmit FIFO is empty and the master initiates, the slave transmits the 8th most recent value in the transmit FIFO. If less than 8 values have been written to the transmit FIFO since the SSI module clock was enabled using the **SSI** bit in the **RGCG1** register, then 0 is transmitted. Care should be taken to ensure that valid data is in the FIFO as needed. The SSI can be configured to generate an interrupt when the FIFO is empty.

### 12.3.2.2 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.

## 12.3.3 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 (SSIIM)** register (see page 434). 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 436 and page 437, respectively).

## 12.3.4 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.

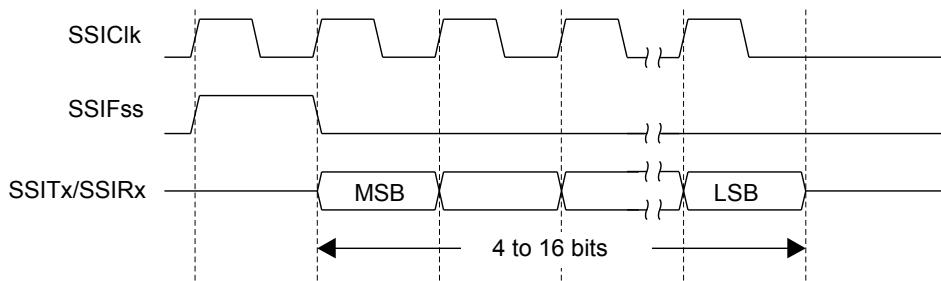
For Texas Instruments synchronous serial frame format, the **SSIFss** pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the SSI and the off-chip slave device drive their output data on the rising edge of **SSIClk**, 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.

#### 12.3.4.1 Texas Instruments Synchronous Serial Frame Format

Figure 12-2 on page 416 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

**Figure 12-2. TI Synchronous Serial Frame Format (Single Transfer)**

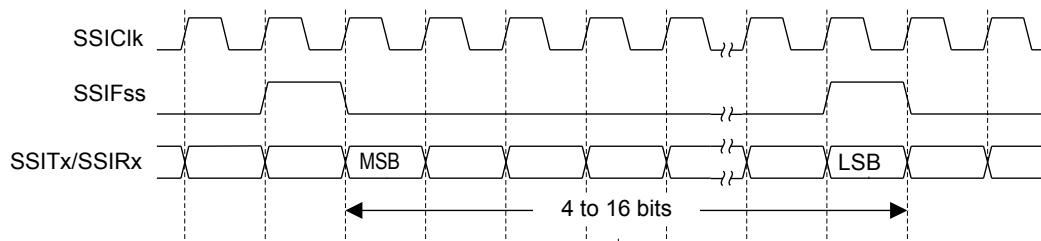


In this mode, **SSIClk** and **SSIFss** are forced Low, and the transmit data line **SSITx** is tristated whenever the SSI is idle. Once the bottom entry of the transmit FIFO contains data, **SSIFss** is pulsed High for one **SSIClk** period. The value to be transmitted is also transferred from the transmit 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 12-3 on page 417 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

**Figure 12-3. TI Synchronous Serial Frame Format (Continuous Transfer)**



#### 12.3.4.2 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.

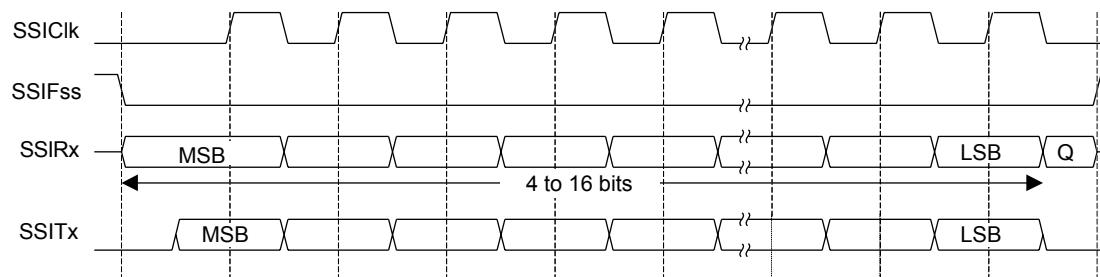
##### **SPH Phase Control Bit**

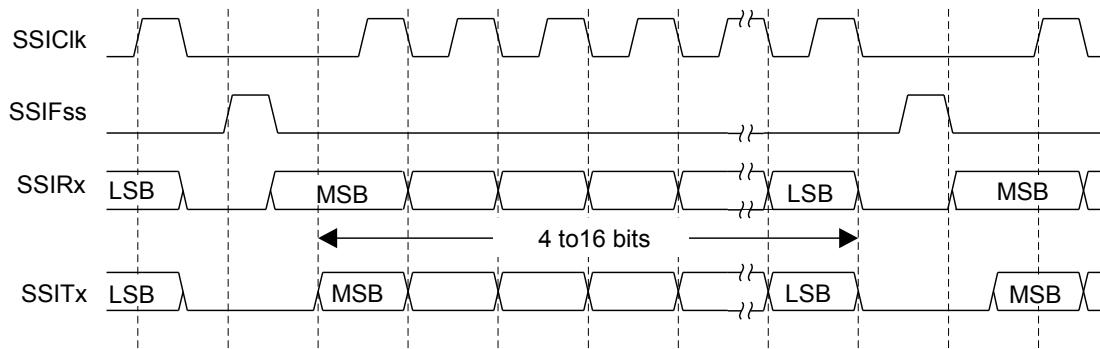
The **SPH** phase control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the **SPH** phase control bit is Low, data is captured on the first clock edge transition. If the **SPH** bit is High, data is captured on the second clock edge transition.

#### 12.3.4.3 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 12-4 on page 417 and Figure 12-5 on page 418.

**Figure 12-4. Freescale SPI Format (Single Transfer) with **SPO=0** and **SPH=0****



**Figure 12-5. 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.

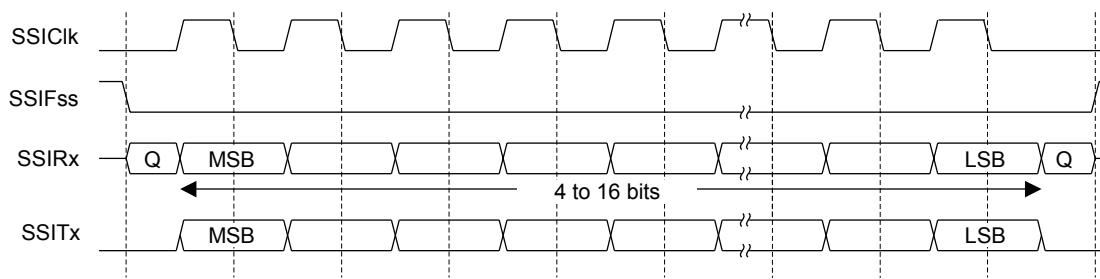
The data is now captured on the rising and propagated on the falling edges of the SSIClk signal.

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.

#### 12.3.4.4 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 12-6 on page 419, which covers both single and continuous transfers.

**Figure 12-6. 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.

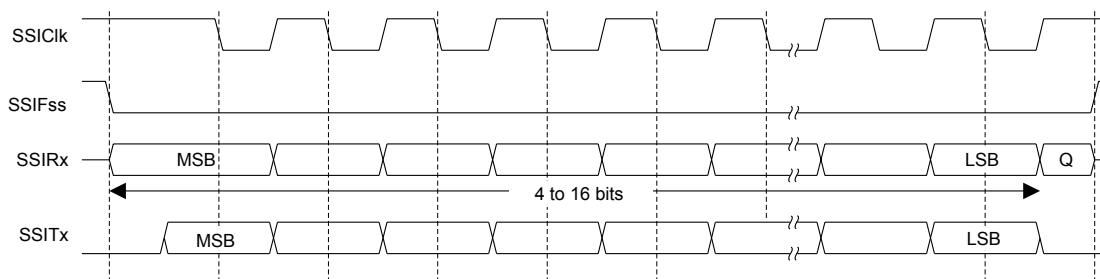
Data is then captured on the falling edges and propagated on the rising edges of the SSIClk signal.

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.

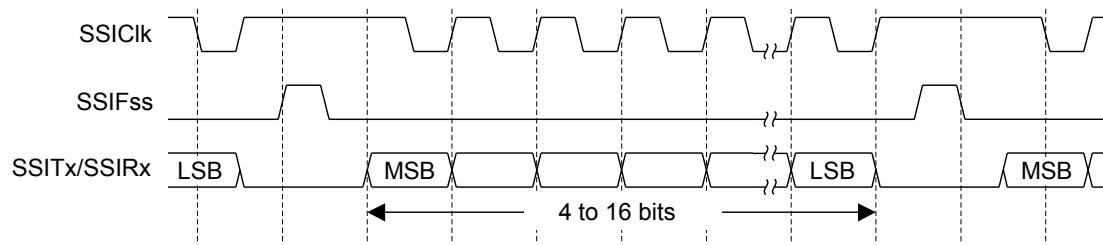
For continuous back-to-back transfers, the SSIFss pin is held Low between successive data words and termination is the same as that of the single word transfer.

#### 12.3.4.5 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 12-7 on page 419 and Figure 12-8 on page 420.

**Figure 12-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0**

**Note:** Q is undefined.

**Figure 12-8. 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.

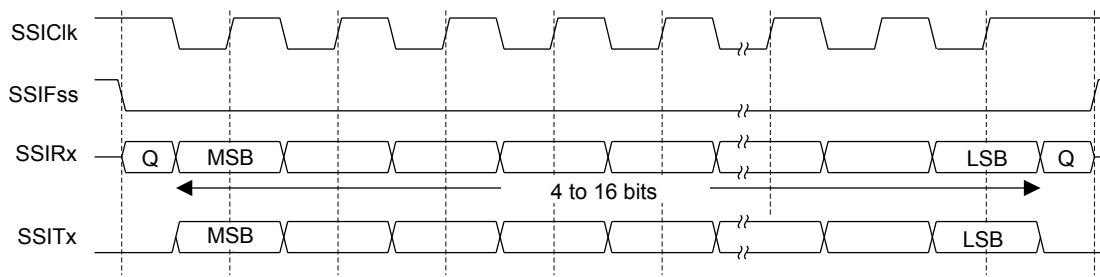
One half period later, valid master data is transferred to the SSITx line. Now that both the master and slave data have been set, the SSIClk master clock pin becomes Low after one further half SSIClk period. This means that data is captured on the falling edges and propagated on the rising edges of the 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.

#### 12.3.4.6 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 12-9 on page 421, which covers both single and continuous transfers.

**Figure 12-9. 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.

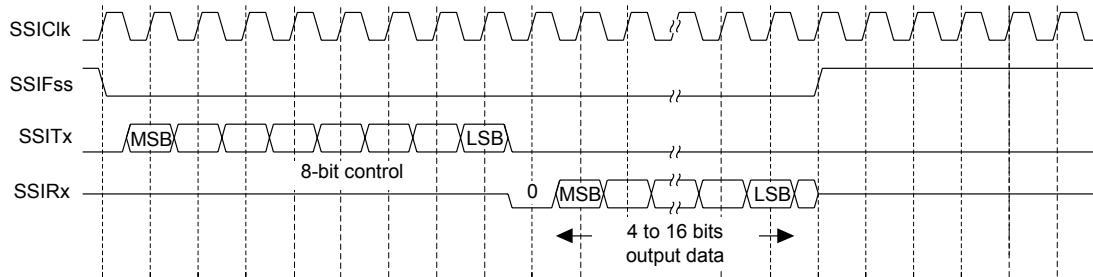
After all bits have been transferred, in the case of a single word transmission, the SSIFss line is returned to its idle high state one SSIClk period after the last bit has been captured.

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.

#### 12.3.4.7 MICROWIRE Frame Format

Figure 12-10 on page 421 shows the MICROWIRE frame format, again for a single frame. Figure 12-11 on page 422 shows the same format when back-to-back frames are transmitted.

**Figure 12-10. 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

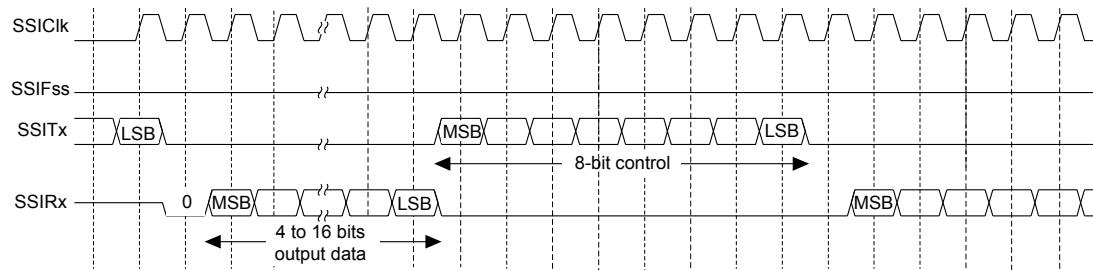
A transmission is triggered by writing a control byte to the transmit FIFO. The falling edge of SSIFss causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic, and the MSB of the 8-bit control frame to be shifted out onto 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.

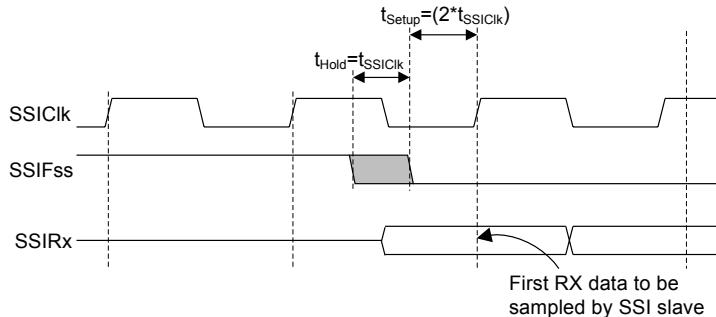
**Figure 12-11. MICROWIRE Frame Format (Continuous Transfer)**



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 12-12 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 12-12. MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements**



## 12.4 Initialization and Configuration

To use the SSI, its peripheral clock must be enabled by setting the **SSE** 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. 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:

$$\begin{aligned} F_{SSIClk} &= FSysClk / (CPSDVSR * (1 + SCR)) \\ 1 \times 10^6 &= 20 \times 10^6 / (CPSDVSR * (1 + SCR)) \end{aligned}$$

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.

## 12.5 Register Map

Table 12-3 on page 424 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 that the SSI module clock must be enabled before the registers can be programmed (see page 199). There must be a delay of 3 system clocks after the SSI module clock is enabled before any SSI module registers are accessed.

**Note:** The SSI must be disabled (see the SSE bit in the **SSICR1** register) before any of the control registers are reprogrammed.

**Table 12-3. SSI Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	SSICR0	R/W	0x0000.0000	SSI Control 0	426
0x004	SSICR1	R/W	0x0000.0000	SSI Control 1	428
0x008	SSIDR	R/W	0x0000.0000	SSI Data	430
0x00C	SSISR	RO	0x0000.0003	SSI Status	431
0x010	SSICPSR	R/W	0x0000.0000	SSI Clock Prescale	433
0x014	SSIIM	R/W	0x0000.0000	SSI Interrupt Mask	434
0x018	SSIRIS	RO	0x0000.0008	SSI Raw Interrupt Status	436
0x01C	SSIMIS	RO	0x0000.0000	SSI Masked Interrupt Status	437

**Table 12-3. SSI Register Map (continued)**

Offset	Name	Type	Reset	Description	See page
0x020	SSIICR	W1C	0x0000.0000	SSI Interrupt Clear	438
0xFD0	SSIPeriphID4	RO	0x0000.0000	SSI Peripheral Identification 4	439
0xFD4	SSIPeriphID5	RO	0x0000.0000	SSI Peripheral Identification 5	440
0xFD8	SSIPeriphID6	RO	0x0000.0000	SSI Peripheral Identification 6	441
0xFDC	SSIPeriphID7	RO	0x0000.0000	SSI Peripheral Identification 7	442
0xFE0	SSIPeriphID0	RO	0x0000.0022	SSI Peripheral Identification 0	443
0xFE4	SSIPeriphID1	RO	0x0000.0000	SSI Peripheral Identification 1	444
0xFE8	SSIPeriphID2	RO	0x0000.0018	SSI Peripheral Identification 2	445
0xFEC	SSIPeriphID3	RO	0x0000.0001	SSI Peripheral Identification 3	446
0xFF0	SSIPCellID0	RO	0x0000.000D	SSI PrimeCell Identification 0	447
0xFF4	SSIPCellID1	RO	0x0000.00F0	SSI PrimeCell Identification 1	448
0xFF8	SSIPCellID2	RO	0x0000.0005	SSI PrimeCell Identification 2	449
0xFFC	SSIPCellID3	RO	0x0000.00B1	SSI PrimeCell Identification 3	450

## 12.6 Register Descriptions

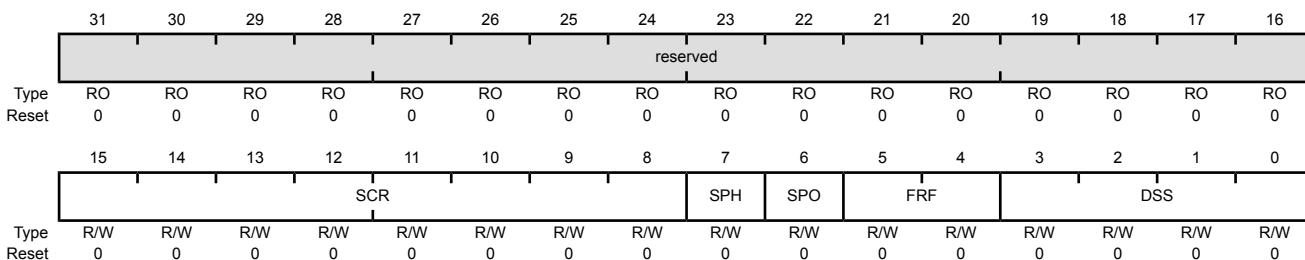
The remainder of this section lists and describes the SSI registers, in numerical order by address offset.

## 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



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	<p>SSI Serial Clock Rate</p> <p>The value SCR is used to generate the transmit and receive bit rate of the SSI. The bit rate is:</p> $BR = FSSIC1k / (CPSDVSR * (1 + SCR))$ <p>where CPSDVSR is an even value from 2-254 programmed in the <b>SSICPSR</b> register, and SCR is a value from 0-255.</p>										
7	SPH	R/W	0	<p>SSI Serial Clock Phase</p> <p>This bit is only applicable to the Freescale SPI Format.</p> <p>The SPH 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.</p> <p>When the SPH bit is 0, data is captured on the first clock edge transition. If SPH is 1, data is captured on the second clock edge transition.</p>										
6	SPO	R/W	0	<p>SSI Serial Clock Polarity</p> <p>This bit is only applicable to the Freescale SPI Format.</p> <p>When the SPO bit is 0, it produces a steady state Low value on the SSIC1k pin. If SPO is 1, a steady state High value is placed on the SSIC1k pin when data is not being transferred.</p>										
5:4	FRF	R/W	0x0	<p>SSI Frame Format Select</p> <p>The FRF values are defined as follows:</p> <table border="0"> <tr> <td>Value</td> <td>Frame Format</td> </tr> <tr> <td>0x0</td> <td>Freescale SPI Frame Format</td> </tr> <tr> <td>0x1</td> <td>Texas Instruments Synchronous Serial Frame Format</td> </tr> <tr> <td>0x2</td> <td>MICROWIRE Frame Format</td> </tr> <tr> <td>0x3</td> <td>Reserved</td> </tr> </table>	Value	Frame Format	0x0	Freescale SPI Frame Format	0x1	Texas Instruments Synchronous Serial Frame Format	0x2	MICROWIRE Frame Format	0x3	Reserved
Value	Frame Format													
0x0	Freescale SPI Frame Format													
0x1	Texas Instruments Synchronous Serial Frame Format													
0x2	MICROWIRE Frame Format													
0x3	Reserved													

---

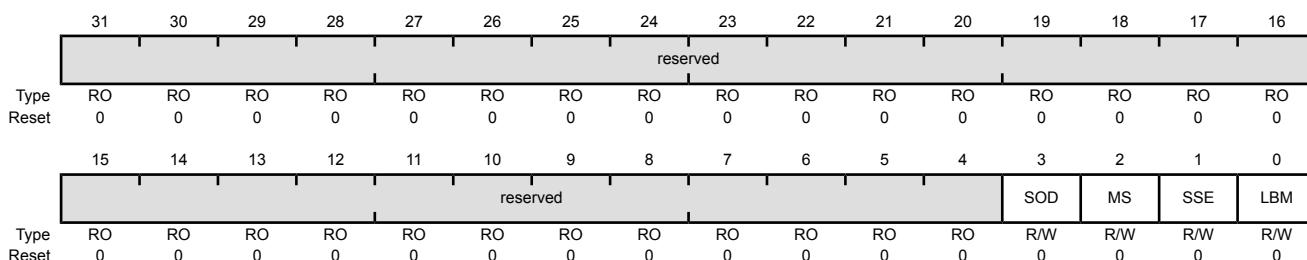
Bit/Field	Name	Type	Reset	Description
3:0	DSS	R/W	0x00	SSI Data Size Select The DSS values are defined as follows:
Value      Data Size				
0x0-0x2   Reserved				
0x3      4-bit data				
0x4      5-bit data				
0x5      6-bit data				
0x6      7-bit data				
0x7      8-bit data				
0x8      9-bit data				
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



#### 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 <b>SSITx</b> output in Slave Output mode.
1	SSI must not drive the <b>SSITx</b> 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 Setting this bit enables SSI operation. The SSE 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 Setting this bit enables Loopback Test mode. The LBM 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

**Important:** This register is read-sensitive. See the register description for details.

**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.

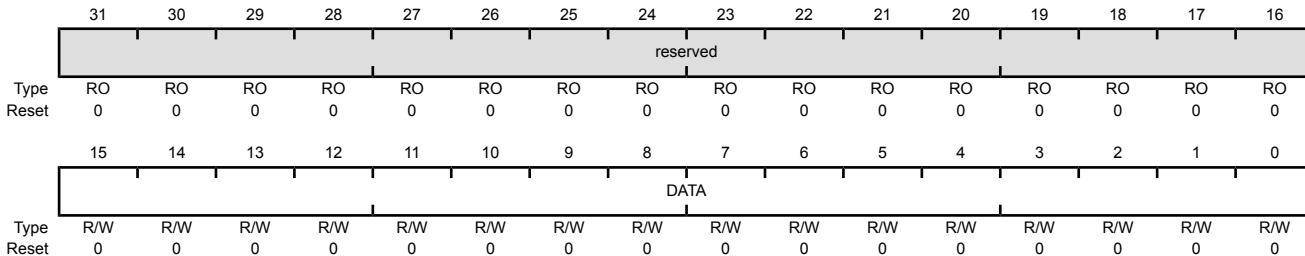
When the SSI is programmed for MICROWIRE frame format, the default size for transmit data is eight bits (the most significant byte is ignored). The receive data size is controlled by the programmer. The transmit FIFO and the receive FIFO are not cleared even when the **SSE** bit in the **SSICR1** register 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



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	<p>SSI Receive/Transmit Data</p> <p>A read operation reads the receive FIFO. A write operation writes the transmit FIFO.</p> <p>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.</p>

## 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															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	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:  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:  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:  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:  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.

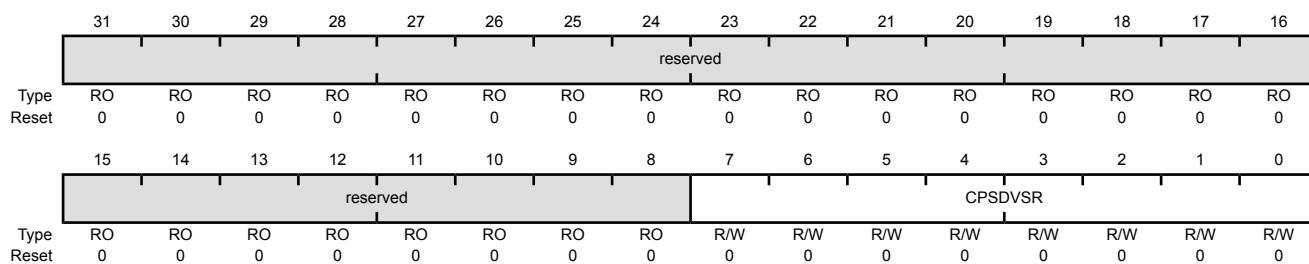
The value programmed into this register must be an even number between 2 and 254. The least-significant bit of the programmed number is hard-coded to zero. If an odd number is written to this register, data read back from this register has the least-significant bit as zero.

### 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 SSIClk. 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															
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      TXIM      R/W      0      SSI Transmit FIFO Interrupt Mask  
The TXIM values are defined as follows:

#### Value      Description

- 0      TX FIFO half-empty or less condition interrupt is masked.
- 1      TX FIFO half-empty or less condition interrupt is not masked.

2      RXIM      R/W      0      SSI Receive FIFO Interrupt Mask  
The RXIM 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 RTIM 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

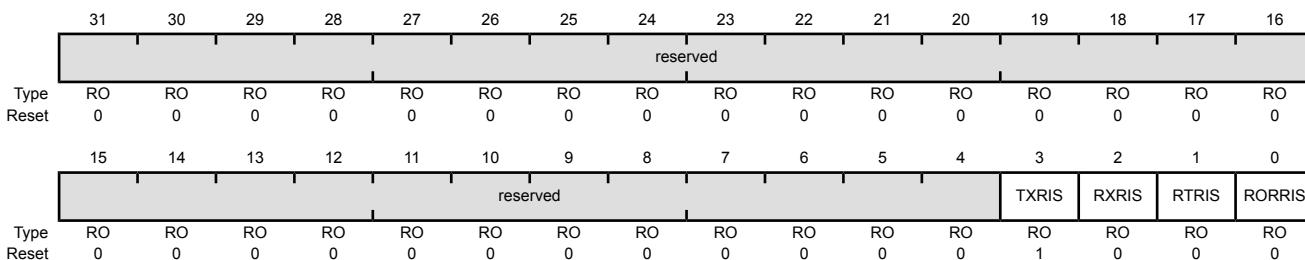
The **SSIRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect.

### SSI Raw Interrupt Status (SSIRIS)

SSI0 base: 0x4000.8000

Offset 0x018

Type RO, reset 0x0000.0008



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 empty 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

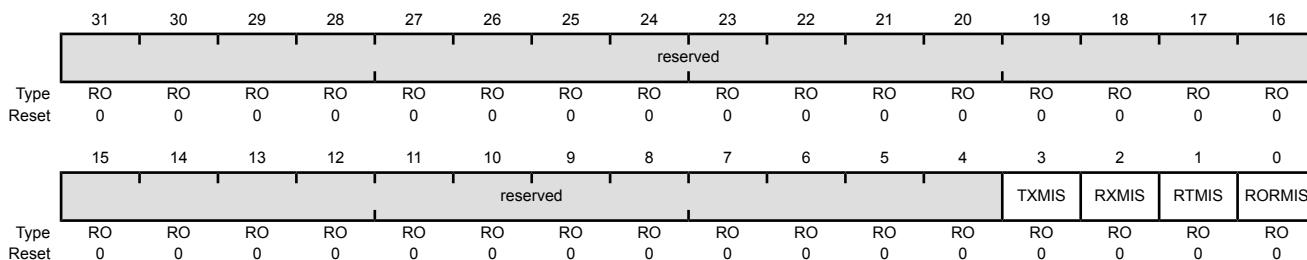
The **SSIMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

### SSI Masked Interrupt Status (SSIMIS)

SSI0 base: 0x4000.8000

Offset 0x01C

Type RO, reset 0x0000.0000



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 empty 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

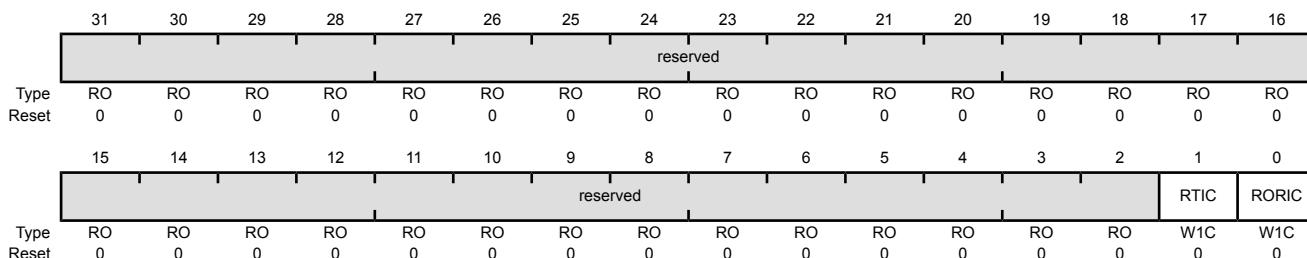
The **SSIICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

### SSI Interrupt Clear (SSIICR)

SSI0 base: 0x4000.8000

Offset 0x020

Type W1C, reset 0x0000.0000



#### 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 **RTIC** 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 **RORIC** values are defined as follows:

Value	Description
0	No effect on interrupt.
1	Clears interrupt.

## Register 10: SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0

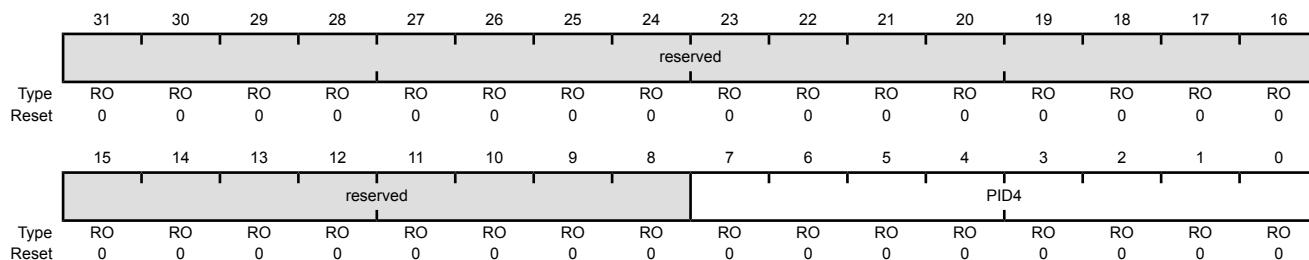
The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### SSI Peripheral Identification 4 (SSIPeriphID4)

SSI0 base: 0x4000.8000

Offset 0xFD0

Type RO, reset 0x0000.0000



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 11: SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4

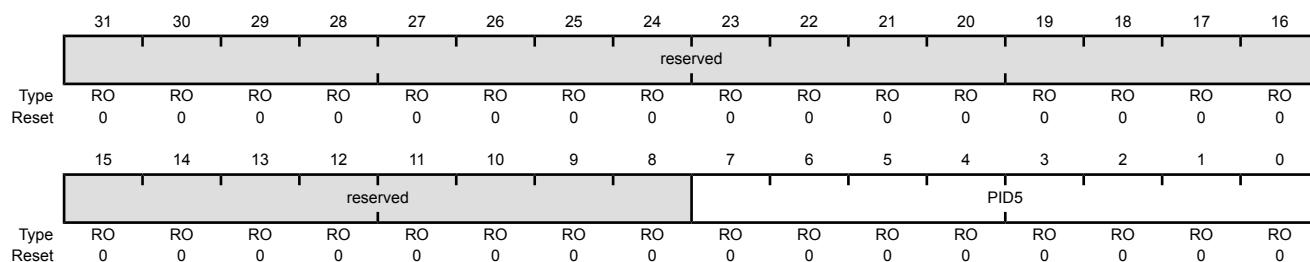
The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### 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 12: SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8

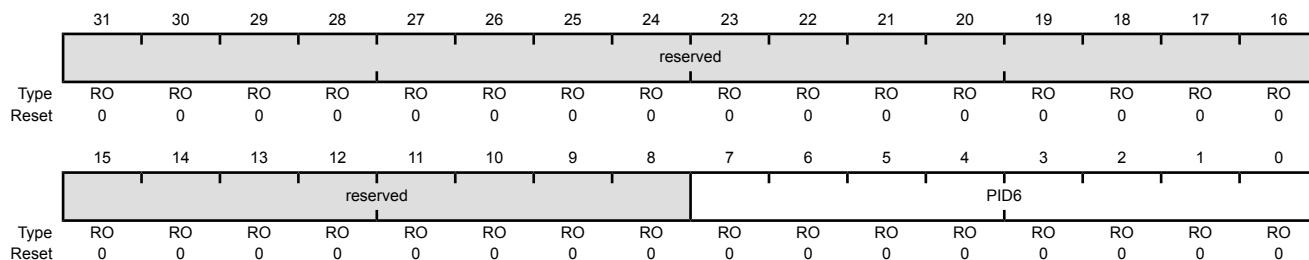
The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### SSI Peripheral Identification 6 (SSIPeriphID6)

SSI0 base: 0x4000.8000

Offset 0xFD8

Type RO, reset 0x0000.0000



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 13: SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC

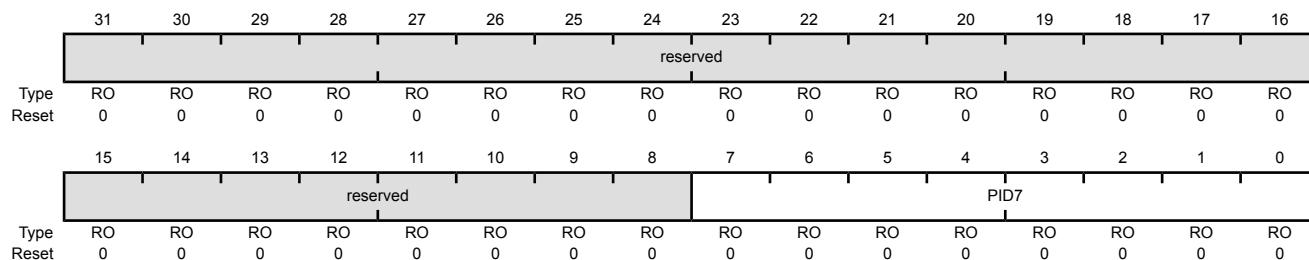
The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### 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 14: SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0

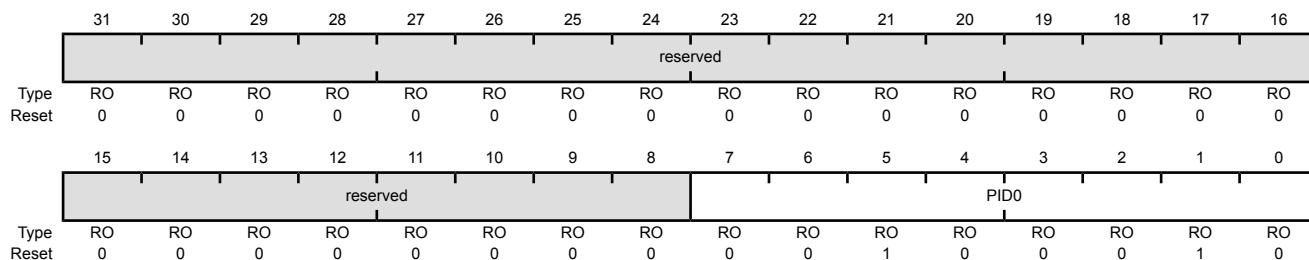
The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### SSI Peripheral Identification 0 (SSIPeriphID0)

SSI0 base: 0x4000.8000

Offset 0xFE0

Type RO, reset 0x0000.0022



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 15: SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4

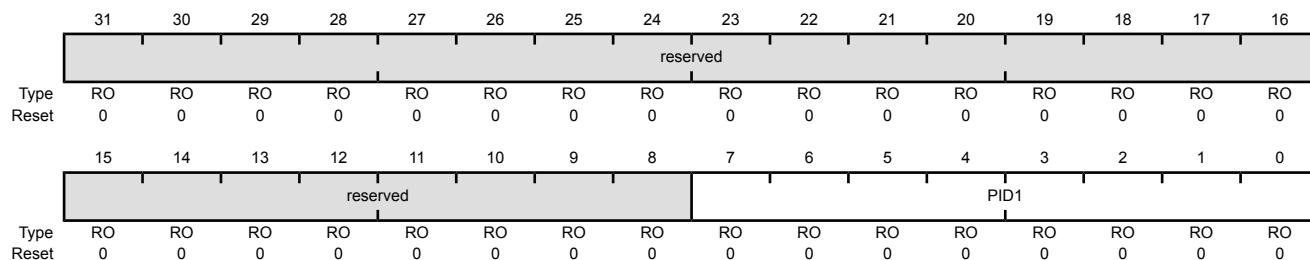
The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### 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 16: SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8

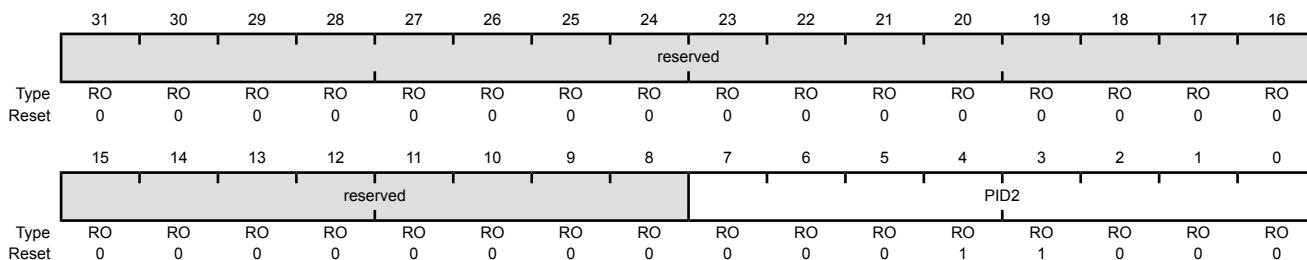
The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### SSI Peripheral Identification 2 (SSIPeriphID2)

SSI0 base: 0x4000.8000

Offset 0xFE8

Type RO, reset 0x0000.0018



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 17: SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC

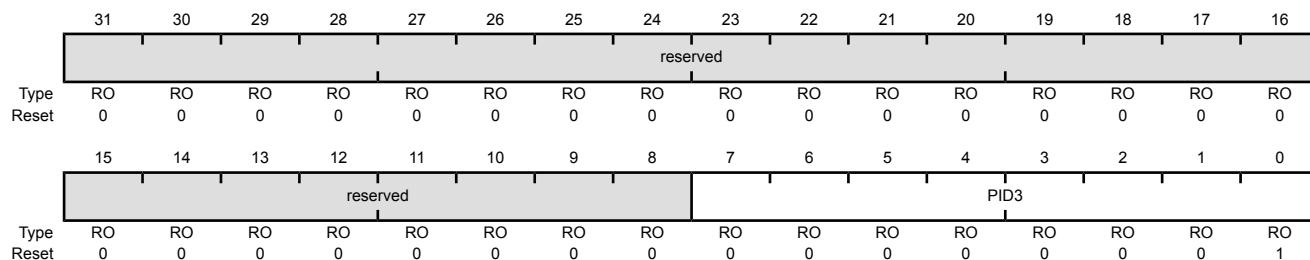
The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### 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 18: SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0

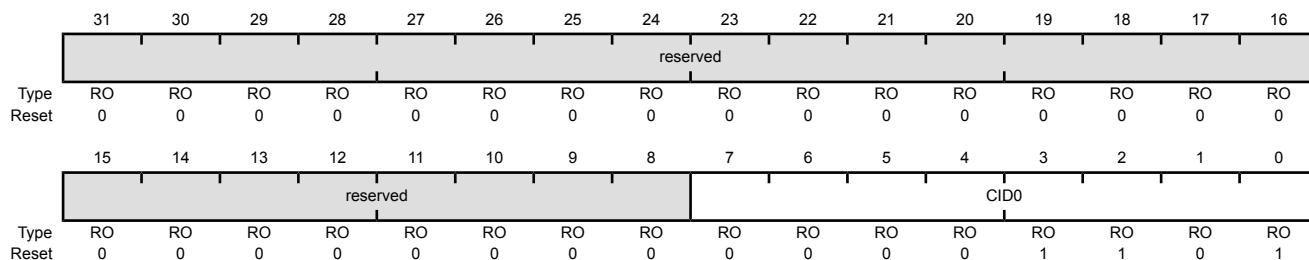
The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

### SSI PrimeCell Identification 0 (SSIPCellID0)

SSI0 base: 0x4000.8000

Offset 0xFF0

Type RO, reset 0x0000.000D



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 19: SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4**

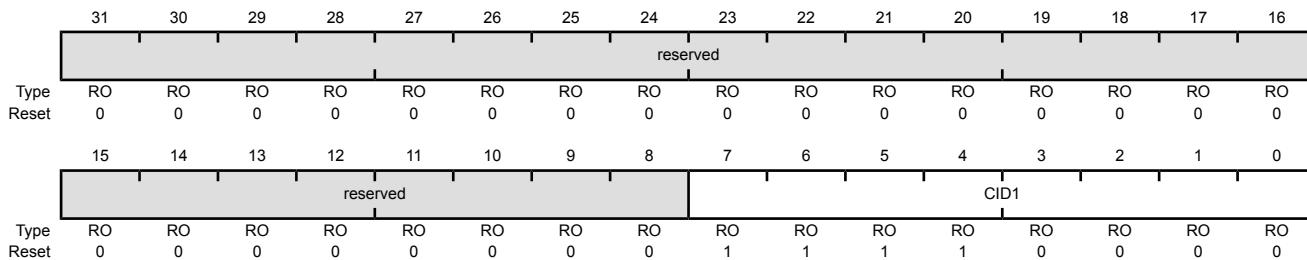
The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

## 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 20: SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8

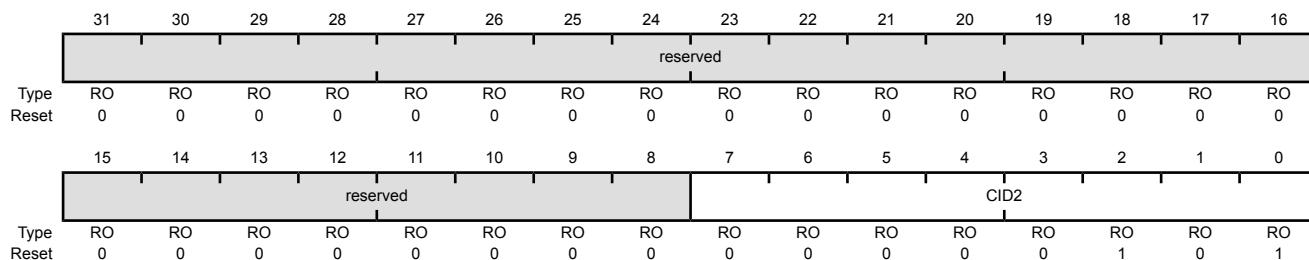
The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

### SSI PrimeCell Identification 2 (SSIPCellID2)

SSI0 base: 0x4000.8000

Offset 0xFF8

Type RO, reset 0x0000.0005



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 21: SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC

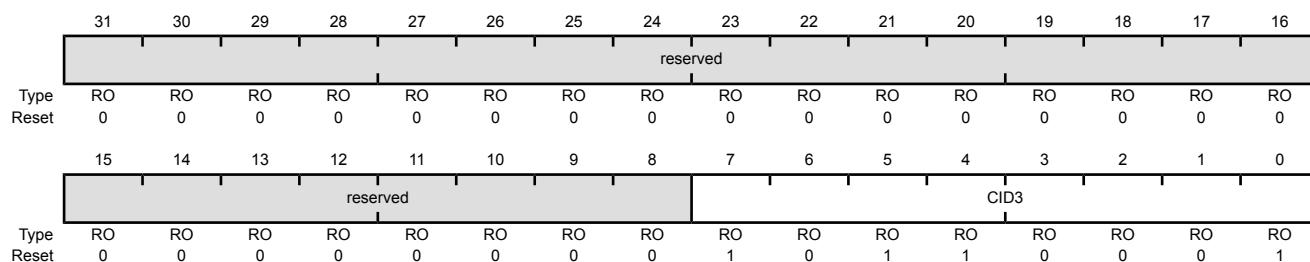
The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

### 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.

## 13 Inter-Integrated Circuit (I<sup>2</sup>C) Interface

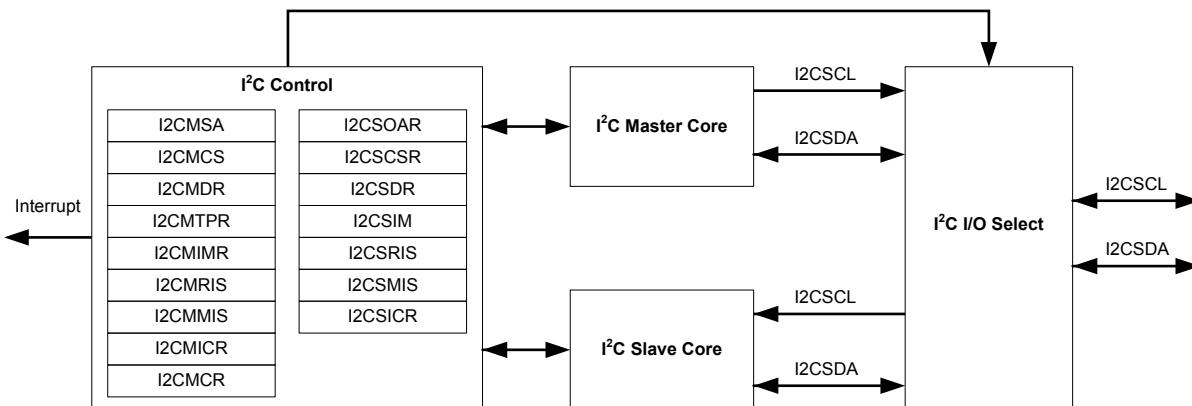
The Inter-Integrated Circuit (I<sup>2</sup>C) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL), and interfaces to external I<sup>2</sup>C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and 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 LM3S811 microcontroller includes one I<sup>2</sup>C module, providing the ability to interact (both send and receive) with other I<sup>2</sup>C devices on the bus.

The Stellaris® I<sup>2</sup>C interface has the following features:

- Devices on the I<sup>2</sup>C bus can be designated as either a master or a slave
  - Supports both sending and receiving data as either a master or a slave
  - Supports simultaneous master and slave operation
- Four I<sup>2</sup>C modes
  - Master transmit
  - Master receive
  - Slave transmit
  - Slave receive
- Two transmission speeds: Standard (100 Kbps) and Fast (400 Kbps)
- Master and slave interrupt generation
  - Master generates interrupts when a transmit or receive operation completes (or aborts due to an error)
  - Slave generates interrupts when data has been sent or requested by a master
- Master with arbitration and clock synchronization, multimaster support, and 7-bit addressing mode

## 13.1 Block Diagram

Figure 13-1. I<sup>2</sup>C Block Diagram



## 13.2 Signal Description

Table 13-1 on page 452 and Table 13-2 on page 452 list the external signals of the I<sup>2</sup>C interface and describe the function of each. The I<sup>2</sup>C interface signals are alternate functions for some GPIO signals and default to be GPIO signals at reset., with the exception of the I<sup>2</sup>C<sub>SCL</sub> and I<sup>2</sup>C<sub>SDA</sub> pins which default to the I<sup>2</sup>C function. The column in the table below titled "Pin Assignment" lists the possible GPIO pin placements for the I<sup>2</sup>C signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 253) should be set to choose the I<sup>2</sup>C function. Note that the I<sup>2</sup>C pins should be set to open drain using the **GPIO Open Drain Select (GPIOODR)** register. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 232.

Table 13-1. I<sup>2</sup>C Signals (48QFP)

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
I <sup>2</sup> C <sub>SCL</sub>	33	I/O	OD	I <sup>2</sup> C clock.
I <sup>2</sup> C <sub>SDA</sub>	34	I/O	OD	I <sup>2</sup> C data.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

Table 13-2. I<sup>2</sup>C Signals (48QFN)

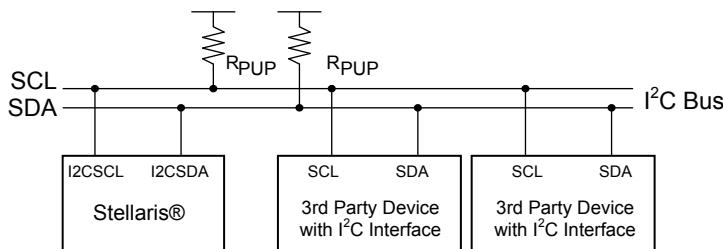
Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
I <sup>2</sup> C <sub>SCL</sub>	33	I/O	OD	I <sup>2</sup> C clock.
I <sup>2</sup> C <sub>SDA</sub>	34	I/O	OD	I <sup>2</sup> C data.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

## 13.3 Functional Description

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 13-2 on page 453.

See "Inter-Integrated Circuit (I<sup>2</sup>C) Interface" on page 561 for I<sup>2</sup>C timing diagrams.

**Figure 13-2. I<sup>2</sup>C Bus Configuration**

### 13.3.1 I<sup>2</sup>C Bus Functional Overview

The I<sup>2</sup>C bus uses only two signals: SDA and SCL, named I<sup>2</sup>CSDA and I<sup>2</sup>CSCl on Stellaris 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 453) 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.

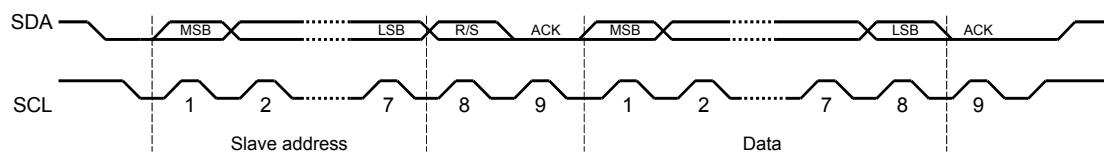
#### 13.3.1.1 START and STOP Conditions

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 13-3 on page 453.

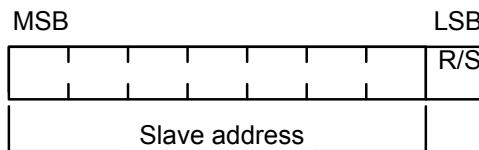
**Figure 13-3. START and STOP Conditions**

#### 13.3.1.2 Data Format with 7-Bit Address

Data transfers follow the format shown in Figure 13-4 on page 454. 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 I<sup>2</sup>CMSA 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.

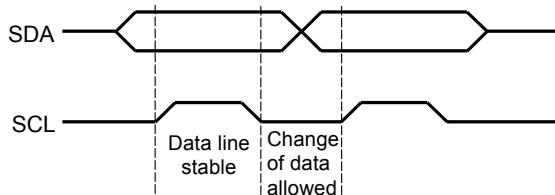
**Figure 13-4. Complete Data Transfer with a 7-Bit Address**

The first seven bits of the first byte make up the slave address (see Figure 13-5 on page 454). 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 13-5. R/S Bit in First Byte**

### 13.3.1.3 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 13-6 on page 454).

**Figure 13-6. Data Validity During Bit Transfer on the  $I^2C$  Bus**

### 13.3.1.4 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 454.

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 acknowledgement 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.

### 13.3.1.5 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.

### 13.3.2 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:

$$\text{SCL\_PERIOD} = 2 * (1 + \text{TIMER\_PRD}) * (\text{SCL\_LP} + \text{SCL\_HP}) * \text{CLK\_PRD}$$

For example:

CLK\_PRD = 50 ns

TIMER\_PRD = 2

SCL\_LP=6

SCL\_HP=4

yields a SCL frequency of:

1/T = 333 Khz

Table 13-3 on page 455 gives examples of timer period, system clock, and speed mode (Standard or Fast).

**Table 13-3. 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
33 MHz	0x10	97.1 Kbps	0x04	330 Kbps
40 MHz	0x13	100 Kbps	0x04	400 Kbps
50 MHz	0x18	100 Kbps	0x06	357 Kbps

### 13.3.3 Interrupts

The I<sup>2</sup>C can generate interrupts when the following conditions are observed:

- Master transaction completed
- Master arbitration lost
- Master transaction error
- Slave transaction received
- Slave transaction requested

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.

#### 13.3.3.1 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), when arbitration is lost, or when an error occurs during a transaction. To enable the I<sup>2</sup>C master interrupt, software must set the `IM` bit in the **I<sup>2</sup>C Master Interrupt Mask (I2CMIMR)** register. When an interrupt condition is met, software must check the `ERROR` and `ARBLST` bits in the **I<sup>2</sup>C Master Control/Status (I2CMCS)** register to verify that an error didn't occur during the last transaction and to ensure that arbitration has not been lost. An error condition is asserted if the last transaction wasn't acknowledged by the slave. If an error is not detected and the master has not lost arbitration, the application can proceed with the transfer. The interrupt is cleared by writing a 1 to the `IC` bit in 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.

#### 13.3.3.2 I<sup>2</sup>C Slave Interrupts

The slave module can generate an interrupt when data has been received or requested. This interrupt is enabled by writing a 1 to the `DATAIM` 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 `DATAIC` bit in 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.

#### 13.3.4 Loopback Operation

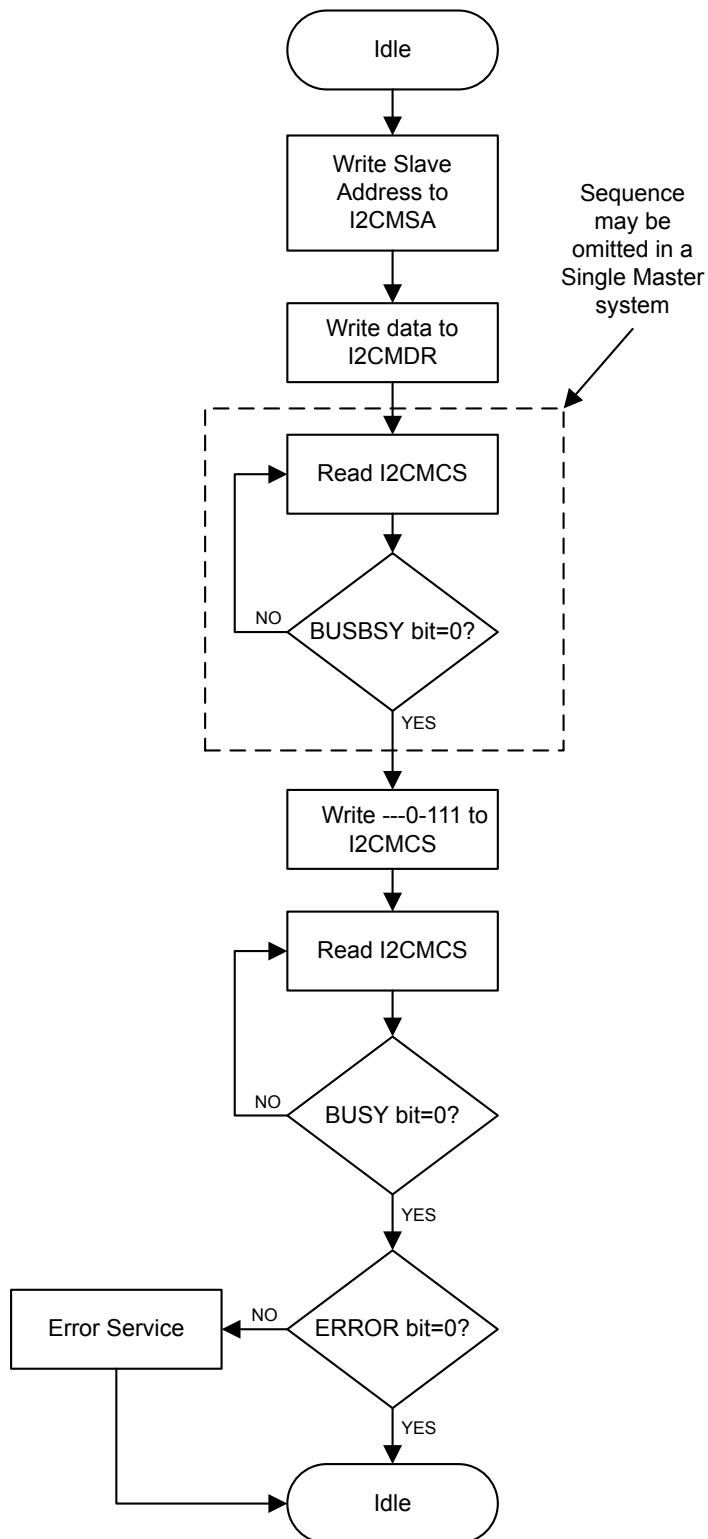
The I<sup>2</sup>C modules can be placed into an internal loopback mode for diagnostic or debug work. This is accomplished by setting the `LPBK` bit in the **I<sup>2</sup>C Master Configuration (I2CMCR)** register. In loopback mode, the SDA and SCL signals from the master and slave modules are tied together.

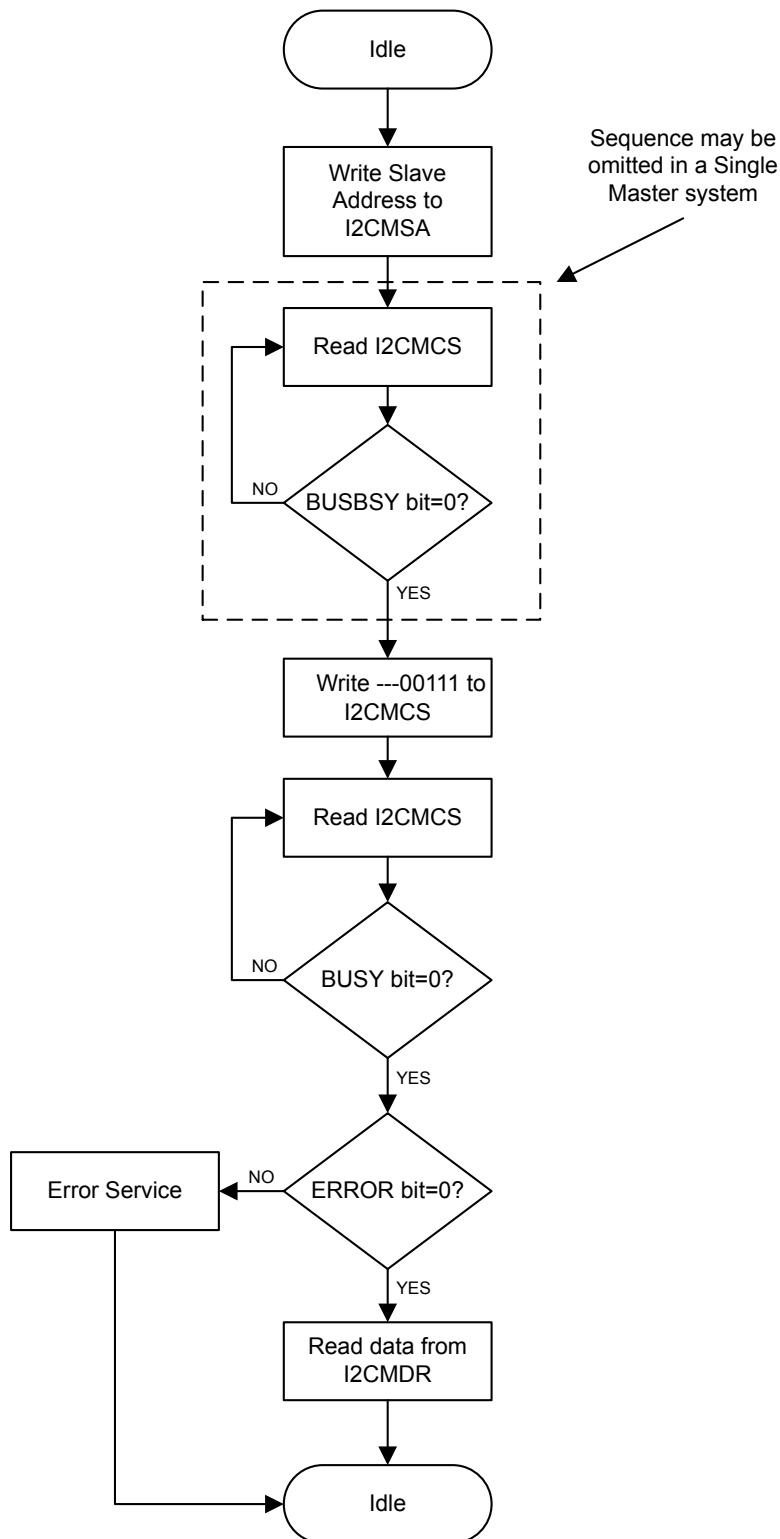
#### 13.3.5 Command Sequence Flow Charts

This section details the steps required to perform the various I<sup>2</sup>C transfer types in both master and slave mode.

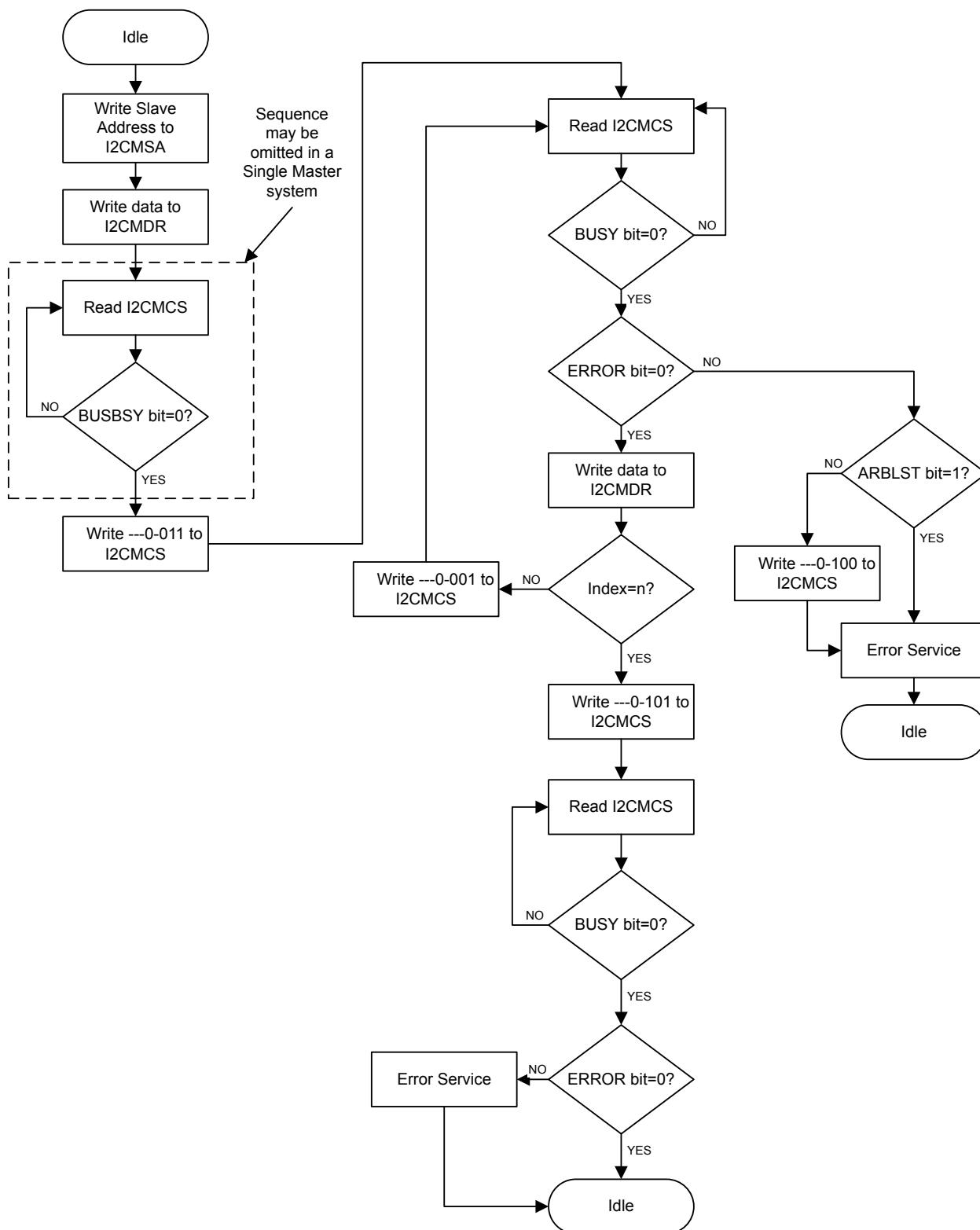
### 13.3.5.1 I<sup>2</sup>C Master Command Sequences

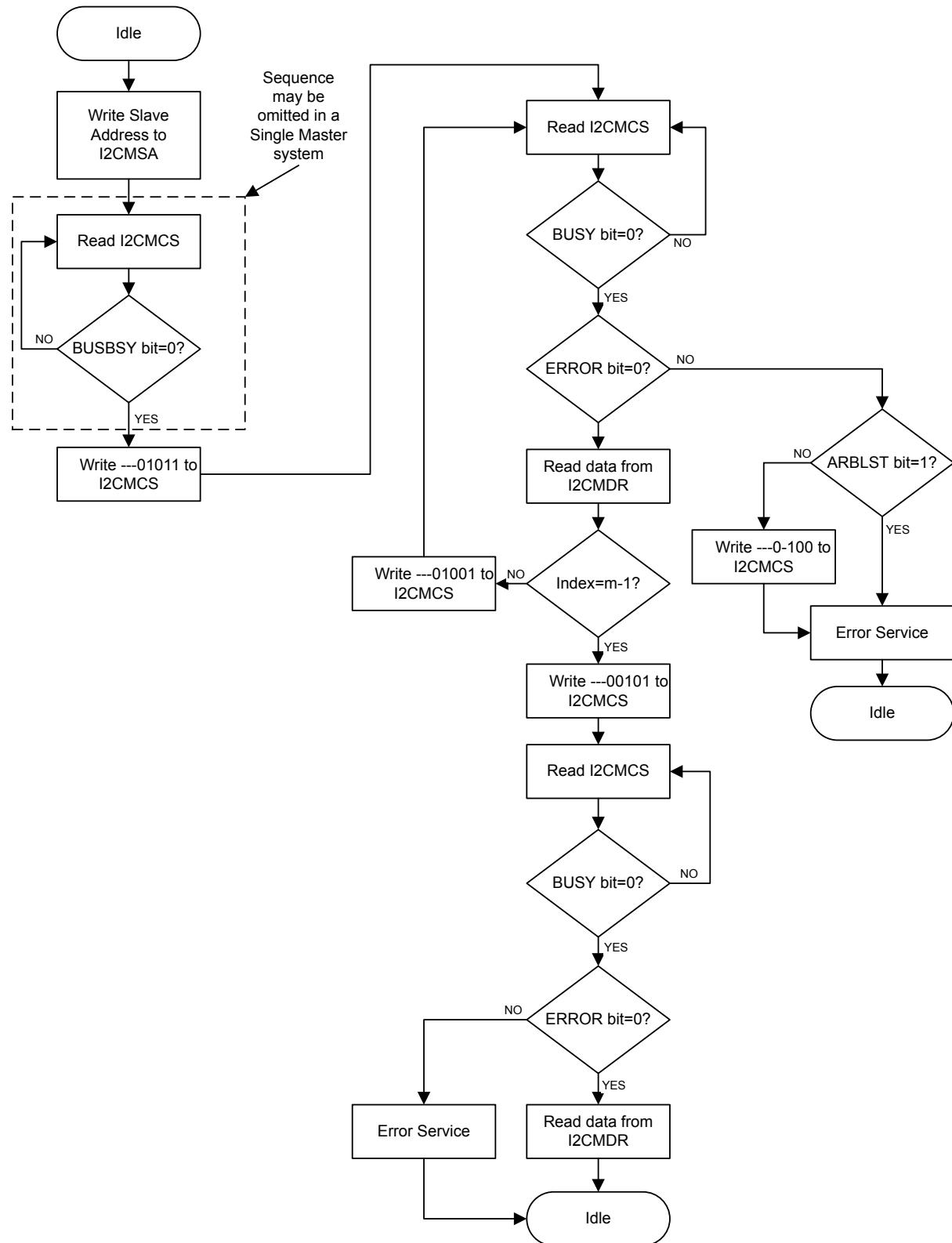
The figures that follow show the command sequences available for the I<sup>2</sup>C master.

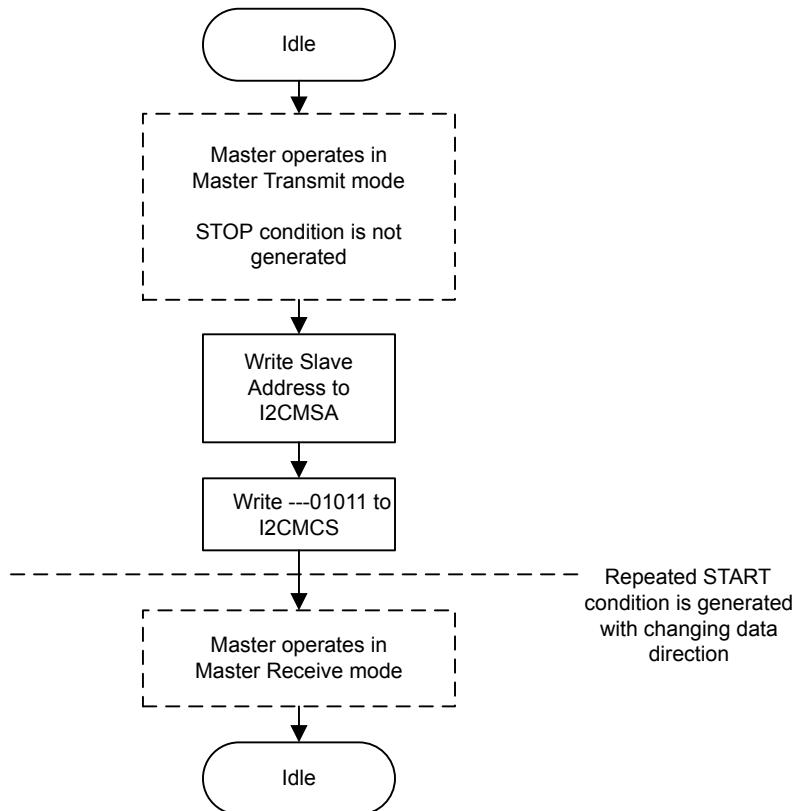
**Figure 13-7. Master Single SEND**

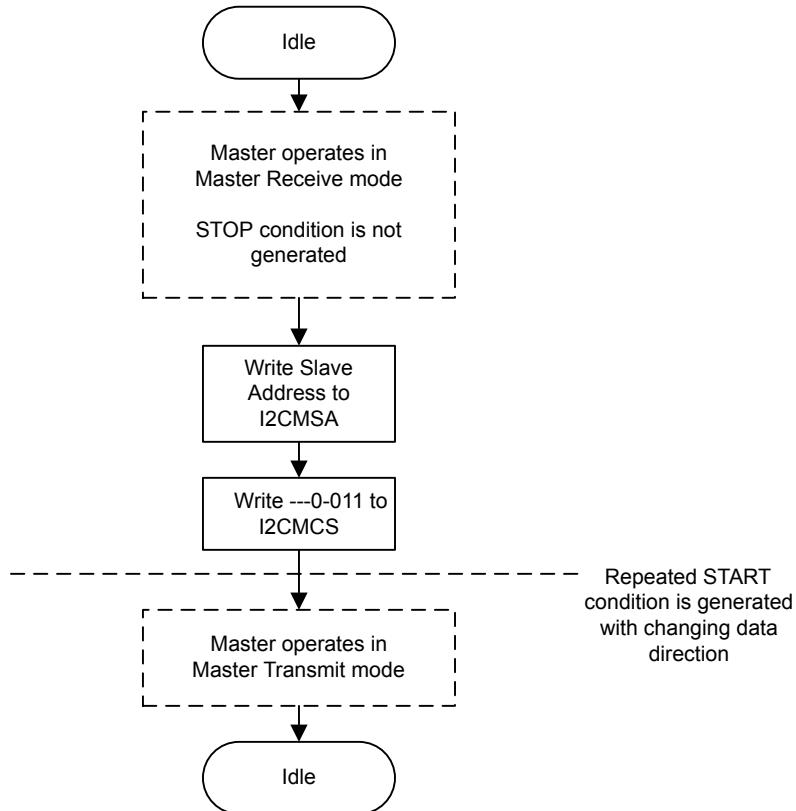
**Figure 13-8. Master Single RECEIVE**

**Figure 13-9. Master Burst SEND**



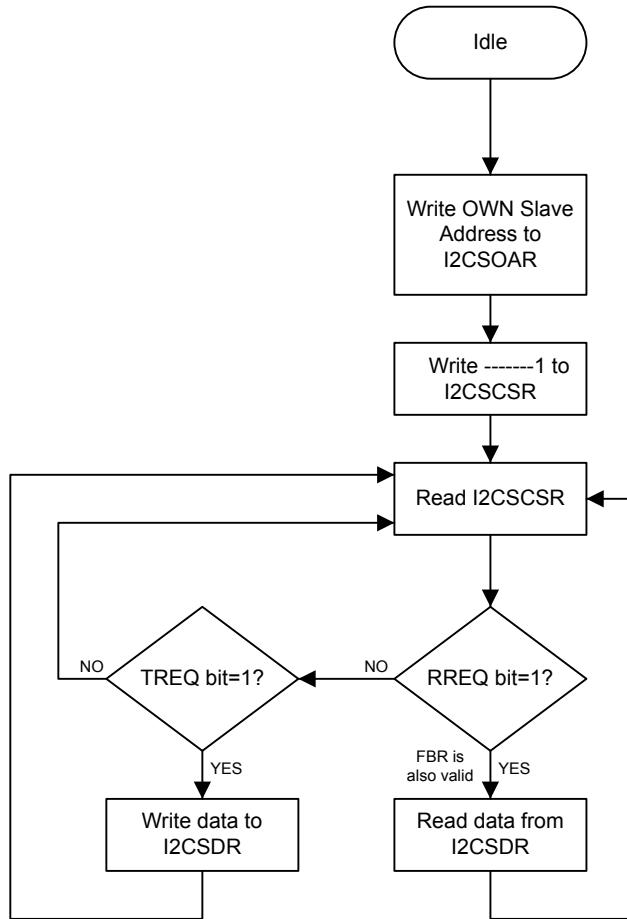
**Figure 13-10. Master Burst RECEIVE**

**Figure 13-11. Master Burst RECEIVE after Burst SEND**

**Figure 13-12. Master Burst SEND after Burst RECEIVE**

### 13.3.5.2 I<sup>2</sup>C Slave Command Sequences

Figure 13-13 on page 464 presents the command sequence available for the I<sup>2</sup>C slave.

**Figure 13-13. Slave Command Sequence**

## 13.4 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.

## 13.5 Register Map

Table 13-4 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 0: 0x4002.0000

Note that the I<sup>2</sup>C module clock must be enabled before the registers can be programmed (see page 199). There must be a delay of 3 system clocks after the I<sup>2</sup>C module clock is enabled before any I<sup>2</sup>C module registers are accessed.

The hw\_i2c.h file in the StellarisWare® Driver Library uses a base address of 0x800 for the I<sup>2</sup>C slave registers. Be aware when using registers with offsets between 0x800 and 0x818 that StellarisWare uses an offset between 0x000 and 0x018 with the slave base address.

**Table 13-4. 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	I2CMMIR	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

**Table 13-4. Inter-Integrated Circuit (I<sup>2</sup>C) Interface Register Map (continued)**

Offset	Name	Type	Reset	Description	See page
<b>I<sup>2</sup>C Slave</b>					
0x800	I2CSOAR	R/W	0x0000.0000	I2C Slave Own Address	480
0x804	I2CSCSR	RO	0x0000.0000	I2C Slave Control/Status	481
0x808	I2CSDR	R/W	0x0000.0000	I2C Slave Data	483
0x80C	I2CSIMR	R/W	0x0000.0000	I2C Slave Interrupt Mask	484
0x810	I2CSRIS	RO	0x0000.0000	I2C Slave Raw Interrupt Status	485
0x814	I2CSMIS	RO	0x0000.0000	I2C Slave Masked Interrupt Status	486
0x818	I2CSICR	WO	0x0000.0000	I2C Slave Interrupt Clear	487

## 13.6 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 (I<sup>2</sup>C 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 0 base: 0x4002.0000

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							
Type	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

### 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 (I2CMCSA)** 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 to 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.

### Reads

#### I2C Master Control/Status (I2CMCS)

I2C 0 base: 0x4002.0000

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
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	DATACK	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 or the transmit data not being acknowledged.
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 BUSY bit is set, the other status bits are not valid.

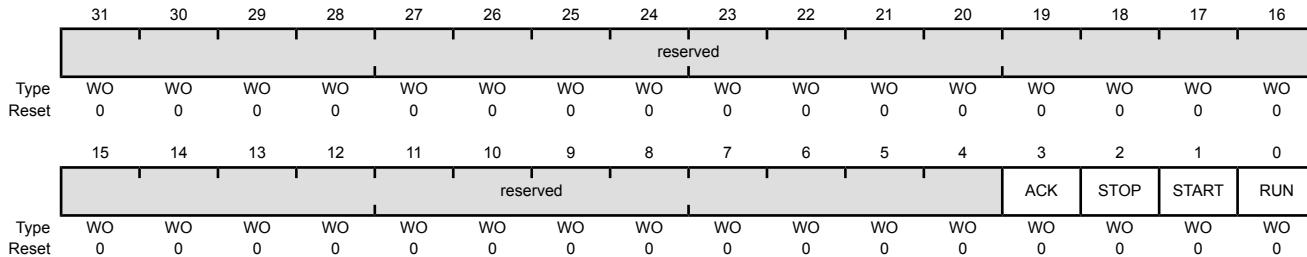
## Writes

### I2C Master Control/Status (I2CMCS)

I2C 0 base: 0x4002.0000

Offset 0x004

Type WO, reset 0x0000.0000



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 13-5 on page 470.
2	STOP	WO	0	Generate STOP When set, causes the generation of the STOP condition. See field decoding in Table 13-5 on page 470.
1	START	WO	0	Generate START When set, causes the generation of a START or repeated START condition. See field decoding in Table 13-5 on page 470.

Bit/Field	Name	Type	Reset	Description
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 13-5 on page 470.

**Table 13-5. Write Field Decoding for I2CMCS[3:0] Field (Sheet 1 of 3)**

Current State	I2CMCS[0]	I2CMCS[3:0]				Description
		R/S	ACK	STOP	START	
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.					NOP.	
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.					NOP.

**Table 13-5. Write Field Decoding for I2CMCS[3:0] Field (Sheet 1 of 3) (continued)**

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
		R/S	ACK	STOP	START	
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

**Important:** This register is read-sensitive. See the register description for details.

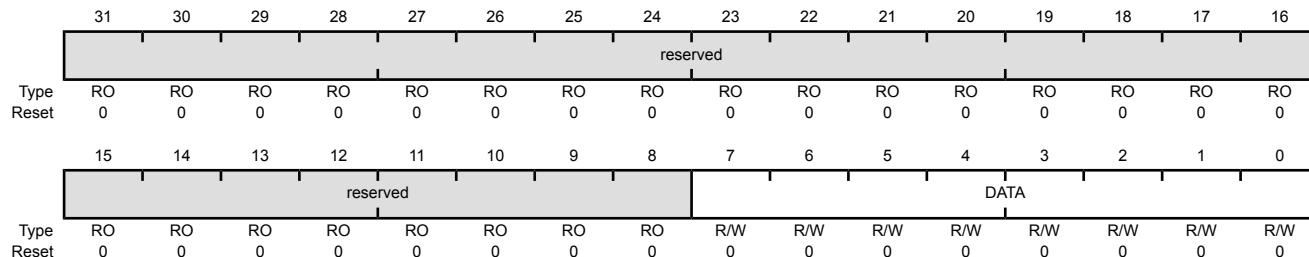
This register contains the data to be transmitted when in the Master Transmit state, and the data received when in the Master Receive state.

### I2C Master Data (I2CMDR)

I2C 0 base: 0x4002.0000

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.

**Caution – Take care not to set bit 7 when accessing this register as unpredictable behavior can occur.**

### I2C Master Timer Period (I2CMTPR)

I2C 0 base: 0x4002.0000

Offset 0x00C

Type R/W, reset 0x0000.0001

reserved															
Type	RO	RO	RO	RO	RO	RO									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved															
Type	RO	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	1
TPR															

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	TPR	R/W	0x1	<p>SCL Clock Period</p> <p>This field specifies the period of the SCL clock.</p> $\text{SCL\_PRD} = 2 * (1 + \text{TPR}) * (\text{SCL\_LP} + \text{SCL\_HP}) * \text{CLK\_PRD}$ <p>where:</p> <p>SCL_PRD is the SCL line period (I<sup>2</sup>C clock).</p> <p>TPR is the Timer Period register value (range of 1 to 127).</p> <p>SCL_LP is the SCL Low period (fixed at 6).</p> <p>SCL_HP is the SCL High period (fixed at 4).</p>

## 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 0 base: 0x4002.0000

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															IM
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

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

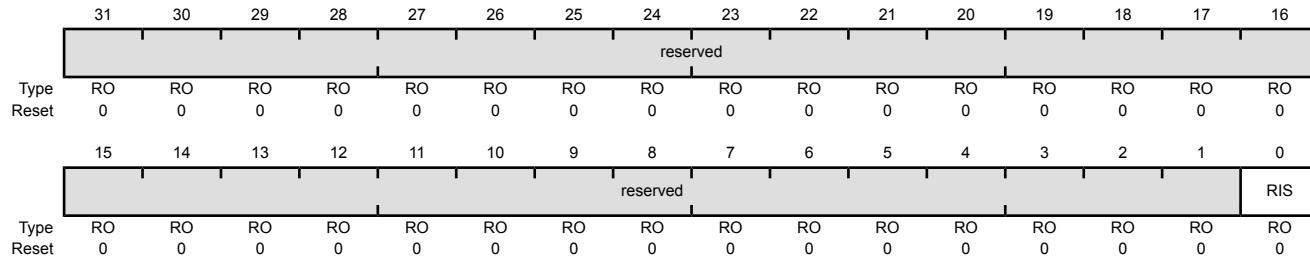
This register specifies whether an interrupt is pending.

### I2C Master Raw Interrupt Status (I2CMRIS)

I2C 0 base: 0x4002.0000

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	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 0 base: 0x4002.0000

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															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
31:1	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	<p>Masked Interrupt Status</p> <p>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.</p>

## 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 0 base: 0x4002.0000

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															IC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	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 0 base: 0x4002.0000

Offset 0x020

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										SFE	MFE	reserved			LPBK
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	RO	RO	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
31:6	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.

## 13.7 Register Descriptions (I<sup>2</sup>C 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 0x800

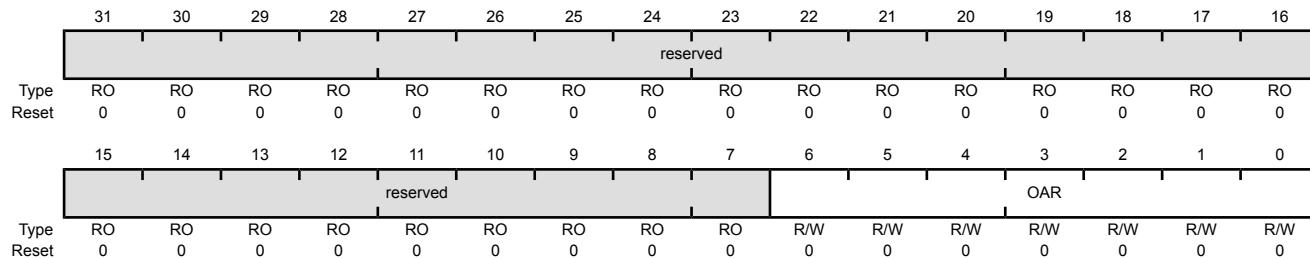
This register consists of seven address bits that identify the Stellaris I<sup>2</sup>C device on the I<sup>2</sup>C bus.

### I2C Slave Own Address (I2CSOAR)

I2C 0 base: 0x4002.0000

Offset 0x800

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 (I2CSCSR), offset 0x804

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 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 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 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 I<sup>2</sup>C slave operation.

### Reads

#### I<sup>2</sup>C Slave Control/Status (I2CSCSR)

I<sup>2</sup>C 0 base: 0x4002.0000

Offset 0x804

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
31:3	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	First Byte Received Indicates that the first byte following the slave's own address is received. This bit is only valid when the RREQ bit is set, and is automatically cleared when data has been read from the <b>I2CSDR</b> register.  <b>Note:</b> This bit is not used for slave transmit operations.
1	TREQ	RO	0	Transmit Request 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.
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 <b>I2CSDR</b> register. Otherwise, no receive data is outstanding.

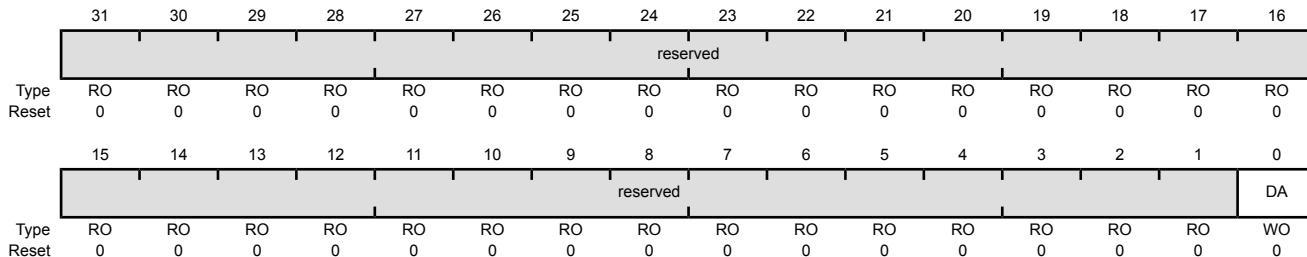
## Writes

### I<sup>2</sup>C Slave Control/Status (I<sup>2</sup>CSCCSR)

I<sup>2</sup>C 0 base: 0x4002.0000

Offset 0x804

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.	
				Once this bit has been set, it should not be set again unless it has been cleared by writing a 0 or by a reset, otherwise transfer failures may occur.

## Register 12: I<sup>2</sup>C Slave Data (I2CSDR), offset 0x808

**Important:** This register is read-sensitive. See the register description for details.

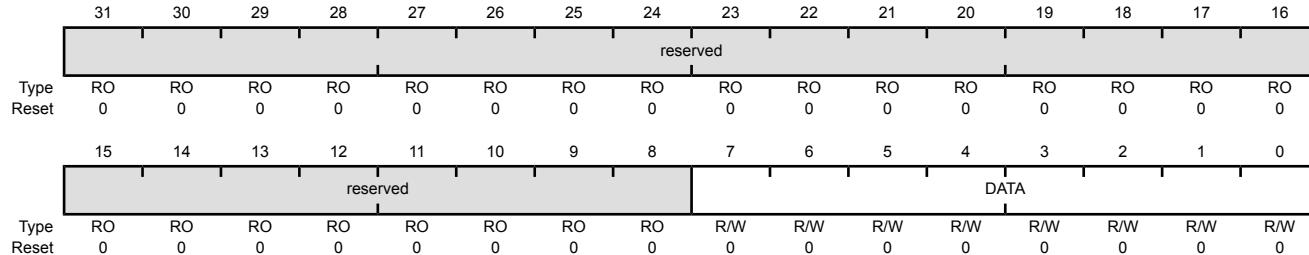
This register contains the data to be transmitted when in the Slave Transmit state, and the data received when in the Slave Receive state.

### I<sup>2</sup>C Slave Data (I2CSDR)

I<sup>2</sup>C 0 base: 0x4002.0000

Offset 0x808

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	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 0x80C

This register controls whether a raw interrupt is promoted to a controller interrupt.

### I<sup>2</sup>C Slave Interrupt Mask (I2CSIMR)

I<sup>2</sup>C 0 base: 0x4002.0000

Offset 0x80C

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															DATAIM
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

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	DATAIM	R/W	0	<p>Data Interrupt Mask</p> <p>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.</p>

## Register 14: I<sup>2</sup>C Slave Raw Interrupt Status (I2CSRIS), offset 0x810

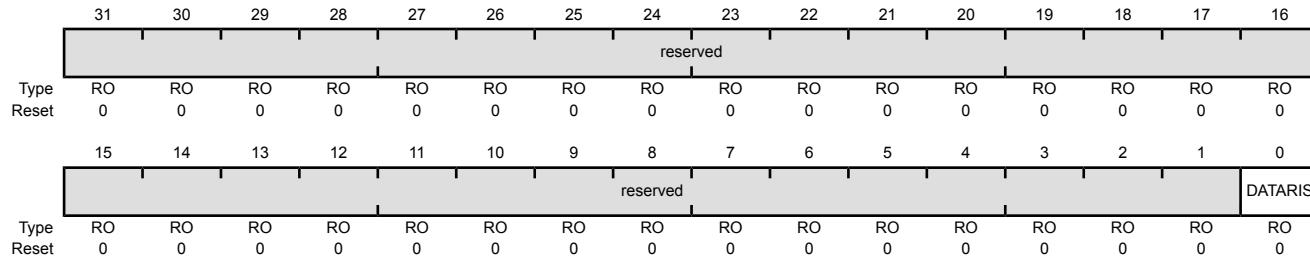
This register specifies whether an interrupt is pending.

### I2C Slave Raw Interrupt Status (I2CSRIS)

I2C 0 base: 0x4002.0000

Offset 0x810

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	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 0x814

This register specifies whether an interrupt was signaled.

### I2C Slave Masked Interrupt Status (I2CSMIS)

I2C 0 base: 0x4002.0000

Offset 0x814

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															DATAMIS
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	DATAMIS	RO	0	<p>Data Masked Interrupt Status</p> <p>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.</p>

## Register 16: I<sup>2</sup>C Slave Interrupt Clear (I2CSICR), offset 0x818

This register clears the raw interrupt. A read of this register returns no meaningful data.

### I<sup>2</sup>C Slave Interrupt Clear (I2CSICR)

I<sup>2</sup>C 0 base: 0x4002.0000

Offset 0x818

Type WO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved																
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	RO	WO														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	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	DATAIC	WO	0	<p>Data Interrupt Clear</p> <p>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.</p>

## 14 Analog Comparator

An analog comparator is a peripheral that compares two analog voltages, and provides a logical output that signals the comparison result.

**Note:** Not all comparators have the option to drive an output pin. See the Comparator Operating Mode tables in “Functional Description” on page 489 for more information.

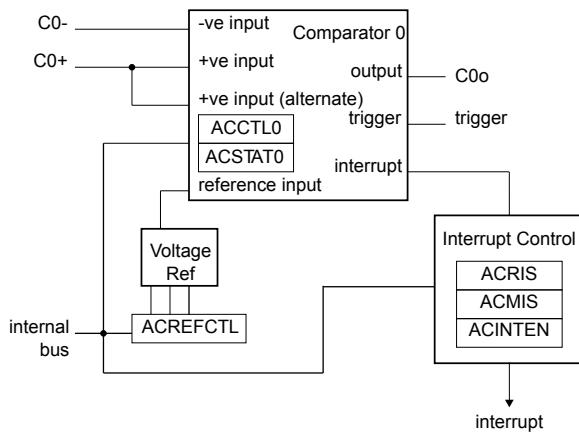
The comparator can provide its output to a device pin, acting as a replacement for an analog comparator on the board, or it can be used to signal the application via interrupts or triggers to the ADC to cause it to start capturing a sample sequence. The interrupt generation and ADC triggering logic is separate. This means, for example, that an interrupt can be generated on a rising edge and the ADC triggered on a falling edge.

The Stellaris® Analog Comparators module has the following features:

- One integrated analog comparator
- Configurable for output to drive an output pin, generate an interrupt, or initiate an ADC sample sequence
- Compare external pin input to external pin input or to internal programmable voltage reference
- Compare a test voltage against any one of these voltages
  - An individual external reference voltage
  - A shared single external reference voltage
  - A shared internal reference voltage

### 14.1 Block Diagram

**Figure 14-1. Analog Comparator Module Block Diagram**



### 14.2 Signal Description

Table 14-1 on page 489 and Table 14-2 on page 489 list the external signals of the Analog Comparators and describe the function of each. The Analog Comparator output signal is an alternate functions for a GPIO signal and default to be a GPIO signal at reset. The column in the table below titled "Pin

"Assignment" lists the possible GPIO pin placements for the Analog Comparator signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 253) should be set to choose the Analog Comparator function. The positive and negative input signal are configured by clearing the DEN bit in the **GPIO Digital Enable (GPIODEN)** register. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 232.

**Table 14-1. Analog Comparators Signals (48QFP)**

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
C0+	42	I	Analog	Analog comparator 0 positive input.
C0-	44	I	Analog	Analog comparator 0 negative input.
C0o	48	O	TTL	Analog comparator 0 output.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

**Table 14-2. Analog Comparators Signals (48QFN)**

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
C0+	42	I	Analog	Analog comparator 0 positive input.
C0-	44	I	Analog	Analog comparator 0 negative input.
C0o	48	O	TTL	Analog comparator 0 output.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

## 14.3 Functional Description

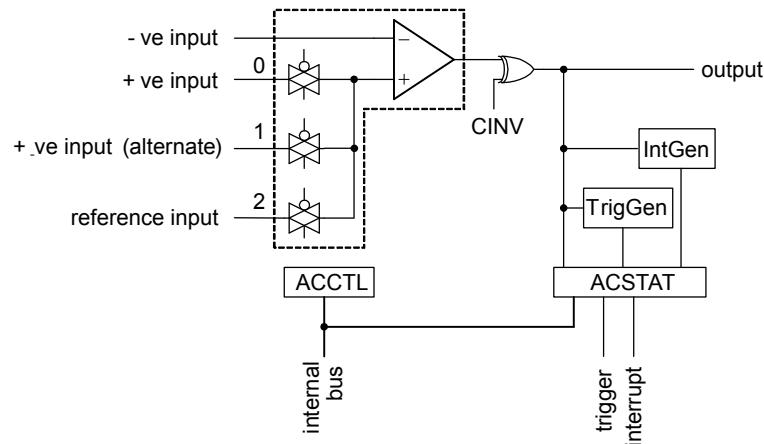
**Important:** It is recommended that the Digital-Input enable (the GPIODEN bit in the GPIO module) for the analog input pin be disabled to prevent excessive current draw from the I/O pads.

The comparator compares the VIN- and VIN+ inputs to produce an output, VOUT.

$$\begin{aligned} \text{VIN-} < \text{VIN+}, \text{ VOUT} &= 1 \\ \text{VIN-} > \text{VIN+}, \text{ VOUT} &= 0 \end{aligned}$$

As shown in Figure 14-2 on page 489, the input source for VIN- is an external input. In addition to an external input, input sources for VIN+ can be the +ve input of comparator 0 or an internal reference.

**Figure 14-2. Structure of Comparator Unit**



A comparator is configured through two status/control registers (**ACCTL** and **ACSTAT**). The internal reference is configured through one control register (**ACREFCTL**). Interrupt status and control is configured through three registers (**ACMIS**, **ACRIS**, and **ACINTEN**). The operating modes of the comparators are shown in the Comparator Operating Mode tables.

Typically, the comparator output is used internally to generate controller interrupts. It may also be used to drive an external pin or generate an analog-to-digital converter (ADC) trigger.

**Important:** The ASRCP bits in the **ACCTLn** register must be set before using the analog comparators. The proper pad configuration for the comparator input and output pins are described in the Comparator Operating Mode tables.

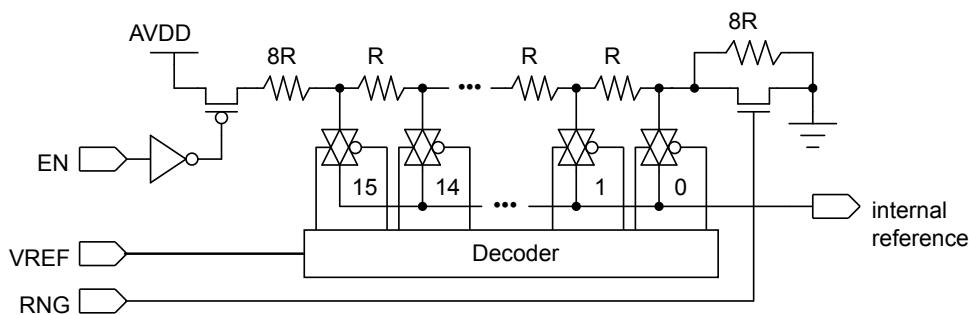
**Table 14-3. Comparator 0 Operating Modes**

ACCTL0	Comparator 0				
ASRCP	VIN-	VIN+	Output	Interrupt	ADC Trigger
00	C0-	C0+	C0o	yes	yes
01	C0-	C0+	C0o	yes	yes
10	C0-	Vref	C0o	yes	yes
11	C0-	reserved	C0o	yes	yes

### 14.3.1 Internal Reference Programming

The structure of the internal reference is shown in Figure 14-3 on page 490. This is controlled by a single configuration register (**ACREFCTL**). Table 14-4 on page 490 shows the programming options to develop specific internal reference values, to compare an external voltage against a particular voltage generated internally.

**Figure 14-3. Comparator Internal Reference Structure**



**Table 14-4. Internal Reference Voltage and ACREFCTL Field Values**

ACREFCTL Register		Output Reference Voltage Based on VREF Field Value
EN Bit Value	RNG Bit Value	
EN=0	RNG=X	0 V (GND) for any value of VREF; however, it is recommended that RNG=1 and VREF=0 for the least noisy ground reference.

**Table 14-4. Internal Reference Voltage and ACREFCTL Field Values (continued)**

ACREFCTL Register		Output Reference Voltage Based on VREF Field Value
EN Bit Value	RNG Bit Value	
EN=1	RNG=0	<p>Total resistance in ladder is 31 R.</p> $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{(VREF + 8)}{31}$ $V_{REF} = 0.85 + 0.106 \times VREF$ <p>The range of internal reference in this mode is 0.85-2.448 V.</p>
	RNG=1	<p>Total resistance in ladder is 23 R.</p> $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{VREF}{23}$ $V_{REF} = 0.143 \times VREF$ <p>The range of internal reference for this mode is 0-2.152 V.</p>

## 14.4 Initialization and Configuration

The following example shows how to configure an analog comparator to read back its output value from an internal register.

1. Enable the analog comparator 0 clock by writing a value of 0x0010.0000 to the **RCGC1** register in the System Control module.
2. In the GPIO module, enable the GPIO port/pin associated with C0– as a GPIO input.
3. Configure the internal voltage reference to 1.65 V by writing the **ACREFCTL** register with the value 0x0000.030C.
4. Configure comparator 0 to use the internal voltage reference and to *not* invert the output by writing the **ACCTL0** register with the value 0x0000.040C.
5. Delay for some time.
6. Read the comparator output value by reading the **ACSTAT0** register's OVAL value.

Change the level of the signal input on C0– to see the OVAL value change.

## 14.5 Register Map

Table 14-5 on page 492 lists the comparator registers. The offset listed is a hexadecimal increment to the register's address, relative to the Analog Comparator base address of 0x4003.C000.

Note that the analog comparator module clock must be enabled before the registers can be programmed (see page 199). There must be a delay of 3 system clocks after the ADC module clock is enabled before any ADC module registers are accessed.

**Table 14-5. Analog Comparators Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	ACMIS	R/W1C	0x0000.0000	Analog Comparator Masked Interrupt Status	493
0x004	ACRIS	RO	0x0000.0000	Analog Comparator Raw Interrupt Status	494
0x008	ACINTEN	R/W	0x0000.0000	Analog Comparator Interrupt Enable	495
0x010	ACREFCTL	R/W	0x0000.0000	Analog Comparator Reference Voltage Control	496
0x020	ACSTAT0	RO	0x0000.0000	Analog Comparator Status 0	497
0x024	ACCTL0	R/W	0x0000.0000	Analog Comparator Control 0	498

## 14.6 Register Descriptions

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

## Register 1: Analog Comparator Masked Interrupt Status (ACMIS), offset 0x000

This register provides a summary of the interrupt status (masked) of the comparator.

### Analog Comparator Masked Interrupt Status (ACMIS)

Base 0x4003.C000

Offset 0x000

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															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	RO	R/W1C														
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
																IN0
Type	RO	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
31:1	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	IN0	R/W1C	0	Comparator 0 Masked Interrupt Status Gives the masked interrupt state of this interrupt. Write 1 to this bit to clear the pending interrupt.

## Register 2: Analog Comparator Raw Interrupt Status (ACRIS), offset 0x004

This register provides a summary of the interrupt status (raw) of the comparator.

### Analog Comparator Raw Interrupt Status (ACRIS)

Base 0x4003.C000  
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															IN0
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	IN0	RO	0	Comparator 0 Interrupt Status When set, indicates that an interrupt has been generated by comparator 0.

## Register 3: Analog Comparator Interrupt Enable (ACINTEN), offset 0x008

This register provides the interrupt enable for the comparator.

### Analog Comparator Interrupt Enable (ACINTEN)

Base 0x4003.C000  
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															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	RO	R/W														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

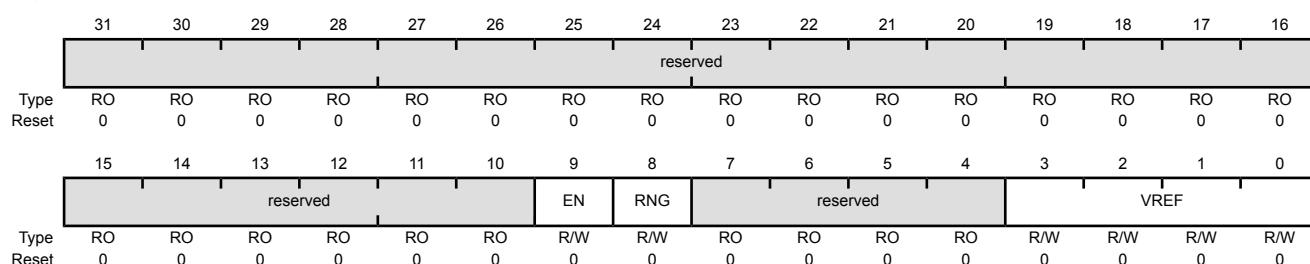
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	IN0	R/W	0	Comparator 0 Interrupt Enable When set, enables the controller interrupt from the comparator 0 output.

## Register 4: Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x010

This register specifies whether the resistor ladder is powered on as well as the range and tap.

### Analog Comparator Reference Voltage Control (ACREFCTL)

Base 0x4003.C000  
Offset 0x010  
Type R/W, reset 0x0000.0000



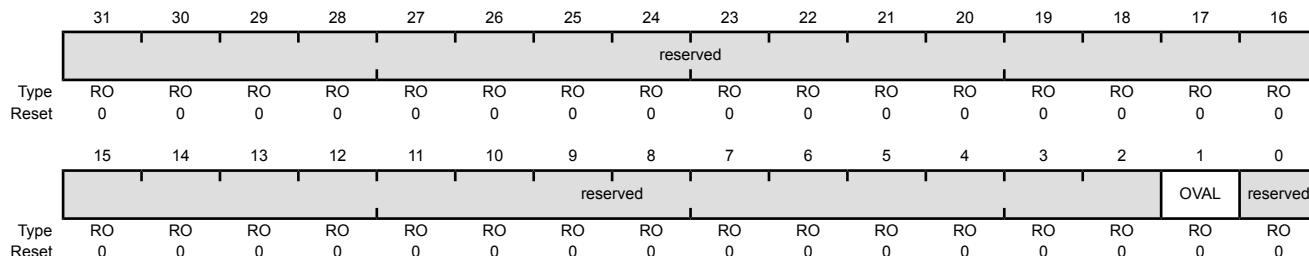
Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	EN	R/W	0	Resistor Ladder Enable The EN bit specifies whether the resistor ladder is powered on. If 0, the resistor ladder is unpowered. If 1, the resistor ladder is connected to the analog V <sub>DD</sub> . This bit is reset to 0 so that the internal reference consumes the least amount of power if not used and programmed.
8	RNG	R/W	0	Resistor Ladder Range The RNG bit specifies the range of the resistor ladder. If 0, the resistor ladder has a total resistance of 31 R. If 1, the resistor ladder has a total resistance of 23 R.
7:4	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	VREF	R/W	0x00	Resistor Ladder Voltage Ref The VREF bit field specifies the resistor ladder tap that is passed through an analog multiplexer. The voltage corresponding to the tap position is the internal reference voltage available for comparison. See Table 14-4 on page 490 for some output reference voltage examples.

## Register 5: Analog Comparator Status 0 (ACSTAT0), offset 0x020

This register specifies the current output value of the comparator.

### Analog Comparator Status 0 (ACSTAT0)

Base 0x4003.C000  
Offset 0x020  
Type RO, 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	OVAL	RO	0	Comparator Output Value The OVAL bit specifies the current output value of the comparator.
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: Analog Comparator Control 0 (ACCTL0), offset 0x024

This register configures the comparator's input and output.

### Analog Comparator Control 0 (ACCTL0)

Base 0x4003.C000  
Offset 0x024  
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
reserved																
Type	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	RO	RO	RO	RO	R/W	R/W	R/W	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
31:12	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.
11	TOEN	R/W	0	Trigger Output Enable The TOEN bit enables the ADC event transmission to the ADC. If 0, the event is suppressed and not sent to the ADC. If 1, the event is transmitted to the ADC.
10:9	ASRCP	R/W	0x00	Analog Source Positive The ASRCP field specifies the source of input voltage to the VIN+ terminal of the comparator. The encodings for this field are as follows:
				Value Function
				0x0 Pin value
				0x1 Pin value of C0+
				0x2 Internal voltage reference
				0x3 Reserved
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	TSLVAL	R/W	0	Trigger Sense Level Value The TSLVAL bit specifies the sense value of the input that generates an ADC event if in Level Sense mode. If 0, an ADC event is generated if the comparator output is Low. Otherwise, an ADC event is generated if the comparator output is High.

Bit/Field	Name	Type	Reset	Description										
6:5	TSEN	R/W	0x0	<p><b>Trigger Sense</b></p> <p>The TSEN field specifies the sense of the comparator output that generates an ADC event. The sense conditioning is as follows:</p> <table> <thead> <tr> <th>Value</th><th>Function</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Level sense, see ISLVAL</td></tr> <tr> <td>0x1</td><td>Falling edge</td></tr> <tr> <td>0x2</td><td>Rising edge</td></tr> <tr> <td>0x3</td><td>Either edge</td></tr> </tbody> </table>	Value	Function	0x0	Level sense, see ISLVAL	0x1	Falling edge	0x2	Rising edge	0x3	Either edge
Value	Function													
0x0	Level sense, see ISLVAL													
0x1	Falling edge													
0x2	Rising edge													
0x3	Either edge													
4	ISLVAL	R/W	0	<p><b>Interrupt Sense Level Value</b></p> <p>The ISLVAL bit specifies the sense value of the input that generates an interrupt if in Level Sense mode. If 0, an interrupt is generated if the comparator output is Low. Otherwise, an interrupt is generated if the comparator output is High.</p>										
3:2	ISEN	R/W	0x0	<p><b>Interrupt Sense</b></p> <p>The ISEN field specifies the sense of the comparator output that generates an interrupt. The sense conditioning is as follows:</p> <table> <thead> <tr> <th>Value</th><th>Function</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Level sense, see ISLVAL</td></tr> <tr> <td>0x1</td><td>Falling edge</td></tr> <tr> <td>0x2</td><td>Rising edge</td></tr> <tr> <td>0x3</td><td>Either edge</td></tr> </tbody> </table>	Value	Function	0x0	Level sense, see ISLVAL	0x1	Falling edge	0x2	Rising edge	0x3	Either edge
Value	Function													
0x0	Level sense, see ISLVAL													
0x1	Falling edge													
0x2	Rising edge													
0x3	Either edge													
1	CINV	R/W	0	<p><b>Comparator Output Invert</b></p> <p>The CINV bit conditionally inverts the output of the comparator. If 0, the output of the comparator is unchanged. If 1, the output of the comparator is inverted prior to being processed by hardware.</p>										
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.										

## 15 Pulse Width Modulator (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.

The Stellaris® PWM module consists of three PWM generator blocks and a control block. The control block determines the polarity of the PWM signals, and which signals are passed through to the pins.

Each PWM generator block produces two PWM signals that can either be independent signals (other than being based on the same timer and therefore having the same frequency) or a single pair of complementary signals with dead-band delays inserted. The output of the PWM generation blocks are managed by the output control block before being passed to the device pins.

The Stellaris PWM module provides a great deal of flexibility. It can generate simple PWM signals, such as those required by a simple charge pump. It can also generate paired PWM signals with dead-band delays, such as those required by a half-H bridge driver. Three generator blocks can also generate the full six channels of gate controls required by a 3-phase inverter bridge.

Each Stellaris PWM module has the following features:

- Three PWM generator blocks, each with one 16-bit counter, two PWM comparators, a PWM signal generator, a dead-band generator, and an interrupt/ADC-trigger selector
- One fault input in hardware to promote low-latency shutdown
- One 16-bit counter
  - Runs in Down or Up/Down mode
  - Output frequency controlled by a 16-bit load value
  - Load value updates can be synchronized
  - Produces output signals at zero and load value
- Two PWM comparators
  - Comparator value updates can be synchronized
  - Produces output signals on match
- PWM generator
  - Output PWM signal is constructed based on actions taken as a result of the counter and PWM comparator output signals
  - Produces two independent PWM signals
- Dead-band generator
  - Produces two PWM signals with programmable dead-band delays suitable for driving a half-H bridge
  - Can be bypassed, leaving input PWM signals unmodified

- Flexible output control block with PWM output enable of each PWM signal
  - PWM output enable of each PWM signal
  - Optional output inversion of each PWM signal (polarity control)
  - Optional fault handling for each PWM signal
  - Synchronization of timers in the PWM generator blocks
  - Interrupt status summary of the PWM generator blocks
- Can initiate an ADC sample sequence

## 15.1 Block Diagram

Figure 15-1 on page 501 provides the Stellaris PWM module unit diagram and Figure 15-2 on page 502 provides a more detailed diagram of a Stellaris PWM generator. The LM3S811 controller contains three generator blocks (PWM0, PWM1, and PWM2) and generates six independent PWM signals or three paired PWM signals with dead-band delays inserted.

**Figure 15-1. PWM Unit Diagram**

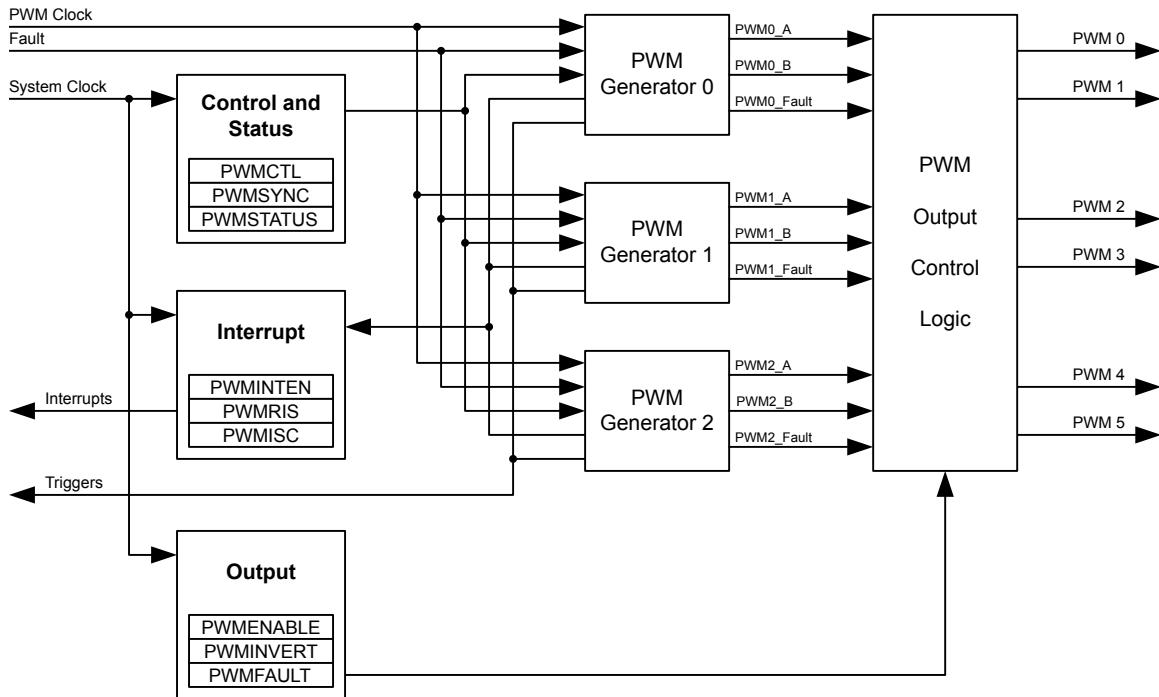
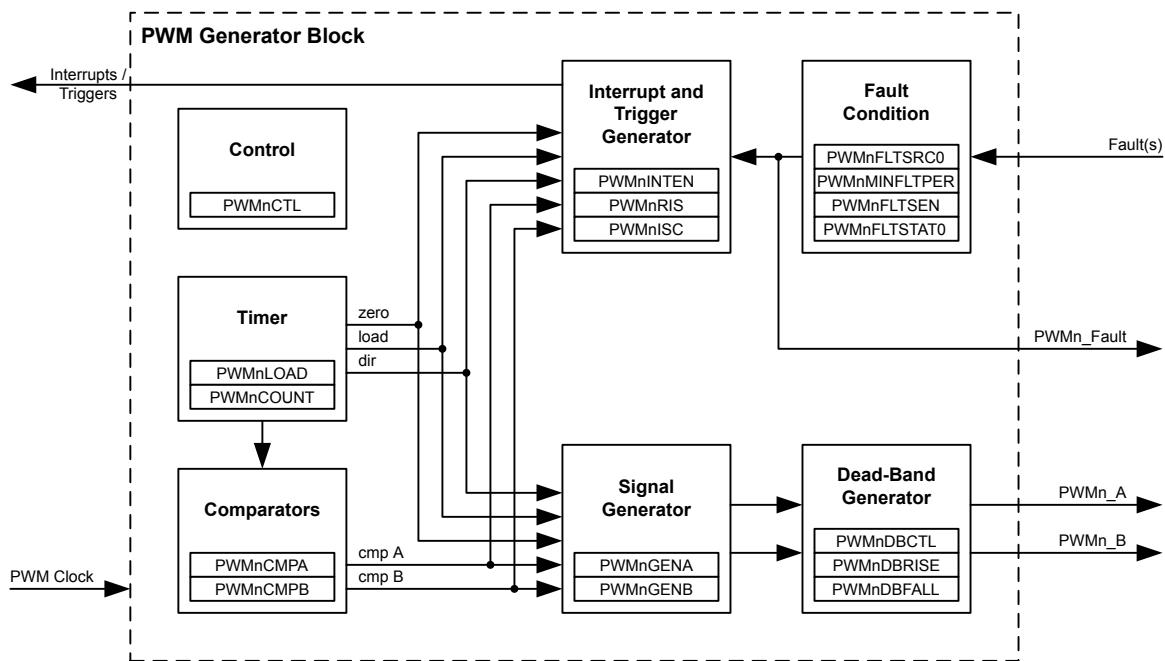


Figure 15-2. PWM Module Block Diagram



## 15.2 Signal Description

Table 15-1 on page 502 and Table 15-2 on page 502 list the external signals of the PWM module and describe the function of each. The PWM controller signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Assignment" lists the possible GPIO pin placements for these PWM signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 253) should be set to choose the PWM function. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 232.

Table 15-1. PWM Signals (48QFP)

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
Fault	47	I	TTL	PWM Fault.
PWM0	25	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
PWM1	26	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.
PWM2	29	O	TTL	PWM 2. This signal is controlled by PWM Generator 1.
PWM3	30	O	TTL	PWM 3. This signal is controlled by PWM Generator 1.
PWM4	35	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
PWM5	36	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

Table 15-2. PWM Signals (48QFN)

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
Fault	47	I	TTL	PWM Fault.
PWM0	25	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
PWM1	26	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.

**Table 15-2. PWM Signals (48QFN) (continued)**

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
PWM2	29	O	TTL	PWM 2. This signal is controlled by PWM Generator 1.
PWM3	30	O	TTL	PWM 3. This signal is controlled by PWM Generator 1.
PWM4	35	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
PWM5	36	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

## 15.3 Functional Description

### 15.3.1 PWM Timer

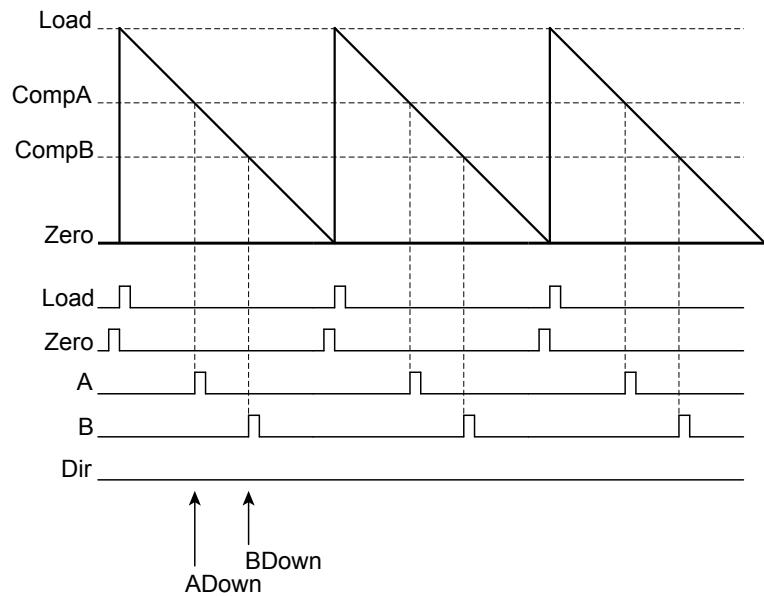
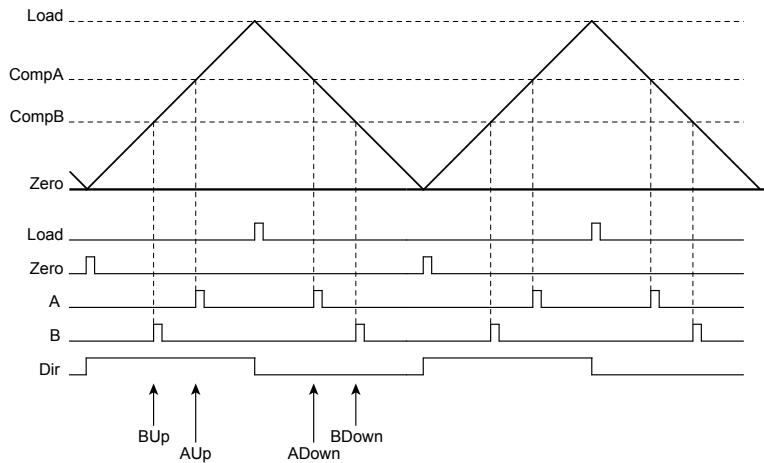
The timer in each PWM generator runs in one of two modes: Count-Down mode or Count-Up/Down mode. In Count-Down mode, the timer counts from the load value to zero, goes back to the load value, and continues counting down. In Count-Up/Down mode, the timer counts from zero up to the load value, back down to zero, back up to the load value, and so on. Generally, Count-Down mode is used for generating left- or right-aligned PWM signals, while the Count-Up/Down mode is used for generating center-aligned PWM signals.

The timers output three signals that are used in the PWM generation process: the direction signal (this is always Low in Count-Down mode, but alternates between Low and High in Count-Up/Down mode), a single-clock-cycle-width High pulse when the counter is zero, and a single-clock-cycle-width High pulse when the counter is equal to the load value. Note that in Count-Down mode, the zero pulse is immediately followed by the load pulse.

### 15.3.2 PWM Comparators

There are two comparators in each PWM generator that monitor the value of the counter; when either match the counter, they output a single-clock-cycle-width High pulse. When in Count-Up/Down mode, these comparators match both when counting up and when counting down; they are therefore qualified by the counter direction signal. These qualified pulses are used in the PWM generation process. If either comparator match value is greater than the counter load value, then that comparator never outputs a High pulse.

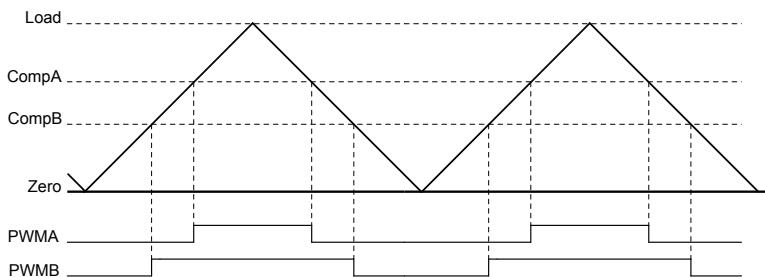
Figure 15-3 on page 504 shows the behavior of the counter and the relationship of these pulses when the counter is in Count-Down mode. Figure 15-4 on page 504 shows the behavior of the counter and the relationship of these pulses when the counter is in Count-Up/Down mode.

**Figure 15-3. PWM Count-Down Mode****Figure 15-4. PWM Count-Up/Down Mode**

### 15.3.3 PWM Signal Generator

The PWM generator takes these pulses (qualified by the direction signal), and generates two PWM signals. In Count-Down mode, there are four events that can affect the PWM signal: zero, load, match A down, and match B down. In Count-Up/Down mode, there are six events that can affect the PWM signal: zero, load, match A down, match A up, match B down, and match B up. The match A or match B events are ignored when they coincide with the zero or load events. If the match A and match B events coincide, the first signal, PWMA, is generated based only on the match A event, and the second signal, PWMB, is generated based only on the match B event.

For each event, the effect on each output PWM signal is programmable: it can be left alone (ignoring the event), it can be toggled, it can be driven Low, or it can be driven High. These actions can be used to generate a pair of PWM signals of various positions and duty cycles, which do or do not overlap. Figure 15-5 on page 505 shows the use of Count-Up/Down mode to generate a pair of center-aligned, overlapped PWM signals that have different duty cycles.

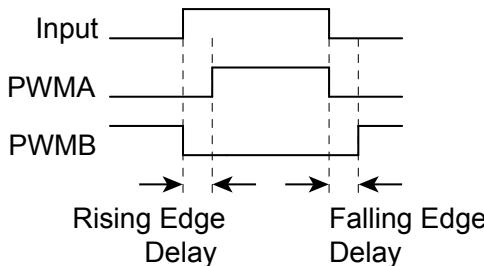
**Figure 15-5. PWM Generation Example In Count-Up/Down Mode**

In this example, the first generator is set to drive High on match A up, drive Low on match A down, and ignore the other four events. The second generator is set to drive High on match B up, drive Low on match B down, and ignore the other four events. Changing the value of comparator A changes the duty cycle of the PWMA signal, and changing the value of comparator B changes the duty cycle of the PWMB signal.

#### 15.3.4 Dead-Band Generator

The two PWM signals produced by the PWM generator are passed to the dead-band generator. If disabled, the PWM signals simply pass through unmodified. If enabled, the second PWM signal is lost and two PWM signals are generated based on the first PWM signal. The first output PWM signal is the input signal with the rising edge delayed by a programmable amount. The second output PWM signal is the inversion of the input signal with a programmable delay added between the falling edge of the input signal and the rising edge of this new signal.

This is therefore a pair of active High signals where one is always High, except for a programmable amount of time at transitions where both are Low. These signals are therefore suitable for driving a half-H bridge, with the dead-band delays preventing shoot-through current from damaging the power electronics. Figure 15-6 on page 505 shows the effect of the dead-band generator on an input PWM signal.

**Figure 15-6. PWM Dead-Band Generator**

#### 15.3.5 Interrupt/ADC-Trigger Selector

The PWM generator also takes the same four (or six) counter events and uses them to generate an interrupt or an ADC trigger. Any of these events or a set of these events can be selected as a source for an interrupt; when any of the selected events occur, an interrupt is generated. Additionally, the same event, a different event, the same set of events, or a different set of events can be selected as a source for an ADC trigger; when any of these selected events occur, an ADC trigger pulse is generated. The selection of events allows the interrupt or ADC trigger to occur at a specific position within the PWM signal. Note that interrupts and ADC triggers are based on the raw events; delays in the PWM signal edges caused by the dead-band generator are not taken into account.

### 15.3.6 Synchronization Methods

There is a global reset capability that can synchronously reset any or all of the counters in the PWM generators. If multiple PWM generators are configured with the same counter load value, this can be used to guarantee that they also have the same count value (this does imply that the PWM generators must be configured before they are synchronized). With this, more than two PWM signals can be produced with a known relationship between the edges of those signals since the counters always have the same values.

The counter load values and comparator match values of the PWM generator can be updated in two ways. The first is immediate update mode, where a new value is used as soon as the counter reaches zero. By waiting for the counter to reach zero, a guaranteed behavior is defined, and overly short or overly long output PWM pulses are prevented.

The other update method is synchronous, where the new value is not used until a global synchronized update signal is asserted, at which point the new value is used as soon as the counter reaches zero. This second mode allows multiple items in multiple PWM generators to be updated simultaneously without odd effects during the update; everything runs from the old values until a point at which they all run from the new values. The Update mode of the load and comparator match values can be individually configured in each PWM generator block. It typically makes sense to use the synchronous update mechanism across PWM generator blocks when the timers in those blocks are synchronized, though this is not required in order for this mechanism to function properly.

### 15.3.7 Fault Conditions

There are two external conditions that affect the PWM block; the signal input on the Fault pin and the stalling of the controller by a debugger. There are two mechanisms available to handle such conditions: the output signals can be forced into an inactive state and/or the PWM timers can be stopped.

Each output signal has a fault bit. If set, a fault input signal causes the corresponding output signal to go into the inactive state. If the inactive state is a safe condition for the signal to be in for an extended period of time, this keeps the output signal from driving the outside world in a dangerous manner during the fault condition. A fault condition can also generate a controller interrupt.

Each PWM generator can also be configured to stop counting during a stall condition. The user can select for the counters to run until they reach zero then stop, or to continue counting and reloading. A stall condition does not generate a controller interrupt.

### 15.3.8 Output Control Block

With each PWM generator block producing two raw PWM signals, the output control block takes care of the final conditioning of the PWM signals before they go to the pins. Via a single register, the set of PWM signals that are actually enabled to the pins can be modified; this can be used, for example, to perform commutation of a brushless DC motor with a single register write (and without modifying the individual PWM generators, which are modified by the feedback control loop). Similarly, fault control can disable any of the PWM signals as well. A final inversion can be applied to any of the PWM signals, making them active Low instead of the default active High.

## 15.4 Initialization and Configuration

The following example shows how to initialize the PWM Generator 0 with a 25-KHz frequency, and with a 25% duty cycle on the `PWM0` pin and a 75% duty cycle on the `PWM1` pin. This example assumes the system clock is 20 MHz.

1. Enable the PWM clock by writing a value of 0x0010.0000 to the **RCGC0** register in the System Control module.
2. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register.
3. Configure the **Run-Mode Clock Configuration (RCC)** register in the System Control module to use the PWM divide (**USEPWMDIV**) and set the divider (**PWMDIV**) to divide by 2 (000).
4. Configure the PWM generator for countdown mode with immediate updates to the parameters.
  - Write the **PWM0CTL** register with a value of 0x0000.0000.
  - Write the **PWM0GENA** register with a value of 0x0000.008C.
  - Write the **PWM0GENB** register with a value of 0x0000.080C.
5. Set the period. For a 25-KHz frequency, the period = 1/25,000, or 40 microseconds. The PWM clock source is 10 MHz; the system clock divided by 2. This translates to 400 clock ticks per period. Use this value to set the **PWM0LOAD** register. In Count-Down mode, set the **Load** field in the **PWM0LOAD** register to the requested period minus one.
  - Write the **PWM0LOAD** register with a value of 0x0000.018F.
6. Set the pulse width of the **PWM0** pin for a 25% duty cycle.
  - Write the **PWM0CMPA** register with a value of 0x0000.012B.
7. Set the pulse width of the **PWM1** pin for a 75% duty cycle.
  - Write the **PWM0CMPB** register with a value of 0x0000.0063.
8. Start the timers in PWM generator 0.
  - Write the **PWM0CTL** register with a value of 0x0000.0001.
9. Enable PWM outputs.
  - Write the **PWMENABLE** register with a value of 0x0000.0003.

## 15.5 Register Map

Table 15-3 on page 507 lists the PWM registers. The offset listed is a hexadecimal increment to the register's address, relative to the PWM base address of 0x4002.8000. Note that the PWM module clock must be enabled before the registers can be programmed (see page 193). There must be a delay of 3 system clocks after the PWM module clock is enabled before any PWM module registers are accessed.

**Table 15-3. PWM Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	PWMCTL	R/W	0x0000.0000	PWM Master Control	510
0x004	PWMSYNC	R/W	0x0000.0000	PWM Time Base Sync	511

**Table 15-3. PWM Register Map (continued)**

Offset	Name	Type	Reset	Description	See page
0x008	PWMENABLE	R/W	0x0000.0000	PWM Output Enable	512
0x00C	PWMINVERT	R/W	0x0000.0000	PWM Output Inversion	513
0x010	PWMFAULT	R/W	0x0000.0000	PWM Output Fault	514
0x014	PWMINTEN	R/W	0x0000.0000	PWM Interrupt Enable	515
0x018	PWMRIS	RO	0x0000.0000	PWM Raw Interrupt Status	516
0x01C	PWMISC	R/W1C	0x0000.0000	PWM Interrupt Status and Clear	517
0x020	PWMSTATUS	RO	0x0000.0000	PWM Status	518
0x040	PWM0CTL	R/W	0x0000.0000	PWM0 Control	519
0x044	PWM0INTEN	R/W	0x0000.0000	PWM0 Interrupt and Trigger Enable	521
0x048	PWM0RIS	RO	0x0000.0000	PWM0 Raw Interrupt Status	524
0x04C	PWM0ISC	R/W1C	0x0000.0000	PWM0 Interrupt Status and Clear	525
0x050	PWM0LOAD	R/W	0x0000.0000	PWM0 Load	526
0x054	PWM0COUNT	RO	0x0000.0000	PWM0 Counter	527
0x058	PWM0CMPA	R/W	0x0000.0000	PWM0 Compare A	528
0x05C	PWM0CMPB	R/W	0x0000.0000	PWM0 Compare B	529
0x060	PWM0GENA	R/W	0x0000.0000	PWM0 Generator A Control	530
0x064	PWM0GENB	R/W	0x0000.0000	PWM0 Generator B Control	533
0x068	PWM0DBCTL	R/W	0x0000.0000	PWM0 Dead-Band Control	536
0x06C	PWM0DBRISE	R/W	0x0000.0000	PWM0 Dead-Band Rising-Edge Delay	537
0x070	PWM0DBFALL	R/W	0x0000.0000	PWM0 Dead-Band Falling-Edge-Delay	538
0x080	PWM1CTL	R/W	0x0000.0000	PWM1 Control	519
0x084	PWM1INTEN	R/W	0x0000.0000	PWM1 Interrupt and Trigger Enable	521
0x088	PWM1RIS	RO	0x0000.0000	PWM1 Raw Interrupt Status	524
0x08C	PWM1ISC	R/W1C	0x0000.0000	PWM1 Interrupt Status and Clear	525
0x090	PWM1LOAD	R/W	0x0000.0000	PWM1 Load	526
0x094	PWM1COUNT	RO	0x0000.0000	PWM1 Counter	527
0x098	PWM1CMPA	R/W	0x0000.0000	PWM1 Compare A	528
0x09C	PWM1CMPB	R/W	0x0000.0000	PWM1 Compare B	529
0x0A0	PWM1GENA	R/W	0x0000.0000	PWM1 Generator A Control	530
0x0A4	PWM1GENB	R/W	0x0000.0000	PWM1 Generator B Control	533
0x0A8	PWM1DBCTL	R/W	0x0000.0000	PWM1 Dead-Band Control	536
0x0AC	PWM1DBRISE	R/W	0x0000.0000	PWM1 Dead-Band Rising-Edge Delay	537

**Table 15-3. PWM Register Map (continued)**

Offset	Name	Type	Reset	Description	See page
0x0B0	PWM1DBFALL	R/W	0x0000.0000	PWM1 Dead-Band Falling-Edge-Delay	538
0x0C0	PWM2CTL	R/W	0x0000.0000	PWM2 Control	519
0x0C4	PWM2INTEN	R/W	0x0000.0000	PWM2 Interrupt and Trigger Enable	521
0x0C8	PWM2RIS	RO	0x0000.0000	PWM2 Raw Interrupt Status	524
0x0CC	PWM2ISC	R/W1C	0x0000.0000	PWM2 Interrupt Status and Clear	525
0x0D0	PWM2LOAD	R/W	0x0000.0000	PWM2 Load	526
0x0D4	PWM2COUNT	RO	0x0000.0000	PWM2 Counter	527
0x0D8	PWM2CMPA	R/W	0x0000.0000	PWM2 Compare A	528
0x0DC	PWM2CMPB	R/W	0x0000.0000	PWM2 Compare B	529
0x0E0	PWM2GENA	R/W	0x0000.0000	PWM2 Generator A Control	530
0x0E4	PWM2GENB	R/W	0x0000.0000	PWM2 Generator B Control	533
0x0E8	PWM2DBCTL	R/W	0x0000.0000	PWM2 Dead-Band Control	536
0x0EC	PWM2DBRISE	R/W	0x0000.0000	PWM2 Dead-Band Rising-Edge Delay	537
0x0F0	PWM2DBFALL	R/W	0x0000.0000	PWM2 Dead-Band Falling-Edge-Delay	538

## 15.6 Register Descriptions

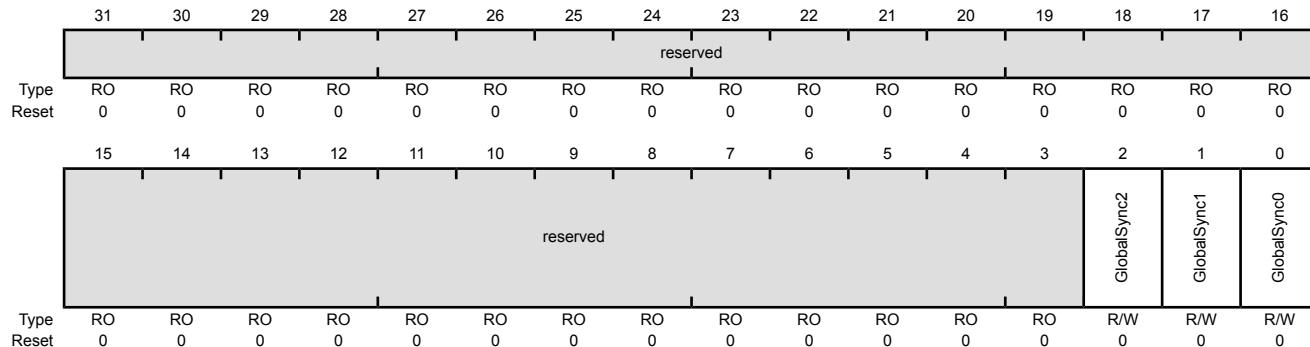
The remainder of this section lists and describes the PWM registers, in numerical order by address offset.

## Register 1: PWM Master Control (PWMCTL), offset 0x000

This register provides master control over the PWM generation blocks.

### PWM Master Control (PWMCTL)

Base 0x4002.8000  
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	GlobalSync2	R/W	0	Update PWM Generator 2 Same as GlobalSync0 but for PWM generator 2.
1	GlobalSync1	R/W	0	Update PWM Generator 1 Same as GlobalSync0 but for PWM generator 1.
0	GlobalSync0	R/W	0	Update PWM Generator 0 Setting this bit causes any queued update to a load or comparator register in PWM generator 0 to be applied the next time the corresponding counter becomes zero. This bit automatically clears when the updates have completed; it cannot be cleared by software.

## Register 2: PWM Time Base Sync (PWMSYNC), offset 0x004

This register provides a method to perform synchronization of the counters in the PWM generation blocks. Writing a bit in this register to 1 causes the specified counter to reset back to 0; writing multiple bits resets multiple counters simultaneously. The bits auto-clear after the reset has occurred; reading them back as zero indicates that the synchronization has completed.

### PWM Time Base Sync (PWMSYNC)

Base 0x4002.8000  
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															
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	Sync2	R/W	0	Reset Generator 2 Counter Performs a reset of the PWM generator 2 counter.
1	Sync1	R/W	0	Reset Generator 1 Counter Performs a reset of the PWM generator 1 counter.
0	Sync0	R/W	0	Reset Generator 0 Counter Performs a reset of the PWM generator 0 counter.

## Register 3: PWM Output Enable (PWMDENABLE), offset 0x008

This register provides a master control of which generated PWM signals are output to device pins. By disabling a PWM output, the generation process can continue (for example, when the time bases are synchronized) without driving PWM signals to the pins. When bits in this register are set, the corresponding PWM signal is passed through to the output stage, which is controlled by the **PWMINVERT** register. When bits are not set, the PWM signal is replaced by a zero value which is also passed to the output stage.

### PWM Output Enable (PWMDENABLE)

Base 0x4002.8000  
Offset 0x008  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	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
Type	RO	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

reserved

reserved

| PWM5En | PWM4En | PWM3En | PWM2En | PWM1En | PWM0En |

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	PWM5En	R/W	0	PWM5 Output Enable When set, allows the generated PWM5 signal to be passed to the device pin.
4	PWM4En	R/W	0	PWM4 Output Enable When set, allows the generated PWM4 signal to be passed to the device pin.
3	PWM3En	R/W	0	PWM3 Output Enable When set, allows the generated PWM3 signal to be passed to the device pin.
2	PWM2En	R/W	0	PWM2 Output Enable When set, allows the generated PWM2 signal to be passed to the device pin.
1	PWM1En	R/W	0	PWM1 Output Enable When set, allows the generated PWM1 signal to be passed to the device pin.
0	PWM0En	R/W	0	PWM0 Output Enable When set, allows the generated PWM0 signal to be passed to the device pin.

## Register 4: PWM Output Inversion (P威MINVERT), offset 0x00C

This register provides a master control of the polarity of the PWM signals on the device pins. The PWM signals generated by the PWM generator are active High; they can optionally be made active Low via this register. Disabled PWM channels are also passed through the output inverter (if so configured) so that inactive channels maintain the correct polarity.

### PWM Output Inversion (P威MINVERT)

Base 0x4002.8000  
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															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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
31:6	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	PWM5Inv	R/W	0	Invert PWM5 Signal When set, the generated PWM5 signal is inverted.
4	PWM4Inv	R/W	0	Invert PWM4 Signal When set, the generated PWM4 signal is inverted.
3	PWM3Inv	R/W	0	Invert PWM3 Signal When set, the generated PWM3 signal is inverted.
2	PWM2Inv	R/W	0	Invert PWM2 Signal When set, the generated PWM2 signal is inverted.
1	PWM1Inv	R/W	0	Invert PWM1 Signal When set, the generated PWM1 signal is inverted.
0	PWM0Inv	R/W	0	Invert PWM0 Signal When set, the generated PWM0 signal is inverted.

## Register 5: PWM Output Fault (PWMFAULT), offset 0x010

This register controls the behavior of the PWM outputs in the presence of fault conditions. Both the fault inputs and debug events are considered fault conditions. On a fault condition, each PWM signal can be passed through unmodified or driven Low. For outputs that are configured for pass-through, the debug event handling on the corresponding PWM generator also determines if the PWM signal continues to be generated.

Fault condition control occurs before the output inverter, so PWM signals driven Low on fault are inverted if the channel is configured for inversion (therefore, the pin is driven High on a fault condition).

### PWM Output Fault (PWMFAULT)

Base 0x4002.8000  
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															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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
31:6	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	Fault5	R/W	0	PWM5 Fault When set, the PWM5 output signal is driven Low on a fault condition.
4	Fault4	R/W	0	PWM4 Fault When set, the PWM4 output signal is driven Low on a fault condition.
3	Fault3	R/W	0	PWM3 Fault When set, the PWM3 output signal is driven Low on a fault condition.
2	Fault2	R/W	0	PWM2 Fault When set, the PWM2 output signal is driven Low on a fault condition.
1	Fault1	R/W	0	PWM1 Fault When set, the PWM1 output signal is driven Low on a fault condition.
0	Fault0	R/W	0	PWM0 Fault When set, the PWM0 output signal is driven Low on a fault condition.

## Register 6: PWM Interrupt Enable (PWMINTEN), offset 0x014

This register controls the global interrupt generation capabilities of the PWM module. The events that can cause an interrupt are the fault input and the individual interrupts from the PWM generators.

### PWM Interrupt Enable (PWMINTEN)

Base 0x4002.8000  
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															IntFault
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															IntPWM2 IntPWM1 IntPWM0
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
31:17	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.
16	IntFault	R/W	0	Fault Interrupt Enable When set, an interrupt occurs when the fault input is asserted.
15:3	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	IntPWM2	R/W	0	PWM2 Interrupt Enable When set, an interrupt occurs when the PWM generator 2 block asserts an interrupt.
1	IntPWM1	R/W	0	PWM1 Interrupt Enable When set, an interrupt occurs when the PWM generator 1 block asserts an interrupt.
0	IntPWM0	R/W	0	PWM0 Interrupt Enable When set, an interrupt occurs when the PWM generator 0 block asserts an interrupt.

## Register 7: PWM Raw Interrupt Status (PWMRIS), offset 0x018

This register provides the current set of interrupt sources that are asserted, regardless of whether they cause an interrupt to be asserted to the controller. The fault interrupt is latched on detection; it must be cleared through the **PWM Interrupt Status and Clear (PWMISC)** register (see page 517). The PWM generator interrupts simply reflect the status of the PWM generators; they are cleared via the interrupt status register in the PWM generator blocks. Bits set to 1 indicate the events that are active; zero bits indicate that the event in question is not active.

### PWM Raw Interrupt Status (PWMRIS)

Base 0x4002.8000

Offset 0x018

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															IntFault
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															IntPWM2
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
31:17	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.
16	IntFault	RO	0	Fault Interrupt Asserted Indicates that the fault input is asserting.
15:3	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	IntPWM2	RO	0	PWM2 Interrupt Asserted Indicates that the PWM generator 2 block is asserting its interrupt.
1	IntPWM1	RO	0	PWM1 Interrupt Asserted Indicates that the PWM generator 1 block is asserting its interrupt.
0	IntPWM0	RO	0	PWM0 Interrupt Asserted Indicates that the PWM generator 0 block is asserting its interrupt.

## Register 8: PWM Interrupt Status and Clear (PWMISC), offset 0x01C

This register provides a summary of the interrupt status of the individual PWM generator blocks. A bit set to 1 indicates that the corresponding generator block is asserting an interrupt. The individual interrupt status registers in each block must be consulted to determine the reason for the interrupt, and used to clear the interrupt. For the fault interrupt, a write of 1 to that bit position clears the latched interrupt status.

### PWM Interrupt Status and Clear (PWMISC)

Base 0x4002.8000  
Offset 0x01C  
Type R/W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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
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
	IntFault	IntPWM2	IntPWM1	IntPWM0												

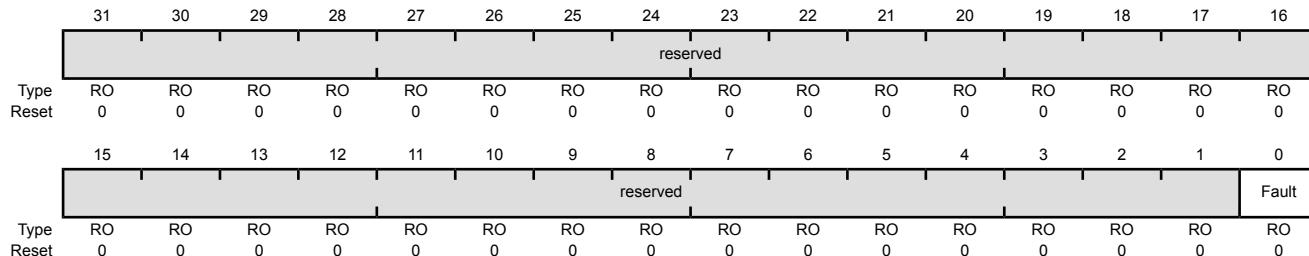
Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	IntFault	R/W1C	0	Fault Interrupt Asserted Indicates that the fault input is asserting an interrupt.
15:3	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	IntPWM2	RO	0	PWM2 Interrupt Status Indicates if the PWM generator 2 block is asserting an interrupt.
1	IntPWM1	RO	0	PWM1 Interrupt Status Indicates if the PWM generator 1 block is asserting an interrupt.
0	IntPWM0	RO	0	PWM0 Interrupt Status Indicates if the PWM generator 0 block is asserting an interrupt.

**Register 9: PWM Status (PWMSTATUS), offset 0x020**

This register provides the status of the FAULT input signal.

**PWM Status (PWMSTATUS)**

Base 0x4002.8000  
Offset 0x020  
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	Fault	RO	0	Fault Interrupt Status When set, indicates the fault input is asserted.

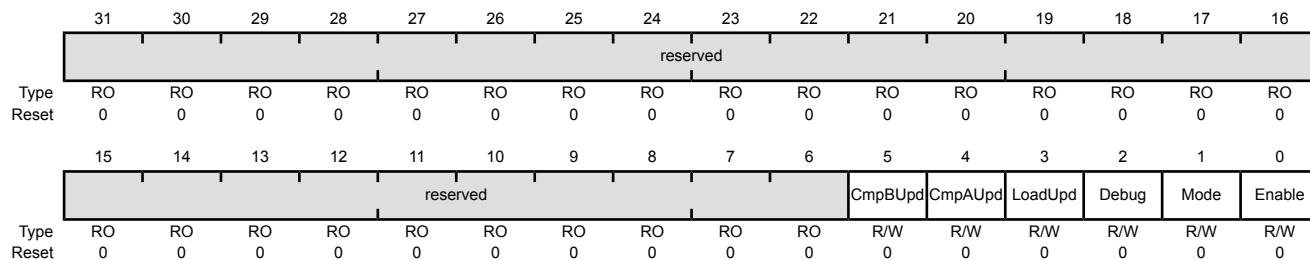
**Register 10: PWM0 Control (PWM0CTL), offset 0x040****Register 11: PWM1 Control (PWM1CTL), offset 0x080****Register 12: PWM2 Control (PWM2CTL), offset 0x0C0**

These registers configure the PWM signal generation blocks (PWM0CTL controls the PWM generator 0 block, and so on). The Register Update mode, Debug mode, Counting mode, and Block Enable mode are all controlled via these registers. The blocks produce the PWM signals, which can be either two independent PWM signals (from the same counter), or a paired set of PWM signals with dead-band delays added.

The PWM0 block produces the PWM0 and PWM1 outputs, the PWM1 block produces the PWM2 and PWM3 outputs, and the PWM2 block produces the PWM4 and PWM5 outputs.

**PWM0 Control (PWM0CTL)**

Base 0x4002.8000  
Offset 0x040  
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	CmpBUpd	R/W	0	Comparator B Update Mode Same as CmpAUpd but for the comparator B register.
4	CmpAUpd	R/W	0	Comparator A Update Mode The Update mode for the comparator A register. When not set, updates to the register are reflected to the comparator the next time the counter is 0. When set, updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the <b>PWM Master Control (PWMCTL)</b> register (see page 510).
3	LoadUpd	R/W	0	Load Register Update Mode The Update mode for the load register. When not set, updates to the register are reflected to the counter the next time the counter is 0. When set, updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the <b>PWM Master Control (PWMCTL)</b> register.
2	Debug	R/W	0	Debug Mode The behavior of the counter in Debug mode. When not set, the counter stops running when it next reaches 0, and continues running again when no longer in Debug mode. When set, the counter always runs.

Bit/Field	Name	Type	Reset	Description
1	Mode	R/W	0	<b>Counter Mode</b> The mode for the counter. When not set, the counter counts down from the load value to 0 and then wraps back to the load value (Count-Down mode). When set, the counter counts up from 0 to the load value, back down to 0, and then repeats (Count-Up/Down mode).
0	Enable	R/W	0	<b>PWM Block Enable</b> Master enable for the PWM generation block. When not set, the entire block is disabled and not clocked. When set, the block is enabled and produces PWM signals.

**Register 13: PWM0 Interrupt and Trigger Enable (PWM0INTEN), offset 0x044****Register 14: PWM1 Interrupt and Trigger Enable (PWM1INTEN), offset 0x084****Register 15: PWM2 Interrupt and Trigger Enable (PWM2INTEN), offset 0x0C4**

These registers control the interrupt and ADC trigger generation capabilities of the PWM generators (**PWM0INTEN** controls the PWM generator 0 block, and so on). The events that can cause an interrupt or an ADC trigger are:

- The counter being equal to the load register
- The counter being equal to zero
- The counter being equal to the comparator A register while counting up
- The counter being equal to the comparator A register while counting down
- The counter being equal to the comparator B register while counting up
- The counter being equal to the comparator B register while counting down

Any combination of these events can generate either an interrupt, or an ADC trigger; though no determination can be made as to the actual event that caused an ADC trigger if more than one is specified.

**PWM0 Interrupt and Trigger Enable (PWM0INTEN)**

Base 0x4002.8000

Offset 0x044

Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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
Type	reserved		TrCmpBD	TrCmpBU	TrCmpAD	TrCmpAU	TrCnLoad	TrCnZero	reserved	IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCnLoad	IntCnZero	
Reset	0	0	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W

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	TrCmpBD	R/W	0	Trigger for Counter=Comparator B Down

Value	Description
1	An ADC trigger pulse is output when the counter matches the value in the <b>PWMnCMPB</b> register value while counting down.
0	No ADC trigger is output.

Bit/Field	Name	Type	Reset	Description
12	TrCmpBU	R/W	0	Trigger for Counter=Comparator B Up  Value Description 1 An ADC trigger pulse is output when the counter matches the value in the <b>PWMnCMPB</b> register value while counting up. 0 No ADC trigger is output.
11	TrCmpAD	R/W	0	Trigger for Counter=Comparator A Down  Value Description 1 An ADC trigger pulse is output when the counter matches the value in the <b>PWMnCMPA</b> register value while counting down. 0 No ADC trigger is output.
10	TrCmpAU	R/W	0	Trigger for Counter=Comparator A Up  Value Description 1 An ADC trigger pulse is output when the counter matches the value in the <b>PWMnCMPA</b> register value while counting up. 0 No ADC trigger is output.
9	TrCntLoad	R/W	0	Trigger for Counter=Load  Value Description 1 An ADC trigger pulse is output when the counter matches the <b>PWMnLOAD</b> register. 0 No ADC trigger is output.
8	TrCntZero	R/W	0	Trigger for Counter=0  Value Description 1 An ADC trigger pulse is output when the counter is 0. 0 No ADC trigger is output.
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	IntCmpBD	R/W	0	Interrupt for Counter=Comparator B Down  Value Description 1 A raw interrupt occurs when the counter matches the value in the <b>PWMnCMPB</b> register value while counting down. 0 No interrupt.

Bit/Field	Name	Type	Reset	Description
4	IntCmpBU	R/W	0	Interrupt for Counter=Comparator B Up  Value Description 1 A raw interrupt occurs when the counter matches the value in the <b>PWMnCMPB</b> register value while counting up. 0 No interrupt.
3	IntCmpAD	R/W	0	Interrupt for Counter=Comparator A Down  Value Description 1 A raw interrupt occurs when the counter matches the value in the <b>PWMnCMPA</b> register value while counting down. 0 No interrupt.
2	IntCmpAU	R/W	0	Interrupt for Counter=Comparator A Up  Value Description 1 A raw interrupt occurs when the counter matches the value in the <b>PWMnCMPA</b> register value while counting up. 0 No interrupt.
1	IntCntLoad	R/W	0	Interrupt for Counter=Load  Value Description 1 A raw interrupt occurs when the counter matches the value in the <b>PWMnLOAD</b> register value. 0 No interrupt.
0	IntCntZero	R/W	0	Interrupt for Counter=0  Value Description 1 A raw interrupt occurs when the counter is zero. 0 No interrupt.

**Register 16: PWM0 Raw Interrupt Status (PWM0RIS), offset 0x048****Register 17: PWM1 Raw Interrupt Status (PWM1RIS), offset 0x088****Register 18: PWM2 Raw Interrupt Status (PWM2RIS), offset 0x0C8**

These registers provide the current set of interrupt sources that are asserted, regardless of whether they cause an interrupt to be asserted to the controller (**PWM0RIS** controls the PWM generator 0 block, and so on). Bits set to 1 indicate the latched events that have occurred; bits set to 0 indicate that the event in question has not occurred.

## PWM0 Raw Interrupt Status (PWM0RIS)

Base 0x4002.8000  
Offset 0x048  
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

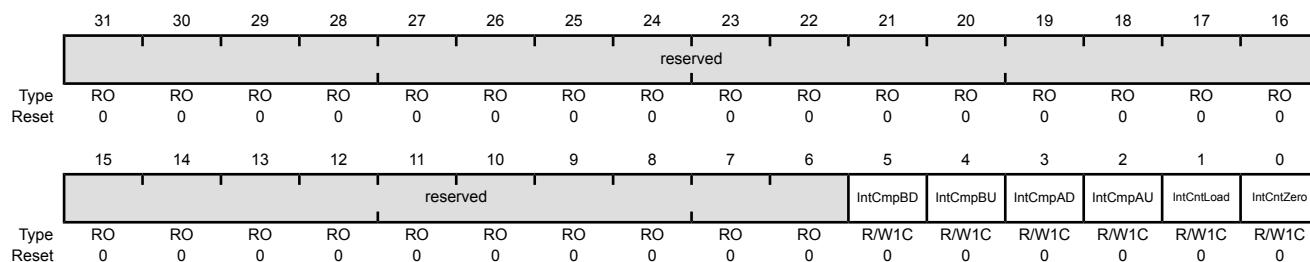
Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	IntCmpBD	RO	0	Comparator B Down Interrupt Status Indicates that the counter has matched the comparator B value while counting down.
4	IntCmpBU	RO	0	Comparator B Up Interrupt Status Indicates that the counter has matched the comparator B value while counting up.
3	IntCmpAD	RO	0	Comparator A Down Interrupt Status Indicates that the counter has matched the comparator A value while counting down.
2	IntCmpAU	RO	0	Comparator A Up Interrupt Status Indicates that the counter has matched the comparator A value while counting up.
1	IntCntLoad	RO	0	Counter=Load Interrupt Status Indicates that the counter has matched the <b>PWMnLOAD</b> register.
0	IntCntZero	RO	0	Counter=0 Interrupt Status Indicates that the counter has matched 0.

**Register 19: PWM0 Interrupt Status and Clear (PWM0ISC), offset 0x04C****Register 20: PWM1 Interrupt Status and Clear (PWM1ISC), offset 0x08C****Register 21: PWM2 Interrupt Status and Clear (PWM2ISC), offset 0x0CC**

These registers provide the current set of interrupt sources that are asserted to the controller (**PWM0ISC** controls the PWM generator 0 block, and so on). Bits set to 1 indicate the latched events that have occurred; bits set to 0 indicate that the event in question has not occurred. These are R/W1C registers; writing a 1 to a bit position clears the corresponding interrupt reason.

**PWM0 Interrupt Status and Clear (PWM0ISC)**

Base 0x4002.8000  
Offset 0x04C  
Type R/W1C, 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	IntCmpBD	R/W1C	0	Comparator B Down Interrupt Indicates that the counter has matched the comparator B value while counting down.
4	IntCmpBU	R/W1C	0	Comparator B Up Interrupt Indicates that the counter has matched the comparator B value while counting up.
3	IntCmpAD	R/W1C	0	Comparator A Down Interrupt Indicates that the counter has matched the comparator A value while counting down.
2	IntCmpAU	R/W1C	0	Comparator A Up Interrupt Indicates that the counter has matched the comparator A value while counting up.
1	IntCntLoad	R/W1C	0	Counter=Load Interrupt Indicates that the counter has matched the <b>PWMnLOAD</b> register.
0	IntCntZero	R/W1C	0	Counter=0 Interrupt Indicates that the counter has matched 0.

**Register 22: PWM0 Load (PWM0LOAD), offset 0x050****Register 23: PWM1 Load (PWM1LOAD), offset 0x090****Register 24: PWM2 Load (PWM2LOAD), offset 0x0D0**

These registers contain the load value for the PWM counter (**PWM0LOAD** controls the PWM generator 0 block, and so on). Based on the counter mode, either this value is loaded into the counter after it reaches zero, or it is the limit of up-counting after which the counter decrements back to zero.

If the Load Value Update mode is immediate, this value is used the next time the counter reaches zero; if the mode is synchronous, it is used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 510). If this register is re-written before the actual update occurs, the previous value is never used and is lost.

## PWM0 Load (PWM0LOAD)

Base 0x4002.8000  
Offset 0x050  
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
	Load															
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

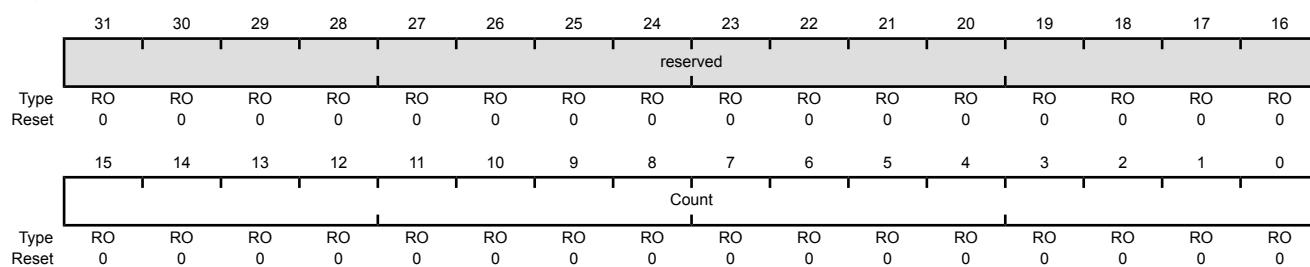
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	Load	R/W	0	Counter Load Value The counter load value.

**Register 25: PWM0 Counter (PWM0COUNT), offset 0x054****Register 26: PWM1 Counter (PWM1COUNT), offset 0x094****Register 27: PWM2 Counter (PWM2COUNT), offset 0x0D4**

These registers contain the current value of the PWM counter (**PWM0COUNT** is the value of the PWM generator 0 block, and so on). When this value matches the load register, a pulse is output; this can drive the generation of a PWM signal (via the **PWMnGENA/PWMnGENB** registers, see page 530 and page 533) or drive an interrupt or ADC trigger (via the **PWMnINTEN** register, see page 521). A pulse with the same capabilities is generated when this value is zero.

**PWM0 Counter (PWM0COUNT)**

Base 0x4002.8000  
Offset 0x054  
Type RO, 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.
15:0	Count	RO	0x00	Counter Value The current value of the counter.

**Register 28: PWM0 Compare A (PWM0CMPA), offset 0x058****Register 29: PWM1 Compare A (PWM1CMPA), offset 0x098****Register 30: PWM2 Compare A (PWM2CMPA), offset 0x0D8**

These registers contain a value to be compared against the counter (PWM0CMPA controls the PWM generator 0 block, and so on). When this value matches the counter, a pulse is output; this can drive the generation of a PWM signal (via the **PWMnGENA/PWMnGENB** registers) or drive an interrupt or ADC trigger (via the **PWMnINTEN** register). If the value of this register is greater than the **PWMnLOAD** register (see page 526), then no pulse is ever output.

If the comparator A update mode is immediate (based on the CmpAUpd bit in the **PWMnCTL** register), this 16-bit CompA value is used the next time the counter reaches zero. If the update mode is synchronous, it is used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 510). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

## PWM0 Compare A (PWM0CMPA)

Base 0x4002.8000

Offset 0x058

Type R/W, reset 0x0000.0000

reserved															
Type	RO														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CompA															
Type	R/W														
Reset	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.
15:0	CompA	R/W	0x00	Comparator A Value The value to be compared against the counter.

**Register 31: PWM0 Compare B (PWM0CMPB), offset 0x05C****Register 32: PWM1 Compare B (PWM1CMPB), offset 0x09C****Register 33: PWM2 Compare B (PWM2CMPB), offset 0x0DC**

These registers contain a value to be compared against the counter (PWM0CMPB controls the PWM generator 0 block, and so on). When this value matches the counter, a pulse is output; this can drive the generation of a PWM signal (via the **PWMnGENA/PWMnGENB** registers) or drive an interrupt or ADC trigger (via the **PWMnINTEN** register). If the value of this register is greater than the **PWMnLOAD** register, no pulse is ever output.

If the comparator B update mode is immediate (based on the CmpBUpd bit in the **PWMnCTL** register), this 16-bit CompB value is used the next time the counter reaches zero. If the update mode is synchronous, it is used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 510). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

## PWM0 Compare B (PWM0CMPB)

Base 0x4002.8000  
Offset 0x05C  
Type R/W, reset 0x0000.0000

reserved															
Type	RO														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CompB															
Type	R/W														
Reset	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.
15:0	CompB	R/W	0x00	Comparator B Value The value to be compared against the counter.

**Register 34: PWM0 Generator A Control (PWM0GENA), offset 0x060****Register 35: PWM1 Generator A Control (PWM1GENA), offset 0x0A0****Register 36: PWM2 Generator A Control (PWM2GENA), offset 0x0E0**

These registers control the generation of the `PWMnA` signal based on the load and zero output pulses from the counter, as well as the compare A and compare B pulses from the comparators (**PWM0GENA** controls the PWM generator 0 block, and so on). When the counter is running in Count-Down mode, only four of these events occur; when running in Count-Up/Down mode, all six occur. These events provide great flexibility in the positioning and duty cycle of the PWM signal that is produced.

The **PWM0GENA** register controls generation of the `PWM0A` signal; **PWM1GENA**, the `PWM1A` signal; and **PWM2GENA**, the `PWM2A` signal.

If a zero or load event coincides with a compare A or compare B event, the zero or load action is taken and the compare A or compare B action is ignored. If a compare A event coincides with a compare B event, the compare A action is taken and the compare B action is ignored.

**PWM0 Generator A Control (PWM0GENA)**

Base 0x4002.8000  
Offset 0x060  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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
Type	RO	RO	RO	RO	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
31:12	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.
11:10	ActCmpBD	R/W	0x0	Action for Comparator B Down  The action to be taken when the counter matches comparator B while counting down.  The table below defines the effect of the event on the output signal.

Value	Description
0x0	Do nothing.
0x1	Invert the output signal.
0x2	Set the output signal to 0.
0x3	Set the output signal to 1.

Bit/Field	Name	Type	Reset	Description										
9:8	ActCmpBU	R/W	0x0	<p>Action for Comparator B Up</p> <p>The action to be taken when the counter matches comparator B while counting up. Occurs only when the Mode bit in the <b>PWMnCTL</b> register (see page 519) is set to 1.</p> <p>The table below defines the effect of the event on the output signal.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Do nothing.</td></tr> <tr> <td>0x1</td><td>Invert the output signal.</td></tr> <tr> <td>0x2</td><td>Set the output signal to 0.</td></tr> <tr> <td>0x3</td><td>Set the output signal to 1.</td></tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert the output signal.	0x2	Set the output signal to 0.	0x3	Set the output signal to 1.
Value	Description													
0x0	Do nothing.													
0x1	Invert the output signal.													
0x2	Set the output signal to 0.													
0x3	Set the output signal to 1.													
7:6	ActCmpAD	R/W	0x0	<p>Action for Comparator A Down</p> <p>The action to be taken when the counter matches comparator A while counting down.</p> <p>The table below defines the effect of the event on the output signal.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Do nothing.</td></tr> <tr> <td>0x1</td><td>Invert the output signal.</td></tr> <tr> <td>0x2</td><td>Set the output signal to 0.</td></tr> <tr> <td>0x3</td><td>Set the output signal to 1.</td></tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert the output signal.	0x2	Set the output signal to 0.	0x3	Set the output signal to 1.
Value	Description													
0x0	Do nothing.													
0x1	Invert the output signal.													
0x2	Set the output signal to 0.													
0x3	Set the output signal to 1.													
5:4	ActCmpAU	R/W	0x0	<p>Action for Comparator A Up</p> <p>The action to be taken when the counter matches comparator A while counting up. Occurs only when the Mode bit in the <b>PWMnCTL</b> register is set to 1.</p> <p>The table below defines the effect of the event on the output signal.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Do nothing.</td></tr> <tr> <td>0x1</td><td>Invert the output signal.</td></tr> <tr> <td>0x2</td><td>Set the output signal to 0.</td></tr> <tr> <td>0x3</td><td>Set the output signal to 1.</td></tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert the output signal.	0x2	Set the output signal to 0.	0x3	Set the output signal to 1.
Value	Description													
0x0	Do nothing.													
0x1	Invert the output signal.													
0x2	Set the output signal to 0.													
0x3	Set the output signal to 1.													
3:2	ActLoad	R/W	0x0	<p>Action for Counter=Load</p> <p>The action to be taken when the counter matches the load value.</p> <p>The table below defines the effect of the event on the output signal.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Do nothing.</td></tr> <tr> <td>0x1</td><td>Invert the output signal.</td></tr> <tr> <td>0x2</td><td>Set the output signal to 0.</td></tr> <tr> <td>0x3</td><td>Set the output signal to 1.</td></tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert the output signal.	0x2	Set the output signal to 0.	0x3	Set the output signal to 1.
Value	Description													
0x0	Do nothing.													
0x1	Invert the output signal.													
0x2	Set the output signal to 0.													
0x3	Set the output signal to 1.													

Bit/Field	Name	Type	Reset	Description
1:0	ActZero	R/W	0x0	Action for Counter=0 The action to be taken when the counter is zero. The table below defines the effect of the event on the output signal.
Value Description				
0x0 Do nothing.				
0x1 Invert the output signal.				
0x2 Set the output signal to 0.				
0x3 Set the output signal to 1.				

**Register 37: PWM0 Generator B Control (PWM0GENB), offset 0x064****Register 38: PWM1 Generator B Control (PWM1GENB), offset 0x0A4****Register 39: PWM2 Generator B Control (PWM2GENB), offset 0x0E4**

These registers control the generation of the `PWMnB` signal based on the load and zero output pulses from the counter, as well as the compare A and compare B pulses from the comparators (**PWM0GENB** controls the PWM generator 0 block, and so on). When the counter is running in Down mode, only four of these events occur; when running in Up/Down mode, all six occur. These events provide great flexibility in the positioning and duty cycle of the PWM signal that is produced.

The **PWM0GENB** register controls generation of the `PWM0B` signal; **PWM1GENB**, the `PWM1B` signal; and **PWM2GENB**, the `PWM2B` signal.

If a zero or load event coincides with a compare A or compare B event, the zero or load action is taken and the compare A or compare B action is ignored. If a compare A event coincides with a compare B event, the compare B action is taken and the compare A action is ignored.

## PWM0 Generator B Control (PWM0GENB)

Base 0x4002.8000  
Offset 0x064  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
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
Type	reserved	ActCmpBD	ActCmpBU	ActCmpAD	ActCmpAU	ActLoad	ActZero									
Reset	0	0	0	0	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit/Field	Name	Type	Reset	Description
31:12	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.
11:10	ActCmpBD	R/W	0x0	Action for Comparator B Down  The action to be taken when the counter matches comparator B while counting down.  The table below defines the effect of the event on the output signal.

Value	Description
0x0	Do nothing.
0x1	Invert the output signal.
0x2	Set the output signal to 0.
0x3	Set the output signal to 1.

Bit/Field	Name	Type	Reset	Description										
9:8	ActCmpBU	R/W	0x0	<p>Action for Comparator B Up</p> <p>The action to be taken when the counter matches comparator B while counting up. Occurs only when the Mode bit in the <b>PWMnCTL</b> register is set to 1.</p> <p>The table below defines the effect of the event on the output signal.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Do nothing.</td></tr> <tr> <td>0x1</td><td>Invert the output signal.</td></tr> <tr> <td>0x2</td><td>Set the output signal to 0.</td></tr> <tr> <td>0x3</td><td>Set the output signal to 1.</td></tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert the output signal.	0x2	Set the output signal to 0.	0x3	Set the output signal to 1.
Value	Description													
0x0	Do nothing.													
0x1	Invert the output signal.													
0x2	Set the output signal to 0.													
0x3	Set the output signal to 1.													
7:6	ActCmpAD	R/W	0x0	<p>Action for Comparator A Down</p> <p>The action to be taken when the counter matches comparator A while counting down.</p> <p>The table below defines the effect of the event on the output signal.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Do nothing.</td></tr> <tr> <td>0x1</td><td>Invert the output signal.</td></tr> <tr> <td>0x2</td><td>Set the output signal to 0.</td></tr> <tr> <td>0x3</td><td>Set the output signal to 1.</td></tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert the output signal.	0x2	Set the output signal to 0.	0x3	Set the output signal to 1.
Value	Description													
0x0	Do nothing.													
0x1	Invert the output signal.													
0x2	Set the output signal to 0.													
0x3	Set the output signal to 1.													
5:4	ActCmpAU	R/W	0x0	<p>Action for Comparator A Up</p> <p>The action to be taken when the counter matches comparator A while counting up. Occurs only when the Mode bit in the <b>PWMnCTL</b> register is set to 1.</p> <p>The table below defines the effect of the event on the output signal.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Do nothing.</td></tr> <tr> <td>0x1</td><td>Invert the output signal.</td></tr> <tr> <td>0x2</td><td>Set the output signal to 0.</td></tr> <tr> <td>0x3</td><td>Set the output signal to 1.</td></tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert the output signal.	0x2	Set the output signal to 0.	0x3	Set the output signal to 1.
Value	Description													
0x0	Do nothing.													
0x1	Invert the output signal.													
0x2	Set the output signal to 0.													
0x3	Set the output signal to 1.													
3:2	ActLoad	R/W	0x0	<p>Action for Counter=Load</p> <p>The action to be taken when the counter matches the load value.</p> <p>The table below defines the effect of the event on the output signal.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Do nothing.</td></tr> <tr> <td>0x1</td><td>Invert the output signal.</td></tr> <tr> <td>0x2</td><td>Set the output signal to 0.</td></tr> <tr> <td>0x3</td><td>Set the output signal to 1.</td></tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert the output signal.	0x2	Set the output signal to 0.	0x3	Set the output signal to 1.
Value	Description													
0x0	Do nothing.													
0x1	Invert the output signal.													
0x2	Set the output signal to 0.													
0x3	Set the output signal to 1.													

Bit/Field	Name	Type	Reset	Description
1:0	ActZero	R/W	0x0	Action for Counter=0 The action to be taken when the counter is 0. The table below defines the effect of the event on the output signal.
Value Description				
0x0 Do nothing.				
0x1 Invert the output signal.				
0x2 Set the output signal to 0.				
0x3 Set the output signal to 1.				

**Register 40: PWM0 Dead-Band Control (PWM0DBCTL), offset 0x068****Register 41: PWM1 Dead-Band Control (PWM1DBCTL), offset 0x0A8****Register 42: PWM2 Dead-Band Control (PWM2DBCTL), offset 0x0E8**

The **PWM0DBCTL** register controls the dead-band generator, which produces the **PWM0** and **PWM1** signals based on the **PWM0A** and **PWM0B** signals. When disabled, the **PWM0A** signal passes through to the **PWM0** signal and the **PWM0B** signal passes through to the **PWM1** signal. When enabled and inverting the resulting waveform, the **PWM0B** signal is ignored; the **PWM0** signal is generated by delaying the rising edge(s) of the **PWM0A** signal by the value in the **PWM0DBRISE** register (see page 537), and the **PWM1** signal is generated by delaying the falling edge(s) of the **PWM0A** signal by the value in the **PWM0DBFALL** register (see page 538). In a similar manner, **PWM2** and **PWM3** are produced from the **PWM1A** and **PWM1B** signals, and **PWM4** and **PWM5** are produced from the **PWM2A** and **PWM2B** signals.

## PWM0 Dead-Band Control (PWM0DBCTL)

Base 0x4002.8000  
Offset 0x068  
Type R/W, reset 0x0000.0000

Type	RO														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type	RO	R/W													
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	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	Enable	R/W	0	Dead-Band Generator Enable When set, the dead-band generator inserts dead bands into the output signals; when clear, it simply passes the PWM signals through.

### Register 43: PWM0 Dead-Band Rising-Edge Delay (PWM0DBRISE), offset 0x06C

### Register 44: PWM1 Dead-Band Rising-Edge Delay (PWM1DBRISE), offset 0x0AC

### Register 45: PWM2 Dead-Band Rising-Edge Delay (PWM2DBRISE), offset 0x0EC

The **PWM0DBRISE** register contains the number of clock ticks to delay the rising edge of the **PWM0A** signal when generating the **PWM0** signal. If the dead-band generator is disabled through the **PWMnDBCTL** register, the **PWM0DBRISE** register is ignored. If the value of this register is larger than the width of a High pulse on the input PWM signal, the rising-edge delay consumes the entire High time of the signal, resulting in no High time on the output. Care must be taken to ensure that the input High time always exceeds the rising-edge delay. In a similar manner, PWM2 is generated from **PWM1A** with its rising edge delayed and **PWM4** is produced from **PWM2A** with its rising edge delayed.

#### PWM0 Dead-Band Rising-Edge Delay (PWM0DBRISE)

Base 0x4002.8000  
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															RiseDelay
Type	RO	RO	RO	RO	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
31:12	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.
11:0	RiseDelay	R/W	0	Dead-Band Rise Delay The number of clock ticks to delay the rising edge.

**Register 46: PWM0 Dead-Band Falling-Edge-Delay (PWM0DBFALL), offset 0x070****Register 47: PWM1 Dead-Band Falling-Edge-Delay (PWM1DBFALL), offset 0x0B0****Register 48: PWM2 Dead-Band Falling-Edge-Delay (PWM2DBFALL), offset 0x0F0**

The **PWM0DBFALL** register contains the number of clock ticks to delay the falling edge of the PWM0A signal when generating the PWM1 signal. If the dead-band generator is disabled, this register is ignored. If the value of this register is larger than the width of a Low pulse on the input PWM signal, the falling-edge delay consumes the entire Low time of the signal, resulting in no Low time on the output. Care must be taken to ensure that the input Low time always exceeds the falling-edge delay. In a similar manner, PWM3 is generated from PWM1A with its falling edge delayed and PWM5 is produced from PWM2A with its falling edge delayed.

## PWM0 Dead-Band Falling-Edge-Delay (PWM0DBFALL)

Base 0x4002.8000

Offset 0x070

Type R/W, reset 0x0000.0000

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								FallDelay								
Type	RO	RO	RO	RO	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	

Bit/Field	Name	Type	Reset	Description
31:12	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.
11:0	FallDelay	R/W	0x00	Dead-Band Fall Delay The number of clock ticks to delay the falling edge.

# 16 Pin Diagram

The LM3S811 microcontroller pin diagrams are shown below.

**Note:** The 48-pin QFN package is OBSOLETE. TI has discontinued production of this device.

Figure 16-1. 48-Pin QFP Package Pin Diagram

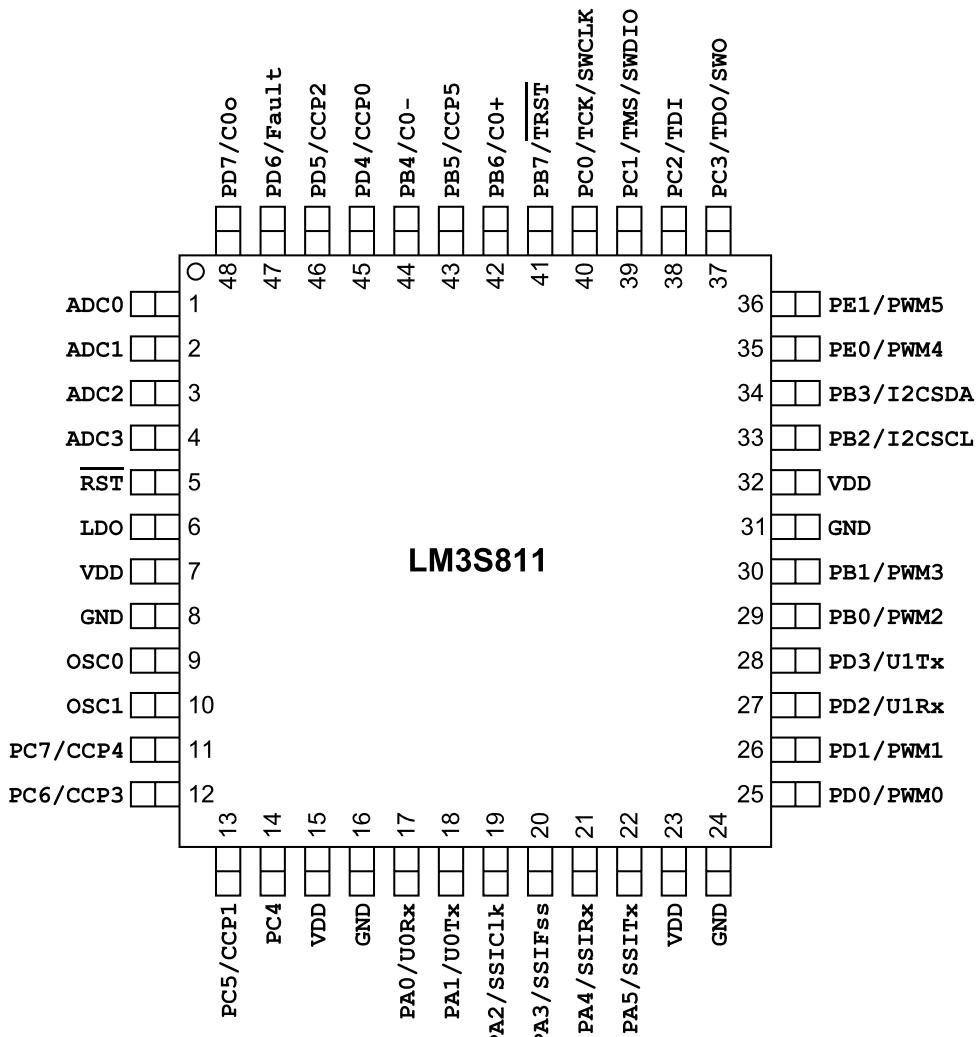
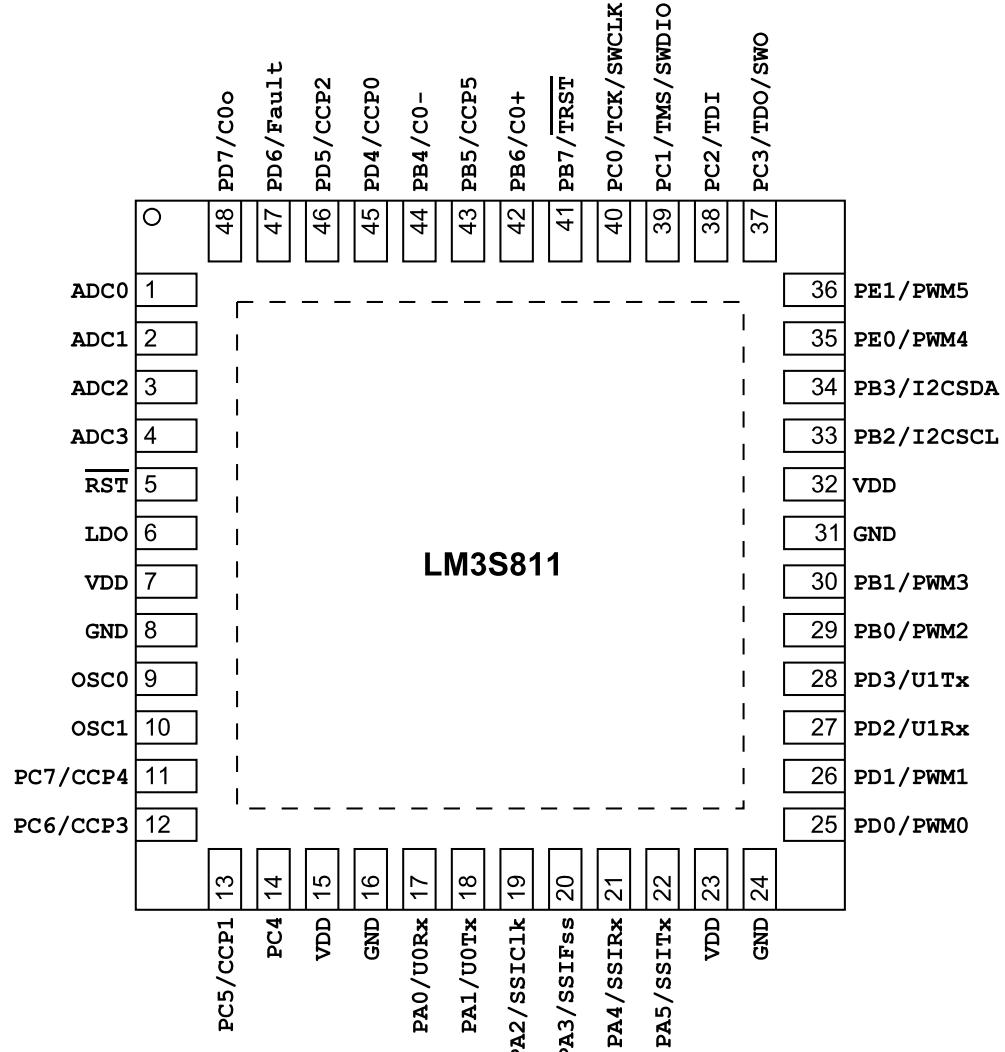


Figure 16-2. 48-Pin QFN Package Pin Diagram<sup>1</sup><sup>1</sup>The thermal pad must be connected to GND.

# 17 Signal Tables

**Important:** All multiplexed pins are GPIOs by default, with the exception of the five JTAG pins ( $\text{PB}7$  and  $\text{PC}[3:0]$ ) which default to the JTAG functionality.

The following tables list the signals available for each pin. Functionality is enabled by software with the **GPIOAFSEL** register. All digital inputs are Schmitt triggered.

- Signals by Pin Number
- Signals by Signal Name
- Signals by Function, Except for GPIO
- GPIO Pins and Alternate Functions
- Connections for Unused Signals

**Important:** The 48-pin QFN package is OBSOLETE. TI has discontinued production of this device.

## 17.1 Signals by Pin Number

Table 17-1. Signals by Pin Number

Pin Number	Pin Name	Pin Type	Buffer Type <sup>a</sup>	Description
1	ADC0	I	Analog	Analog-to-digital converter input 0.
2	ADC1	I	Analog	Analog-to-digital converter input 1.
3	ADC2	I	Analog	Analog-to-digital converter input 2.
4	ADC3	I	Analog	Analog-to-digital converter input 3.
5	RST	I	TTL	System reset input.
6	LDO	-	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.
7	VDD	-	Power	Positive supply for I/O and some logic.
8	GND	-	Power	Ground reference for logic and I/O pins.
9	OSCO	I	Analog	Main oscillator crystal input or an external clock reference input.
10	OSC1	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
11	PC7	I/O	TTL	GPIO port C bit 7.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
12	PC6	I/O	TTL	GPIO port C bit 6.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
13	PC5	I/O	TTL	GPIO port C bit 5.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
14	PC4	I/O	TTL	GPIO port C bit 4.
15	VDD	-	Power	Positive supply for I/O and some logic.
16	GND	-	Power	Ground reference for logic and I/O pins.
17	PA0	I/O	TTL	GPIO port A bit 0.
	U0Rx	I	TTL	UART module 0 receive.
18	PA1	I/O	TTL	GPIO port A bit 1.
	U0Tx	O	TTL	UART module 0 transmit.

**Table 17-1. Signals by Pin Number (continued)**

Pin Number	Pin Name	Pin Type	Buffer Type <sup>a</sup>	Description
19	PA2	I/O	TTL	GPIO port A bit 2.
	SSIClk	I/O	TTL	SSI clock.
20	PA3	I/O	TTL	GPIO port A bit 3.
	SSIFss	I/O	TTL	SSI frame.
21	PA4	I/O	TTL	GPIO port A bit 4.
	SSIRx	I	TTL	SSI receive.
22	PA5	I/O	TTL	GPIO port A bit 5.
	SSITx	O	TTL	SSI transmit.
23	VDD	-	Power	Positive supply for I/O and some logic.
24	GND	-	Power	Ground reference for logic and I/O pins.
25	PDO	I/O	TTL	GPIO port D bit 0.
	PWM0	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
26	PD1	I/O	TTL	GPIO port D bit 1.
	PWM1	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.
27	PD2	I/O	TTL	GPIO port D bit 2.
	U1Rx	I	TTL	UART module 1 receive.
28	PD3	I/O	TTL	GPIO port D bit 3.
	U1Tx	O	TTL	UART module 1 transmit.
29	PB0	I/O	TTL	GPIO port B bit 0.
	PWM2	O	TTL	PWM 2. This signal is controlled by PWM Generator 1.
30	PB1	I/O	TTL	GPIO port B bit 1.
	PWM3	O	TTL	PWM 3. This signal is controlled by PWM Generator 1.
31	GND	-	Power	Ground reference for logic and I/O pins.
32	VDD	-	Power	Positive supply for I/O and some logic.
33	PB2	I/O	TTL	GPIO port B bit 2.
	I2CSCL	I/O	OD	I <sup>2</sup> C clock.
34	PB3	I/O	TTL	GPIO port B bit 3.
	I2CSDA	I/O	OD	I <sup>2</sup> C data.
35	PE0	I/O	TTL	GPIO port E bit 0.
	PWM4	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
36	PE1	I/O	TTL	GPIO port E bit 1.
	PWM5	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
37	PC3	I/O	TTL	GPIO port C bit 3.
	SWO	O	TTL	JTAG TDO and SWO.
	TDO	O	TTL	JTAG TDO and SWO.
38	PC2	I/O	TTL	GPIO port C bit 2.
	TDI	I	TTL	JTAG TDI.
39	PC1	I/O	TTL	GPIO port C bit 1.
	SWDIO	I/O	TTL	JTAG TMS and SWDIO.
	TMS	I/O	TTL	JTAG TMS and SWDIO.

**Table 17-1. Signals by Pin Number (continued)**

Pin Number	Pin Name	Pin Type	Buffer Type <sup>a</sup>	Description
40	PC0	I/O	TTL	GPIO port C bit 0.
	SWCLK	I	TTL	JTAG/SWD CLK.
	TCK	I	TTL	JTAG/SWD CLK.
41	PB7	I/O	TTL	GPIO port B bit 7.
	TRST	I	TTL	JTAG TRST.
42	PB6	I/O	TTL	GPIO port B bit 6.
	C0+	I	Analog	Analog comparator 0 positive input.
43	PB5	I/O	TTL	GPIO port B bit 5.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
44	PB4	I/O	TTL	GPIO port B bit 4.
	C0-	I	Analog	Analog comparator 0 negative input.
45	PD4	I/O	TTL	GPIO port D bit 4.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
46	PD5	I/O	TTL	GPIO port D bit 5.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
47	PD6	I/O	TTL	GPIO port D bit 6.
	Fault	I	TTL	PWM Fault.
48	PD7	I/O	TTL	GPIO port D bit 7.
	C0o	O	TTL	Analog comparator 0 output.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

## 17.2 Signals by Signal Name

**Table 17-2. Signals by Signal Name**

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
ADC0	1	I	Analog	Analog-to-digital converter input 0.
ADC1	2	I	Analog	Analog-to-digital converter input 1.
ADC2	3	I	Analog	Analog-to-digital converter input 2.
ADC3	4	I	Analog	Analog-to-digital converter input 3.
C0+	42	I	Analog	Analog comparator 0 positive input.
C0-	44	I	Analog	Analog comparator 0 negative input.
C0o	48	O	TTL	Analog comparator 0 output.
CCP0	45	I/O	TTL	Capture/Compare/PWM 0.
CCP1	13	I/O	TTL	Capture/Compare/PWM 1.
CCP2	46	I/O	TTL	Capture/Compare/PWM 2.
CCP3	12	I/O	TTL	Capture/Compare/PWM 3.
CCP4	11	I/O	TTL	Capture/Compare/PWM 4.
CCP5	43	I/O	TTL	Capture/Compare/PWM 5.
Fault	47	I	TTL	PWM Fault.

**Table 17-2. Signals by Signal Name (*continued*)**

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
GND	8 16 24 31	-	Power	Ground reference for logic and I/O pins.
I <sup>2</sup> C SCL	33	I/O	OD	I <sup>2</sup> C clock.
I <sup>2</sup> C SDA	34	I/O	OD	I <sup>2</sup> C data.
LDO	6	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater.
OSC0	9	I	Analog	Main oscillator crystal input or an external clock reference input.
OSC1	10	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
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.
PB0	29	I/O	TTL	GPIO port B bit 0.
PB1	30	I/O	TTL	GPIO port B bit 1.
PB2	33	I/O	TTL	GPIO port B bit 2.
PB3	34	I/O	TTL	GPIO port B bit 3.
PB4	44	I/O	TTL	GPIO port B bit 4.
PB5	43	I/O	TTL	GPIO port B bit 5.
PB6	42	I/O	TTL	GPIO port B bit 6.
PB7	41	I/O	TTL	GPIO port B bit 7.
PC0	40	I/O	TTL	GPIO port C bit 0.
PC1	39	I/O	TTL	GPIO port C bit 1.
PC2	38	I/O	TTL	GPIO port C bit 2.
PC3	37	I/O	TTL	GPIO port C bit 3.
PC4	14	I/O	TTL	GPIO port C bit 4.
PC5	13	I/O	TTL	GPIO port C bit 5.
PC6	12	I/O	TTL	GPIO port C bit 6.
PC7	11	I/O	TTL	GPIO port C bit 7.
PD0	25	I/O	TTL	GPIO port D bit 0.
PD1	26	I/O	TTL	GPIO port D bit 1.
PD2	27	I/O	TTL	GPIO port D bit 2.
PD3	28	I/O	TTL	GPIO port D bit 3.
PD4	45	I/O	TTL	GPIO port D bit 4.
PD5	46	I/O	TTL	GPIO port D bit 5.
PD6	47	I/O	TTL	GPIO port D bit 6.
PD7	48	I/O	TTL	GPIO port D bit 7.

**Table 17-2. Signals by Signal Name (*continued*)**

Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
PE0	35	I/O	TTL	GPIO port E bit 0.
PE1	36	I/O	TTL	GPIO port E bit 1.
PWM0	25	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
PWM1	26	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.
PWM2	29	O	TTL	PWM 2. This signal is controlled by PWM Generator 1.
PWM3	30	O	TTL	PWM 3. This signal is controlled by PWM Generator 1.
PWM4	35	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
PWM5	36	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
RST	5	I	TTL	System reset input.
SSIClk	19	I/O	TTL	SSI clock.
SSIFss	20	I/O	TTL	SSI frame.
SSIRx	21	I	TTL	SSI receive.
SSITx	22	O	TTL	SSI transmit.
SWCLK	40	I	TTL	JTAG/SWD CLK.
SWDIO	39	I/O	TTL	JTAG TMS and SWDIO.
SWO	37	O	TTL	JTAG TDO and SWO.
TCK	40	I	TTL	JTAG/SWD CLK.
TDI	38	I	TTL	JTAG TDI.
TDO	37	O	TTL	JTAG TDO and SWO.
TMS	39	I/O	TTL	JTAG TMS and SWDIO.
TRST	41	I	TTL	JTAG TRST.
U0Rx	17	I	TTL	UART module 0 receive.
U0Tx	18	O	TTL	UART module 0 transmit.
U1Rx	27	I	TTL	UART module 1 receive.
U1Tx	28	O	TTL	UART module 1 transmit.
VDD	7 15 23 32	-	Power	Positive supply for I/O and some logic.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

## 17.3 Signals by Function, Except for GPIO

**Table 17-3. Signals by Function, Except for GPIO**

Function	Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
ADC	ADC0	1	I	Analog	Analog-to-digital converter input 0.
	ADC1	2	I	Analog	Analog-to-digital converter input 1.
	ADC2	3	I	Analog	Analog-to-digital converter input 2.
	ADC3	4	I	Analog	Analog-to-digital converter input 3.
Analog Comparators	C0+	42	I	Analog	Analog comparator 0 positive input.
	C0-	44	I	Analog	Analog comparator 0 negative input.
	C0o	48	O	TTL	Analog comparator 0 output.

**Table 17-3. Signals by Function, Except for GPIO (continued)**

Function	Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
General-Purpose Timers	CCP0	45	I/O	TTL	Capture/Compare/PWM 0.
	CCP1	13	I/O	TTL	Capture/Compare/PWM 1.
	CCP2	46	I/O	TTL	Capture/Compare/PWM 2.
	CCP3	12	I/O	TTL	Capture/Compare/PWM 3.
	CCP4	11	I/O	TTL	Capture/Compare/PWM 4.
	CCP5	43	I/O	TTL	Capture/Compare/PWM 5.
I2C	I2CSCL	33	I/O	OD	I <sup>2</sup> C clock.
	I2CSDA	34	I/O	OD	I <sup>2</sup> C data.
JTAG/SWD/SWO	SWCLK	40	I	TTL	JTAG/SWD CLK.
	SWDIO	39	I/O	TTL	JTAG TMS and SWDIO.
	SWO	37	O	TTL	JTAG TDO and SWO.
	TCK	40	I	TTL	JTAG/SWD CLK.
	TDI	38	I	TTL	JTAG TDI.
	TDO	37	O	TTL	JTAG TDO and SWO.
	TMS	39	I/O	TTL	JTAG TMS and SWDIO.
	TRST	41	I	TTL	JTAG TRST.
PWM	Fault	47	I	TTL	PWM Fault.
	PWM0	25	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
	PWM1	26	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.
	PWM2	29	O	TTL	PWM 2. This signal is controlled by PWM Generator 1.
	PWM3	30	O	TTL	PWM 3. This signal is controlled by PWM Generator 1.
	PWM4	35	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
	PWM5	36	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
Power	GND	8 16 24 31	-	Power	Ground reference for logic and I/O pins.
	LDO	6	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 $\mu$ F or greater.
	VDD	7 15 23 32	-	Power	Positive supply for I/O and some logic.
SSI	SSIClk	19	I/O	TTL	SSI clock.
	SSIFFS	20	I/O	TTL	SSI frame.
	SSIRx	21	I	TTL	SSI receive.
	SSITx	22	O	TTL	SSI transmit.

**Table 17-3. Signals by Function, Except for GPIO (continued)**

Function	Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
System Control & Clocks	OSC0	9	I	Analog	Main oscillator crystal input or an external clock reference input.
	OSC1	10	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
	RST	5	I	TTL	System reset input.
UART	U0Rx	17	I	TTL	UART module 0 receive.
	U0Tx	18	O	TTL	UART module 0 transmit.
	U1Rx	27	I	TTL	UART module 1 receive.
	U1Tx	28	O	TTL	UART module 1 transmit.

a. The TTL designation indicates the pin has TTL-compatible voltage levels.

## 17.4 GPIO Pins and Alternate Functions

**Table 17-4. GPIO Pins and Alternate Functions**

IO	Pin Number	Multiplexed Function	Multiplexed Function
PA0	17	U0Rx	
PA1	18	U0Tx	
PA2	19	SSIClk	
PA3	20	SSIFss	
PA4	21	SSIRx	
PA5	22	SSITx	
PB0	29	PWM2	
PB1	30	PWM3	
PB2	33	I2CSCL	
PB3	34	I2CSDA	
PB4	44	C0-	
PB5	43	CCP5	
PB6	42	C0+	
PB7	41	TRST	
PC0	40	TCK	SWCLK
PC1	39	TMS	SWDIO
PC2	38	TDI	
PC3	37	TDO	SWO
PC4	14		
PC5	13	CCP1	
PC6	12	CCP3	
PC7	11	CCP4	
PD0	25	PWM0	
PD1	26	PWM1	
PD2	27	U1Rx	
PD3	28	U1Tx	
PD4	45	CCP0	

**Table 17-4. GPIO Pins and Alternate Functions (continued)**

IO	Pin Number	Multiplexed Function	Multiplexed Function
PD5	46	CCP2	
PD6	47	Fault	
PD7	48	C0o	
PE0	35	PWM4	
PE1	36	PWM5	

## 17.5 Connections for Unused Signals

Table 17-5 on page 548 show how to handle signals for functions that are not used in a particular system implementation. Two options are shown in the table: an acceptable practice and a preferred practice for reduced power consumption and improved EMC characteristics.

**Table 17-5. Connections for Unused Signals**

Function	Signal Name	Pin Number	Acceptable Practice	Preferred Practice
ADC	ADC0	1	NC	GNDA
	ADC1	2		
	ADC2	3		
	ADC3	4		
GPIO	All unused GPIOs	-	NC	GND
System Control	OSC0	9	NC	GND
	OSC1	10	NC	NC
	$\overline{RST}$	5	Pull up as shown in Figure 5-1 on page 157	Connect through a capacitor to GND as close to pin as possible

# 18 Operating Characteristics

**Table 18-1. Temperature Characteristics**

Characteristic	Symbol	Value	Unit
Industrial operating temperature range	T <sub>A</sub>	-40 to +85	°C
Unpowered storage temperature range	T <sub>S</sub>	-65 to +150	°C

**Table 18-2. Thermal Characteristics**

Characteristic	Symbol	Value	Unit
Thermal resistance (junction to ambient) <sup>a</sup>	θ <sub>JA</sub>	50 (48-pin QFP) 26 (48-pin QFN)	°C/W
Junction temperature <sup>b</sup>	T <sub>J</sub>	T <sub>A</sub> + (P • θ <sub>JA</sub> )	°C
Maximum junction temperature	T <sub>JMAX</sub>	115 <sub>c</sub>	°C

a. Junction to ambient thermal resistance θ<sub>JA</sub> numbers are determined by a package simulator.

b. Power dissipation is a function of temperature.

c. T<sub>JMAX</sub> calculation is based on power consumption values and conditions as specified in "Power Specifications".

**Table 18-3. ESD Absolute Maximum Ratings<sup>a</sup>**

Parameter Name	Min	Nom	Max	Unit
V <sub>ESDHBM</sub>	-	-	2.0	kV
V <sub>ESDCDM</sub>	-	-	1.0	kV
V <sub>ESDMM</sub>	-	-	100	V

a. All Stellaris parts are ESD tested following the JEDEC standard.

# 19 Electrical Characteristics

## 19.1 DC Characteristics

### 19.1.1 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 19-1. Maximum Ratings**

Characteristic <sup>a</sup>	Symbol	Value	Unit
Supply voltage range ( $V_{DD}$ )	$V_{DD}$	0.0 to +3.6	V
Input voltage	$V_{IN}$	-0.3 to 5.5	V
Input voltage for a GPIO configured as an analog input		-0.3 to $V_{DD} + 0.3$	V
Maximum current for pins, excluding pins operating as GPIOs	I	100	mA
Maximum current for GPIO pins	I	100	mA
Maximum input voltage on a non-power pin when the microcontroller is unpowered	$V_{NON}$	300	mV

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  $V_{DD}$ ).

### 19.1.2 Recommended DC Operating Conditions

**Table 19-2. Recommended DC Operating Conditions**

Parameter	Parameter Name	Min	Nom	Max	Unit
$V_{DD}$	Supply voltage	3.0	3.3	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
$V_{OH}$	High-level output voltage	2.4	-	-	V
$V_{OL}$	Low-level output voltage	-	-	0.4	V
$I_{OH}$	High-level source current, $V_{OH}=2.4\text{ V}$				
	2-mA Drive	2.0	-	-	mA
	4-mA Drive	4.0	-	-	mA
	8-mA Drive	8.0	-	-	mA
$I_{OL}$	Low-level sink current, $V_{OL}=0.4\text{ V}$				
	2-mA Drive	2.0	-	-	mA
	4-mA Drive	4.0	-	-	mA
	8-mA Drive	8.0	-	-	mA

### 19.1.3 On-Chip Low Drop-Out (LDO) Regulator Characteristics

**Table 19-3. LDO Regulator Characteristics**

Parameter	Parameter Name	Min	Nom	Max	Unit
$V_{LDOOUT}$	Programmable internal (logic) power supply output value	2.25	-	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	$\mu F$

### 19.1.4 GPIO Module Characteristics

**Table 19-4. GPIO Module DC Characteristics**

Parameter	Parameter Name	Min	Nom	Max	Unit
$R_{GPIOPU}$	GPIO internal pull-up resistor	50	-	110	k $\Omega$
$R_{GPIOPD}$	GPIO internal pull-down resistor	55	-	180	k $\Omega$
$I_{LKG}$	GPIO input leakage current <sup>a</sup>	-	-	2	$\mu A$

a. The leakage current is measured with GND or  $V_{DD}$  applied to the corresponding pin(s). The leakage of digital port pins is measured individually. The port pin is configured as an input and the pullup/pulldown resistor is disabled.

### 19.1.5 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
- Temperature = 25°C

**Table 19-5. Detailed Power Specifications**

Parameter	Parameter Name	Conditions	Nom	Max	Unit
$I_{DD\_RUN}$	Run mode 1 (Flash loop)	LDO = 2.50 V Code = while(1){} executed out of Flash Peripherals = All clock-gated ON System Clock = 50 MHz (with PLL)	95	110	mA
	Run mode 2 (Flash loop)	LDO = 2.50 V Code = while(1){} executed out of Flash Peripherals = All clock-gated OFF System Clock = 50 MHz (with PLL)	60	75	mA
	Run mode 1 (SRAM loop)	LDO = 2.50 V Code = while(1){} executed in SRAM Peripherals = All clock-gated ON System Clock = 50 MHz (with PLL)	85	95	mA
	Run mode 2 (SRAM loop)	LDO = 2.50 V Code = while(1){} executed in SRAM Peripherals = All clock-gated OFF System Clock = 50 MHz (with PLL)	50	60	mA
$I_{DD\_SLEEP}$	Sleep mode	LDO = 2.50 V Peripherals = All clock-gated OFF System Clock = 50 MHz (with PLL)	19	22	mA
$I_{DD\_DEEPSLEEP}$	Deep-Sleep mode	LDO = 2.25 V Peripherals = All OFF System Clock = MOSC/16	950	1150	$\mu$ A

### 19.1.6 Flash Memory Characteristics

**Table 19-6. Flash Memory Characteristics**

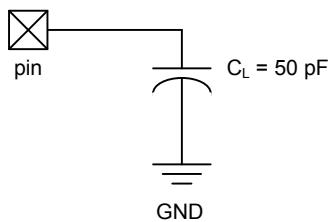
Parameter	Parameter Name	Min	Nom	Max	Unit
$PE_{CYC}$	Number of guaranteed program/erase cycles before failure <sup>a</sup>	10,000	100,000	-	cycles
$T_{RET}$	Data retention at average operating temperature of 85°C	10	-	-	years
$T_{PROG}$	Word program time	20	-	-	$\mu$ s
$T_{ERASE}$	Page erase time	20	-	-	ms
$T_{ME}$	Mass erase time	-	-	250	ms

a. A program/erase cycle is defined as switching the bits from 1-> 0 -> 1.

## 19.2 AC Characteristics

### 19.2.1 Load Conditions

Unless otherwise specified, the following conditions are true for all timing measurements. Timing measurements are for 4-mA drive strength.

**Figure 19-1. Load Conditions**

## 19.2.2 Clocks

**Table 19-7. Phase Locked Loop (PLL) Characteristics**

Parameter	Parameter Name	Min	Nom	Max	Unit
$f_{\text{ref\_crystal}}$	Crystal reference <sup>a</sup>	3.579545	-	8.192	MHz
$f_{\text{ref\_ext}}$	External clock reference <sup>a</sup>	3.579545	-	8.192	MHz
$f_{\text{pll}}$	PLL frequency <sup>b</sup>	-	200	-	MHz
$T_{\text{READY}}$	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 19-8. Clock Characteristics**

Parameter	Parameter Name	Min	Nom	Max	Unit
$f_{\text{IOsc}}$	Internal oscillator frequency	7	12	22	MHz
$f_{\text{Mosc}}$	Main oscillator frequency	1	-	8	MHz
$t_{\text{Mosc\_per}}$	Main oscillator period	125	-	1000	ns
$f_{\text{ref\_crystal\_bypass}}$	Crystal reference using the main oscillator (PLL in BYPASS mode) <sup>a</sup>	1	-	8	MHz
$f_{\text{ref\_ext\_bypass}}$	External clock reference (PLL in BYPASS mode) <sup>a</sup>	0	-	50	MHz
$f_{\text{system\_clock}}$	System clock	0	-	50	MHz

a. The ADC must be clocked from the PLL or directly from a 16.667-MHz clock source to operate properly.

### 19.2.2.1 System Clock Specifications with ADC Operation

**Table 19-9. System Clock Characteristics with ADC Operation**

Parameter	Parameter Name	Min	Nom	Max	Unit
$f_{\text{sysadc}}$	System clock frequency when the ADC module is operating (when PLL is bypassed)	16	-	-	MHz

## 19.2.3 JTAG and Boundary Scan

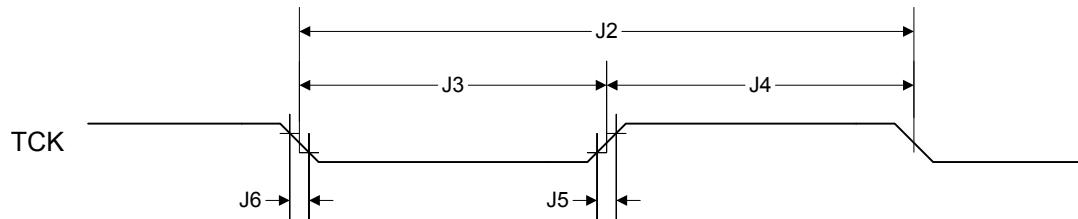
**Table 19-10. JTAG Characteristics**

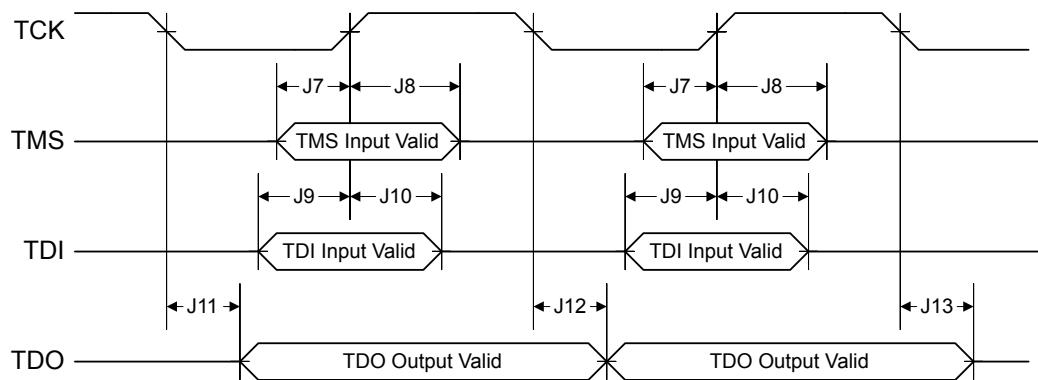
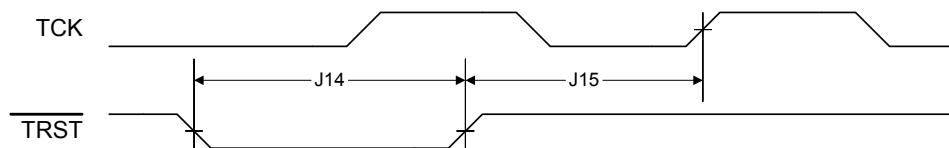
Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
J1	$f_{\text{TCK}}$	TCK operational clock frequency	0	-	10	MHz
J2	$t_{\text{TCK}}$	TCK operational clock period	100	-	-	ns

Table 19-10. JTAG Characteristics (continued)

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
J3	$t_{TCK\_LOW}$	TCK clock Low time	-	$t_{TCK}/2$	-	ns
J4	$t_{TCK\_HIGH}$	TCK clock High time	-	$t_{TCK}/2$	-	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	2-mA drive	-	23	35	ns
		4-mA drive		15	26	ns
		8-mA drive		14	25	ns
		8-mA drive with slew rate control		18	29	ns
J12 $t_{TDO\_DV}$	TCK fall to Data Valid from Data Valid	2-mA drive	-	21	35	ns
		4-mA drive		14	25	ns
		8-mA drive		13	24	ns
		8-mA drive with slew rate control		18	28	ns
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
J14	$t_{TRST}$	TRST assertion time	100	-	-	ns
J15	$t_{TRST\_SU}$	TRST setup time to TCK rise	10	-	-	ns

Figure 19-2. JTAG Test Clock Input Timing



**Figure 19-3. JTAG Test Access Port (TAP) Timing****Figure 19-4. JTAG TRST Timing**

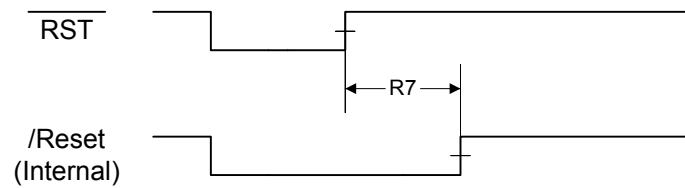
## 19.2.4 Reset

**Table 19-11. Reset Characteristics**

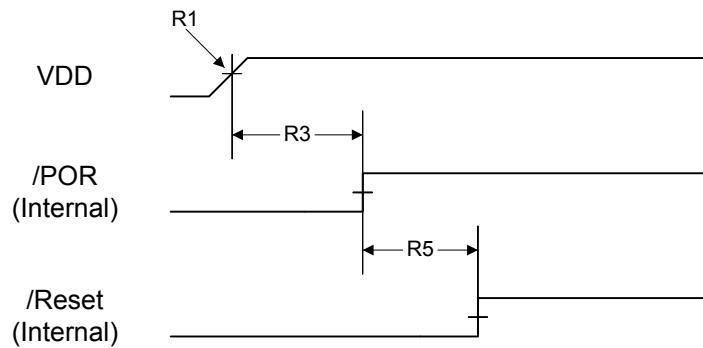
Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
R1	$V_{TH}$	Reset threshold	-	2.0	-	V
R2	$V_{BTH}$	Brown-Out threshold	2.85	2.9	2.95	V
R3	$T_{POR}$	Power-On Reset timeout	-	10	-	ms
R4	$T_{BOR}$	Brown-Out timeout	-	500	-	$\mu$ s
R5	$T_{IRPOR}$	Internal reset timeout after POR	15	-	30	ms
R6	$T_{IRBOR}$	Internal reset timeout after BOR <sup>a</sup>	2.5	-	20	$\mu$ s
R7	$T_{IRHWR}$	Internal reset timeout after hardware reset ( $RST$ pin)	2.9	-	29	$\mu$ s
R8	$T_{IRSWR}$	Internal reset timeout after software-initiated system reset <sup>a</sup>	2.5	-	20	$\mu$ s
R9	$T_{IRWDR}$	Internal reset timeout after watchdog reset <sup>a</sup>	2.5	-	20	$\mu$ s
R10	$T_{IRLDOR}$	Internal reset timeout after LDO reset <sup>a</sup>	2.5	-	20	$\mu$ s
R11	$T_{VDDRISE}$	Supply voltage ( $V_{DD}$ ) rise time (0 V-3.3 V)	-	-	100	ms

a.  $20 * t_{MOSC\_per}$

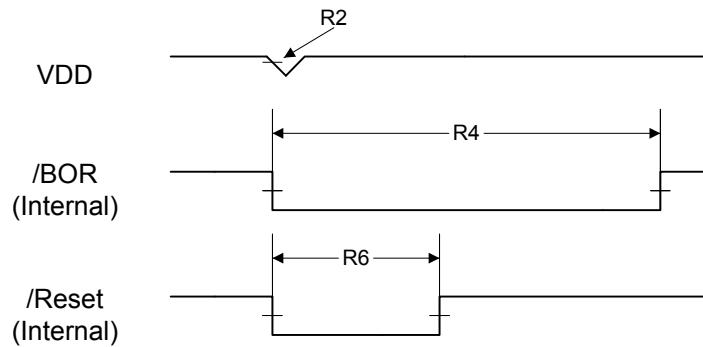
**Figure 19-5. External Reset Timing ( $\overline{\text{RST}}$ )**

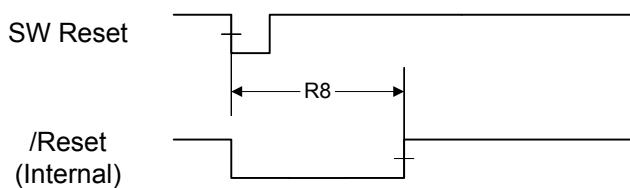
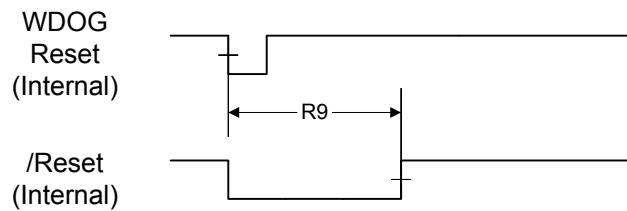
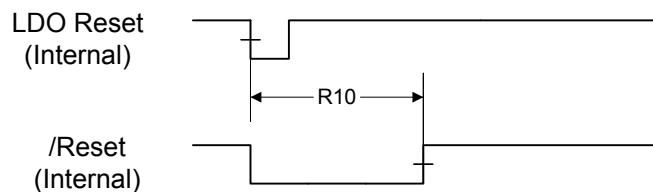


**Figure 19-6. Power-On Reset Timing**



**Figure 19-7. Brown-Out Reset Timing**



**Figure 19-8. Software Reset Timing****Figure 19-9. Watchdog Reset Timing****Figure 19-10. LDO Reset Timing**

### 19.2.5 Sleep Modes

**Table 19-12. Sleep Modes AC Characteristics<sup>a</sup>**

Parameter No	Parameter	Parameter Name	Min	Nom	Max	Unit
D1	$t_{WAKE\_S}$	Time to wake from interrupt in sleep or deep-sleep mode, not using the PLL	-	-	7	system clocks
D2	$t_{WAKE\_PLL\_S}$	Time to wake from interrupt in sleep or deep-sleep mode when using the PLL	-	-	$T_{READY}$	ms

a. Values in this table assume the IOSC is the clock source during sleep or deep-sleep mode.

### 19.2.6 General-Purpose I/O (GPIO)

**Note:** All GPIOs are 5 V-tolerant.

**Table 19-13. 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

**19.2.7 Analog-to-Digital Converter****Table 19-14. ADC Characteristics<sup>a</sup>**

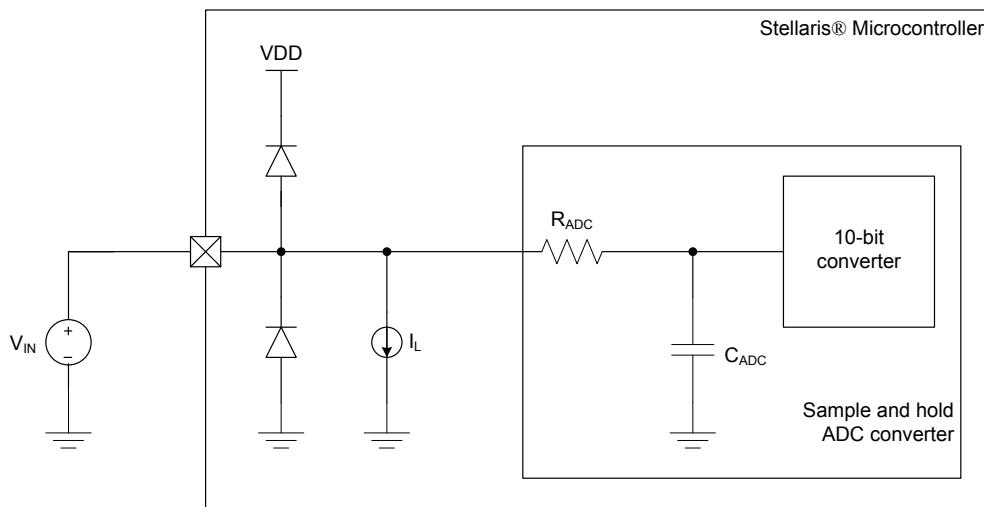
Parameter	Parameter Name	Min	Nom	Max	Unit
$V_{ADCIN}$	Maximum single-ended, full-scale analog input voltage	-	-	3.0	V
	Minimum single-ended, full-scale analog input voltage	0.0	-	-	V
	Maximum differential, full-scale analog input voltage	-	-	1.5	V
	Minimum differential, full-scale analog input voltage	0.0	-	-	V
N	Resolution		10		bits
$f_{ADC}$	ADC internal clock frequency <sup>b</sup>	14	16.667	18	MHz
$t_{ADCCONV}$	Conversion time <sup>c</sup>		2		μs
$f_{ADCCONV}$	Conversion rate <sup>c</sup>		520.833		k samples/s
$t_{LT}$	Latency from trigger to start of conversion	-	2	-	system clocks
$I_L$	ADC input leakage	-	-	±3.0	μA
$R_{ADC}$	ADC equivalent resistance	-	-	10	kΩ
$C_{ADC}$	ADC equivalent capacitance	0.9	1.0	1.1	pF
$E_L$	Integral nonlinearity error	-	-	±3	LSB
$E_D$	Differential nonlinearity error	-	-	±2	LSB
$E_O$	Offset error	-	-	+6 <sup>d</sup>	LSB
$E_G$	Full-scale gain error	-	-	±3	LSB
$E_{TS}$	Temperature sensor accuracy	-	-	±5	°C

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. The ADC must be clocked from the PLL or directly from an external clock source to operate properly.

c. The conversion time and rate scale from the specified number if the ADC internal clock frequency is any value other than 16.667 MHz.

d. The offset error listed above is the conversion result with 0 V applied to the ADC input.

**Figure 19-11. ADC Input Equivalency Diagram****Table 19-15. ADC Module Internal Reference Characteristics**

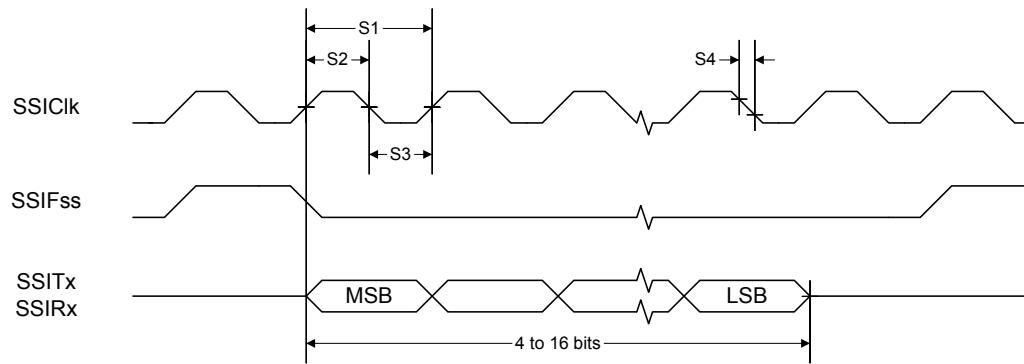
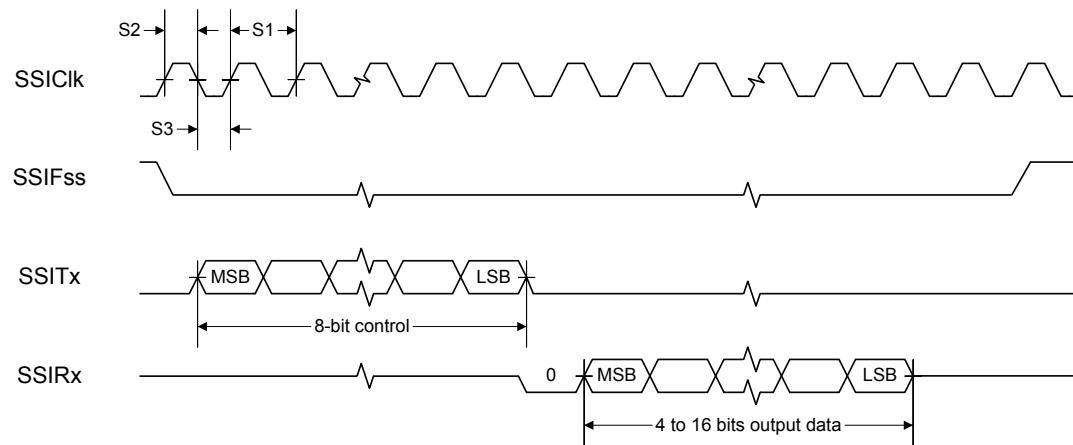
Parameter	Parameter Name	Min	Nom	Max	Unit
$V_{REFI}$	Internal voltage reference for ADC	-	3.0	-	V
$E_{IR}$	Internal voltage reference error	-	-	$\pm 2.5$	%

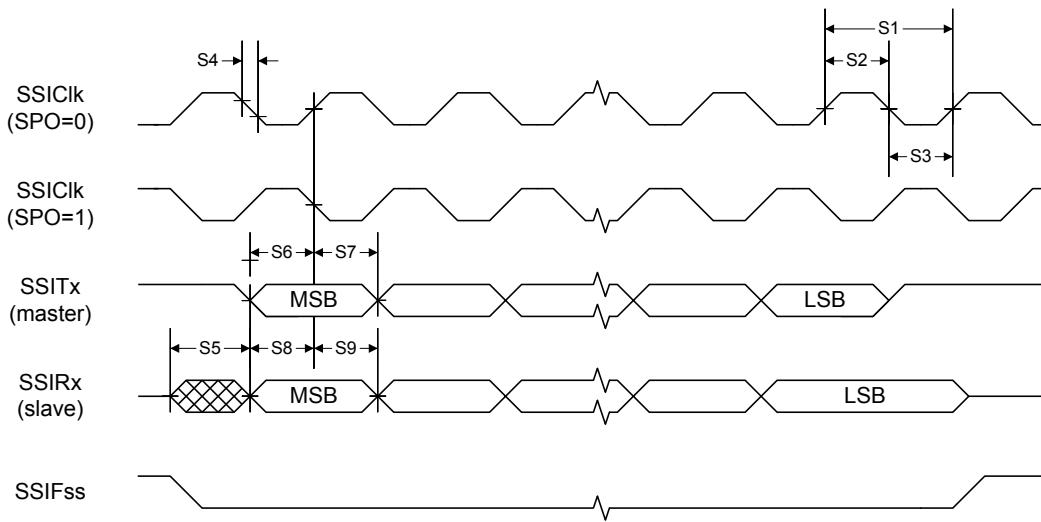
### 19.2.8 Synchronous Serial Interface (SSI)

**Table 19-16. 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	-	0.5	-	$t_{clk\_per}$
S3	$t_{clk\_low}$	SSIClk low time	-	0.5	-	$t_{clk\_per}$
S4	$t_{clkrf}$	SSIClk rise/fall time <sup>a</sup>	-	6	10	ns
S5	$t_{DMd}$	Data from master valid delay time	0	-	1	system clocks
S6	$t_{DMs}$	Data from master setup time	1	-	-	system clocks
S7	$t_{DMh}$	Data from master hold time	2	-	-	system clocks
S8	$t_{DSS}$	Data from slave setup time	1	-	-	system clocks
S9	$t_{DSh}$	Data from slave hold time	2	-	-	system clocks

a. Note that the delays shown are using 8-mA drive strength.

**Figure 19-12. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement****Figure 19-13. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer**

**Figure 19-14. SSI Timing for SPI Frame Format (FRF=00), with SPH=1**

### 19.2.9 Inter-Integrated Circuit (I<sup>2</sup>C) Interface

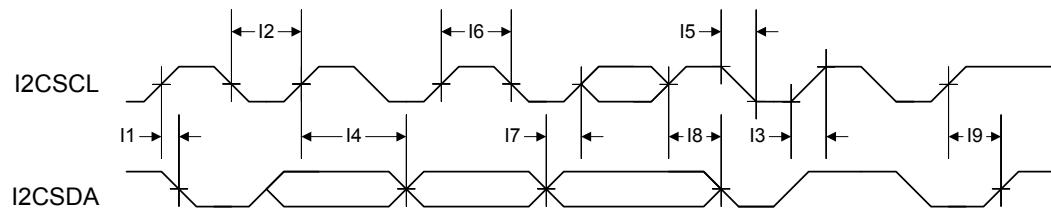
**Table 19-17. I<sup>2</sup>C Characteristics**

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
I1 <sup>a</sup>	t <sub>SCH</sub>	Start condition hold time	36	-	-	system clocks
I2 <sup>a</sup>	t <sub>LP</sub>	Clock Low period	36	-	-	system clocks
I3 <sup>b</sup>	t <sub>SRT</sub>	I <sup>2</sup> CSCL/I <sup>2</sup> CSDA rise time ( $V_{IL} = 0.5\text{ V}$ to $V_{IH} = 2.4\text{ V}$ )	-	-	(see note b)	ns
I4 <sup>a</sup>	t <sub>DH</sub>	Data hold time	2	-	-	system clocks
I5 <sup>c</sup>	t <sub>SFT</sub>	I <sup>2</sup> CSCL/I <sup>2</sup> CSDA fall time ( $V_{IH} = 2.4\text{ V}$ to $V_{IL} = 0.5\text{ V}$ )	-	9	10	ns
I6 <sup>a</sup>	t <sub>HT</sub>	Clock High time	24	-	-	system clocks
I7 <sup>a</sup>	t <sub>DS</sub>	Data setup time	18	-	-	system clocks
I8 <sup>a</sup>	t <sub>SCSR</sub>	Start condition setup time (for repeated start condition only)	36	-	-	system clocks
I9 <sup>a</sup>	t <sub>SCS</sub>	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 (I<sup>2</sup>CMTPR)** register; a TPR programmed for the maximum I<sup>2</sup>CSCL 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>CSCL 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>CSCL and I<sup>2</sup>CSDA are open-drain-type outputs, which the controller can only actively drive Low, the time I<sup>2</sup>CSCL or I<sup>2</sup>CSDA 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 19-15. I<sup>2</sup>C Timing**

### 19.2.10 Analog Comparator

**Table 19-18. Analog Comparator Characteristics**

Parameter	Parameter Name	Min	Nom	Max	Unit
$V_{OS}$	Input offset voltage	-	$\pm 10$	$\pm 25$	mV
$V_{CM}$	Input common mode voltage range	0	-	$V_{DD}-1.5$	V
$C_{MRR}$	Common mode rejection ratio	50	-	-	dB
$T_{RT}$	Response time	-	-	1	$\mu s$
$T_{MC}$	Comparator mode change to Output Valid	-	-	10	$\mu s$

**Table 19-19. Analog Comparator Voltage Reference Characteristics**

Parameter	Parameter Name	Min	Nom	Max	Unit
$R_{HR}$	Resolution high range	-	$V_{DD}/31$	-	LSB
$R_{LR}$	Resolution low range	-	$V_{DD}/23$	-	LSB
$A_{HR}$	Absolute accuracy high range	-	-	$\pm 1/2$	LSB
$A_{LR}$	Absolute accuracy low range	-	-	$\pm 1/4$	LSB

# A Serial Flash Loader

## A.1 Serial Flash Loader

The Stellaris® serial flash loader is a preprogrammed flash-resident utility used to download code to the flash memory of a device without the use of a debug interface. The serial flash loader uses a simple packet interface to provide synchronous communication with the device. The flash loader runs off the crystal and does not enable the PLL, so its speed is determined by the crystal used. The two serial interfaces that can be used are the UART0 and SSI0 interfaces. For simplicity, both the data format and communication protocol are identical for both serial interfaces.

## A.2 Interfaces

Once communication with the flash loader is established via one of the serial interfaces, that interface is used until the flash loader is reset or new code takes over. For example, once you start communicating using the SSI port, communications with the flash 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 flash 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 crystal frequency of the board that is running the serial flash loader. This is actually the same as the hardware limitation for the maximum baud rate for any UART on a Stellaris device which is calculated as follows:

$$\text{Max Baud Rate} = \text{System Clock Frequency} / 16$$

In order to determine the baud rate, the serial flash loader needs to determine the relationship between its own crystal frequency and the baud rate. This is enough information for the flash 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 flash loader two bytes that are both 0x55. This generates a series of pulses to the flash 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 flash 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 flash loader acknowledges that it has received a synchronization pattern correctly. For example, the time to wait for data back from the flash loader should be calculated as at least  $2 * (20(\text{bits}/\text{sync})/\text{baud rate} (\text{bits/sec}))$ . For a baud rate of 115200, this time is  $2 * (20/115200)$  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 415 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 crystal frequency of the board running

the flash loader. Since the host device is the master, the SSI on the flash 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 serial flash loader command, COMMAND\_SEND\_DATA (see “COMMAND\_SEND\_DATA (0x24)” on page 566).

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 flash loader sends a packet of data in the same format that it receives a packet. The flash 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 flash loader. Once the device communicating with the flash 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 flash 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 flash loader, as the

flash 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 flash loader.

## A.4 Commands

The next section defines the list of commands that can be sent to the flash 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 flash 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 flash 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 flash 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 flash 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. The caller should limit transfers of data to a maximum 8 bytes of packet data to allow the flash to program successfully and not overflow input buffers of the serial interfaces. 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 flash loader sends a NAK to this command, the flash loader does not increment the current address to allow retransmission of the previous 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 flash 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 flash 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 flash loader device to reset. This is useful when downloading a new image that overwrote the flash loader and wants to start from a full 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 flash loader if a critical error occurs and the host device wants to restart communication with the flash loader.

```
Byte[0] = 3
Byte[1] = checksum(Byte[2])
Byte[2] = COMMAND_RESET
```

The flash loader responds with an ACK signal back to the host device before actually executing the software reset to the device running the flash loader. This allows the host to know that the command was received successfully and the part will be reset.

# B 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>The Cortex-M3 Processor</b>															
R0, type R/W, , reset - (see page 51)															
DATA															
R1, type R/W, , reset - (see page 51)															
DATA															
R2, type R/W, , reset - (see page 51)															
DATA															
R3, type R/W, , reset - (see page 51)															
DATA															
R4, type R/W, , reset - (see page 51)															
DATA															
R5, type R/W, , reset - (see page 51)															
DATA															
R6, type R/W, , reset - (see page 51)															
DATA															
R7, type R/W, , reset - (see page 51)															
DATA															
R8, type R/W, , reset - (see page 51)															
DATA															
R9, type R/W, , reset - (see page 51)															
DATA															
R10, type R/W, , reset - (see page 51)															
DATA															
R11, type R/W, , reset - (see page 51)															
DATA															
R12, type R/W, , reset - (see page 51)															
DATA															
SP, type R/W, , reset - (see page 52)															
SP															
LR, type R/W, , reset 0xFFFF.FFFF (see page 53)															
LINK															
PC, type R/W, , reset - (see page 54)															
PC															

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>PSR</b> , type R/W, , reset 0x0100.0000 (see page 55)															
N	Z	C	V	Q	ICI / IT	THUMB									
		ICI / IT													ISRNUM
<b>PRIMASK</b> , type R/W, , reset 0x0000.0000 (see page 59)															
															PRIMASK
<b>FAULTMASK</b> , type R/W, , reset 0x0000.0000 (see page 60)															
															FAULTMASK
<b>BASEPRI</b> , type R/W, , reset 0x0000.0000 (see page 61)															
															BASEPRI
<b>CONTROL</b> , type R/W, , reset 0x0000.0000 (see page 62)															
															ASP TMPL
<b>Cortex-M3 Peripherals</b>															
<b>System Timer (SysTick) Registers</b>															
Base 0xE000.E000															
<b>STCTRL</b> , type R/W, offset 0x010, reset 0x0000.0000															
															COUNT
															CLK_SRC INTEN ENABLE
<b>STRELOAD</b> , type R/W, offset 0x014, reset 0x0000.0000															
															RELOAD
															RELOAD
<b>STCURRENT</b> , type R/WC, offset 0x018, reset 0x0000.0000															
															CURRENT
															CURRENT
<b>Cortex-M3 Peripherals</b>															
<b>Nested Vectored Interrupt Controller (NVIC) Registers</b>															
Base 0xE000.E000															
<b>EN0</b> , type R/W, offset 0x100, reset 0x0000.0000															
															INT
															INT
<b>DIS0</b> , type R/W, offset 0x180, reset 0x0000.0000															
															INT
															INT
<b>PEND0</b> , type R/W, offset 0x200, reset 0x0000.0000															
															INT
															INT
<b>UNPEND0</b> , type R/W, offset 0x280, reset 0x0000.0000															
															INT
															INT
<b>ACTIVE0</b> , type RO, offset 0x300, reset 0x0000.0000															
															INT
															INT
<b>PRI0</b> , type R/W, offset 0x400, reset 0x0000.0000															
INTD															INTC
INTB															INTA
<b>PRI1</b> , type R/W, offset 0x404, reset 0x0000.0000															
INTD															INTC
INTB															INTA

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>PRI2, type R/W, offset 0x408, reset 0x0000.0000</b>																							
INTD								INTC															
INTB								INTA															
<b>PRI3, type R/W, offset 0x40C, reset 0x0000.0000</b>																							
INTD								INTC															
INTB								INTA															
<b>PRI4, type R/W, offset 0x410, reset 0x0000.0000</b>																							
INTD								INTC															
INTB								INTA															
<b>PRI5, type R/W, offset 0x414, reset 0x0000.0000</b>																							
INTD								INTC															
INTB								INTA															
<b>PRI6, type R/W, offset 0x418, reset 0x0000.0000</b>																							
INTD								INTC															
INTB								INTA															
<b>PRI7, type R/W, offset 0x41C, reset 0x0000.0000</b>																							
INTD								INTC															
INTB								INTA															
<b>SWTRIG, type WO, offset 0xF00, reset 0x0000.0000</b>																							
															INTID								
<b>Cortex-M3 Peripherals</b>																							
<b>System Control Block (SCB) Registers</b>																							
Base 0xE000.E000																							
<b>CPUID, type RO, offset 0xD00, reset 0x410F.C231</b>																							
IMP								VAR				CON											
PARTNO												REV											
<b>INTCTRL, type R/W, offset 0xD04, reset 0x0000.0000</b>																							
NMISET			PENDSV	UNPENDSV	PENDSTSET	PENDSTCLR		ISRPRE	ISRPEND						VECPEND								
VECPEND				RETBASE									VECACT										
<b>VTABLE, type R/W, offset 0xD08, reset 0x0000.0000</b>																							
BASE				OFFSET																			
OFFSET																							
<b>APINT, type R/W, offset 0xD0C, reset 0xFA05.0000</b>																							
VECTKEY																							
_ENDIANESS								PRIGROUP						SYSRESREQ	VECTCLRACT	VECTRESET							
<b>SYSCTRL, type R/W, offset 0xD10, reset 0x0000.0000</b>																							
													SEVONPEND	SLEEPDEEP	SLEEPEXIT								
<b>CFGCTRL, type R/W, offset 0xD14, reset 0x0000.0000</b>																							
								STKALIGN	BFHFNIGN				DIV0	UNALIGNED	MAINPEND	BASETHR							
<b>SYSPRI1, type R/W, offset 0xD18, reset 0x0000.0000</b>																							
USAGE																							
BUS				MEM																			
<b>SYSPRI2, type R/W, offset 0xD1C, reset 0x0000.0000</b>																							
SVC																							
<b>SYSPRI3, type R/W, offset 0xD20, reset 0x0000.0000</b>																							
TICK								PENDSV															
								DEBUG															

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>SYSHNDCTRL</b> , type R/W, offset 0xD24, reset 0x0000.0000																
SVC	BUSP	MEMP	USAGEP	TICK	PNDSV		MON	SVCA				USGA		USAGE	BUS	MEM
<b>FAULTSTAT</b> , type R/W1C, offset 0xD28, reset 0x0000.0000																
BFARV		BSTKE	BUSTKE	IMPRE	PRECISE	IBUS	MMARV			MSTKE	MUSTKE			DERR	IERR	
<b>HFAULTSTAT</b> , type R/W1C, offset 0xD2C, reset 0x0000.0000																
DBG	FORCED														VECT	
<b>MMADDR</b> , type R/W, offset 0xD34, reset -																
ADDR																
ADDR																
<b>FAULTADDR</b> , type R/W, offset 0xD38, reset -																
ADDR																
ADDR																
<b>Cortex-M3 Peripherals</b>																
<b>Memory Protection Unit (MPU) Registers</b>																
Base 0xE000.E000																
<b>MPUTYPE</b> , type RO, offset 0xD90, reset 0x0000.0800																
IREGION																
DREGION																
<b>MPUCTRL</b> , type R/W, offset 0xD94, reset 0x0000.0000																
PRIVDEFEN																
HFNMIENA																
<b>MPUNUMBER</b> , type R/W, offset 0xD98, reset 0x0000.0000																
NUMBER																
<b>MPUBASE</b> , type R/W, offset 0xD9C, reset 0x0000.0000																
ADDR																
ADDR																
<b>MPUBASE1</b> , type R/W, offset 0xDA4, reset 0x0000.0000																
ADDR																
<b>MPUBASE2</b> , type R/W, offset 0xDAC, reset 0x0000.0000																
VALID																
<b>MPUBASE3</b> , type R/W, offset 0xDB4, reset 0x0000.0000																
ADDR																
<b>MPUATTR</b> , type R/W, offset 0xDA0, reset 0x0000.0000																
XN				AP				TEX		S	C	B				
SRD																
<b>MPUATTR1</b> , type R/W, offset 0xDA8, reset 0x0000.0000																
XN				AP				TEX		S	C	B				
SRD																
<b>MPUATTR2</b> , type R/W, offset 0xDB0, reset 0x0000.0000																
XN				AP				TEX		S	C	B				
SRD																
<b>MPUATTR3</b> , type R/W, offset 0=DB8, reset 0x0000.0000																
XN				AP				TEX		S	C	B				
SRD																

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															
DID0, type RO, offset 0x000, reset - (see page 167)															
VER															
MAJOR															
MINOR															
PBORCTL, type R/W, offset 0x030, reset 0x0000.7FFD (see page 169)															
BORTIM															
BORIOR BORWT															
LDOPCTL, type R/W, offset 0x034, reset 0x0000.0000 (see page 170)															
VADJ															
RIS, type RO, offset 0x050, reset 0x0000.0000 (see page 171)															
PLLFRIS CLRIS IOFRIS MOFRIS LDORIS BORRIS PLLFRIS															
IMC, type R/W, offset 0x054, reset 0x0000.0000 (see page 172)															
PLLLIM CLIM IOFIM MOFIM LDOIM BORIM PLLFIM															
MISC, type R/W1C, offset 0x058, reset 0x0000.0000 (see page 173)															
PLLLMIS CLMIS IOFMIS MOFMIS LDOMIS BORMIS															
RESC, type R/W, offset 0x05C, reset - (see page 174)															
LDO SW WDT BOR POR EXT															
RCC, type R/W, offset 0x060, reset 0x078E.3AC0 (see page 175)															
ACG SYSDIV USESYSDIV USEPWMDIV PWMDIV															
PWRDN OEN BYPASS PLLVER XTAL OSCSRC IOSCVER MOSCVER IOSCDIS MOSCDIS															
PLLCFG, type RO, offset 0x064, reset - (see page 179)															
OD F R															
DSLPCLKCFG, type R/W, offset 0x144, reset 0x0780.0000 (see page 180)															
IOSC															
CLKVCLR, type R/W, offset 0x150, reset 0x0000.0000 (see page 181)															
VERCLR															
LDOARST, type R/W, offset 0x160, reset 0x0000.0000 (see page 182)															
LDOARST															
DID1, type RO, offset 0x004, reset - (see page 183)															
VER FAM PARTNO															
TEMP PKG ROHS QUAL															
DC0, type RO, offset 0x008, reset 0x001F.001F (see page 185)															
SRAMSZ FLASHSZ															
DC1, type RO, offset 0x010, reset 0x0011.32BF (see page 186)															
ADC															
MINSYSDIV MAXADCSPD MPU TEMPSNS PLL WDT SWO SWD JTAG															
DC2, type RO, offset 0x014, reset 0x0107.1013 (see page 188)															
I2C0 COMPO SSI0 TIMER2 TIMER1 TIMERO															
DC3, type RO, offset 0x018, reset 0xBF0F.01FF (see page 190)															
32KHZ	CCP5	CCP4	CCP3	CCP2	CCP1	CCP0	C0O	C0PLUS	C0MINUS	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0

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>DC4, type RO, offset 0x01C, reset 0x0000.001F</b> (see page 192)															
<b>RCGC0, type R/W, offset 0x100, reset 0x00000040</b> (see page 193)															
<b>SCGC0, type R/W, offset 0x110, reset 0x00000040</b> (see page 195)															
<b>DCGC0, type R/W, offset 0x120, reset 0x00000040</b> (see page 197)															
<b>RCGC1, type R/W, offset 0x104, reset 0x00000000</b> (see page 199)															
<b>SCGC1, type R/W, offset 0x114, reset 0x00000000</b> (see page 201)															
<b>DCGC1, type R/W, offset 0x124, reset 0x00000000</b> (see page 203)															
<b>RCGC2, type R/W, offset 0x108, reset 0x00000000</b> (see page 205)															
<b>SCGC2, type R/W, offset 0x118, reset 0x00000000</b> (see page 206)															
<b>DCGC2, type R/W, offset 0x128, reset 0x00000000</b> (see page 208)															
<b>SRCR0, type R/W, offset 0x040, reset 0x00000000</b> (see page 210)															
<b>SRCR1, type R/W, offset 0x044, reset 0x00000000</b> (see page 211)															
<b>SRCR2, type R/W, offset 0x048, reset 0x00000000</b> (see page 213)															
<b>Internal Memory</b>															
<b>Flash Memory Control Registers (Flash Control Offset)</b>															
Base 0x400F.D000															
<b>FMA, type R/W, offset 0x000, reset 0x0000.0000</b>															
OFFSET															
<b>FMD, type R/W, offset 0x004, reset 0x0000.0000</b>															
DATA															
<b>FMC, type R/W, offset 0x008, reset 0x0000.0000</b>															
WRKEY															
COMT MERASE ERASE WRITE															

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>FCRIS, type RO, offset 0x00C, reset 0x0000.0000</b>															
<b>FCIM, type R/W, offset 0x010, reset 0x0000.0000</b>															
<b>FCMISC, type R/W1C, offset 0x014, reset 0x0000.0000</b>															
<b>Internal Memory</b>															
<b>Flash Memory Protection Registers (System Control Offset)</b>															
Base 0x400F.E000															
<b>USECRL, type R/W, offset 0x140, reset 0x31</b>															
<b>FMPRE, type R/W, offset 0x130, reset 0xBFFF.FFFF</b>															
<b>FMPPE, type R/W, offset 0x134, reset 0xFFFF.FFFF</b>															
<b>General-Purpose Input/Outputs (GPIOs)</b>															
GPIO Port A base: 0x4000.4000															
GPIO Port B base: 0x4000.5000															
GPIO Port C base: 0x4000.6000															
GPIO Port D base: 0x4000.7000															
GPIO Port E base: 0x4002.4000															
<b>GPIODATA, type R/W, offset 0x000, reset 0x0000.0000 (see page 244)</b>															
<b>GPIODIR, type R/W, offset 0x400, reset 0x0000.0000 (see page 245)</b>															
<b>GPIOIS, type R/W, offset 0x404, reset 0x0000.0000 (see page 246)</b>															
<b>GPIOIBE, type R/W, offset 0x408, reset 0x0000.0000 (see page 247)</b>															
<b>GPIOIEV, type R/W, offset 0x40C, reset 0x0000.0000 (see page 248)</b>															
<b>GPIOIM, type R/W, offset 0x410, reset 0x0000.0000 (see page 249)</b>															
<b>GPIORIS, type RO, offset 0x414, reset 0x0000.0000 (see page 250)</b>															
<b>GPIOOMIS, type RO, offset 0x418, reset 0x0000.0000 (see page 251)</b>															
<b>GPIOICR, type W1C, offset 0x41C, reset 0x0000.0000 (see page 252)</b>															

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>GPIOAFSEL</b> , type R/W, offset 0x420, reset - (see page 253)															
AFSEL															
<b>GPIODR2R</b> , type R/W, offset 0x500, reset 0x0000.00FF (see page 255)															
DRV2															
<b>GPIODR4R</b> , type R/W, offset 0x504, reset 0x0000.0000 (see page 256)															
DRV4															
<b>GPIODR8R</b> , type R/W, offset 0x508, reset 0x0000.0000 (see page 257)															
DRV8															
<b>GPIOODR</b> , type R/W, offset 0x50C, reset 0x0000.0000 (see page 258)															
ODE															
<b>GPIOPUR</b> , type R/W, offset 0x510, reset 0x0000.00FF (see page 259)															
PUE															
<b>GPIOPDR</b> , type R/W, offset 0x514, reset 0x0000.0000 (see page 260)															
PDE															
<b>GPIOSLR</b> , type R/W, offset 0x518, reset 0x0000.0000 (see page 261)															
SRL															
<b>GPIODEN</b> , type R/W, offset 0x51C, reset 0x0000.00FF (see page 262)															
DEN															
<b>GPIOPeriphID4</b> , type RO, offset 0xFD0, reset 0x0000.0000 (see page 263)															
PID4															
<b>GPIOPeriphID5</b> , type RO, offset 0xFD4, reset 0x0000.0000 (see page 264)															
PID5															
<b>GPIOPeriphID6</b> , type RO, offset 0xFD8, reset 0x0000.0000 (see page 265)															
PID6															
<b>GPIOPeriphID7</b> , type RO, offset 0xFDC, reset 0x0000.0000 (see page 266)															
PID7															
<b>GPIOPeriphID0</b> , type RO, offset 0xFE0, reset 0x0000.0061 (see page 267)															
PID0															
<b>GPIOPeriphID1</b> , type RO, offset 0xFE4, reset 0x0000.0000 (see page 268)															
PID1															
<b>GPIOPeriphID2</b> , type RO, offset 0xFE8, reset 0x0000.0018 (see page 269)															
PID2															
<b>GPIOPeriphID3</b> , type RO, offset 0xFEC, reset 0x0000.0001 (see page 270)															
PID3															

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>GPIOPCellID0</b> , type RO, offset 0xFF0, reset 0x0000.000D (see page 271)															
CID0															
<b>GPIOPCellID1</b> , type RO, offset 0xFF4, reset 0x0000.00F0 (see page 272)															
CID1															
<b>GPIOPCellID2</b> , type RO, offset 0xFF8, reset 0x0000.0005 (see page 273)															
CID2															
<b>GPIOPCellID3</b> , type RO, offset 0xFFC, reset 0x0000.00B1 (see page 274)															
CID3															
<b>General-Purpose Timers</b>															
Timer0 base: 0x4003.0000															
Timer1 base: 0x4003.1000															
Timer2 base: 0x4003.2000															
<b>GPTMCFG</b> , type R/W, offset 0x000, reset 0x0000.0000 (see page 288)															
GPTMCFG															
<b>GPTMTAMR</b> , type R/W, offset 0x004, reset 0x0000.0000 (see page 289)															
TAAMS    TACMR    TAMR															
<b>GPTMTBMR</b> , type R/W, offset 0x008, reset 0x0000.0000 (see page 291)															
TBAMS    TBCMR    TBMR															
<b>GPTMCTL</b> , type R/W, offset 0x00C, reset 0x0000.0000 (see page 293)															
TBPWML    TBOTE    TBEVENT    TBSTALL    TBEN    TAPWML    TAOTE    RTCEN    TAEVENT    TASTALL    TAEN															
<b>GPTMIMR</b> , type R/W, offset 0x018, reset 0x0000.0000 (see page 296)															
CBEIM    CBMIM    TBTOIM    RTCIM    CAEIM    CAMIM    TATOIM															
<b>GPTMRIS</b> , type RO, offset 0x01C, reset 0x0000.0000 (see page 298)															
CBERIS    CBMRIS    TBTORIS    RTCRIS    CAERIS    CAMRIS    TATORIS															
<b>GPTMMIS</b> , type RO, offset 0x020, reset 0x0000.0000 (see page 299)															
CBEMIS    CBMMIS    TBTONIS    RTCMIS    CAEMIS    CAMMIS    TATOMIS															
<b>GPTMICR</b> , type W1C, offset 0x024, reset 0x0000.0000 (see page 300)															
CBECINT    CBMCINT    TBTOCINT    RTCCINT    CAECINT    CAMCINT    TATOCINT															
<b>GPTMTAILR</b> , type R/W, offset 0x028, reset 0xFFFF.FFFF (see page 302)															
TAILRH TAILRL															
<b>GPTMTBILR</b> , type R/W, offset 0x02C, reset 0x0000.FFFF (see page 303)															
TBILRL															
<b>GPTMTAMATCHR</b> , type R/W, offset 0x030, reset 0xFFFF.FFFF (see page 304)															
TAMRH TAMRL															
<b>GPTMTBMATCHR</b> , type R/W, offset 0x034, reset 0x0000.FFFF (see page 305)															
TBMRL															

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>GPTMTAPR</b> , type R/W, offset 0x038, reset 0x0000.0000 (see page 306)															
TAPSR															
<b>GPTMTBPR</b> , type R/W, offset 0x03C, reset 0x0000.0000 (see page 307)															
TBPSR															
<b>GPTMTAPMR</b> , type R/W, offset 0x040, reset 0x0000.0000 (see page 308)															
TAPSMR															
<b>GPTMTBPMR</b> , type R/W, offset 0x044, reset 0x0000.0000 (see page 309)															
TBPSMR															
<b>GPTMTAR</b> , type RO, offset 0x048, reset 0xFFFF.FFFF (see page 310)															
TARH															
<b>GPTMTBR</b> , type RO, offset 0x04C, reset 0x0000.FFFF (see page 311)															
TBRL															
<b>Watchdog Timer</b>															
Base 0x4000.0000															
<b>WDTLOAD</b> , type R/W, offset 0x000, reset 0xFFFF.FFFF (see page 316)															
WDTLoad															
<b>WDTVVALUE</b> , type RO, offset 0x004, reset 0xFFFF.FFFF (see page 317)															
WDTValue															
<b>WDTCTL</b> , type R/W, offset 0x008, reset 0x0000.0000 (see page 318)															
RESEN    INTEN															
<b>WDTICR</b> , type WO, offset 0x00C, reset - (see page 319)															
WDTIntClr															
<b>WDTRIS</b> , type RO, offset 0x010, reset 0x0000.0000 (see page 320)															
WDTRIS															
<b>WDTMIS</b> , type RO, offset 0x014, reset 0x0000.0000 (see page 321)															
WDTMIS															
<b>WDTTEST</b> , type R/W, offset 0x418, reset 0x0000.0000 (see page 322)															
STALL															
<b>WDTLOCK</b> , type R/W, offset 0xC00, reset 0x0000.0000 (see page 323)															
WDTLock															
<b>WDTPeriphID4</b> , type RO, offset 0xFD0, reset 0x0000.0000 (see page 324)															
PID4															
<b>WDTPeriphID5</b> , type RO, offset 0xFD4, reset 0x0000.0000 (see page 325)															
PID5															
<b>WDTPeriphID6</b> , type RO, offset 0xFD8, reset 0x0000.0000 (see page 326)															
PID6															

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>WDTPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000 (see page 327)</b>															
PID7															
<b>WDTPeriphID0, type RO, offset 0xFE0, reset 0x0000.0005 (see page 328)</b>															
PID0															
<b>WDTPeriphID1, type RO, offset 0xFE4, reset 0x0000.0018 (see page 329)</b>															
PID1															
<b>WDTPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018 (see page 330)</b>															
PID2															
<b>WDTPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001 (see page 331)</b>															
PID3															
<b>WDTPCellID0, type RO, offset 0xFF0, reset 0x0000.000D (see page 332)</b>															
CID0															
<b>WDTPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0 (see page 333)</b>															
CID1															
<b>WDTPCellID2, type RO, offset 0xFF8, reset 0x0000.0005 (see page 334)</b>															
CID2															
<b>WDTPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1 (see page 335)</b>															
CID3															
<b>Analog-to-Digital Converter (ADC)</b>															
Base 0x4003.8000															
<b>ADCACTSS, type R/W, offset 0x000, reset 0x0000.0000 (see page 346)</b>															
ASEN3 ASEN2 ASEN1 ASENO															
<b>ADCRIS, type RO, offset 0x004, reset 0x0000.0000 (see page 347)</b>															
INR3 INR2 INR1 INR0															
<b>ADCIM, type R/W, offset 0x008, reset 0x0000.0000 (see page 348)</b>															
MASK3 MASK2 MASK1 MASK0															
<b>ADCISC, type R/W1C, offset 0x00C, reset 0x0000.0000 (see page 349)</b>															
IN3 IN2 IN1 IN0															
<b>ADCOSTAT, type R/W1C, offset 0x010, reset 0x0000.0000 (see page 350)</b>															
OV3 OV2 OV1 OV0															
<b>ADCEMUX, type R/W, offset 0x014, reset 0x0000.0000 (see page 351)</b>															
EM3 EM2 EM1 EM0															
<b>ADCUSTAT, type R/W1C, offset 0x018, reset 0x0000.0000 (see page 355)</b>															
UV3 UV2 UV1 UV0															
<b>ADCSSPRI, type R/W, offset 0x020, reset 0x0000.3210 (see page 356)</b>															
SS3 SS2 SS1 SS0															

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>ADCPSSI</b> , type WO, offset 0x028, reset - (see page 358)																	
														SS3	SS2	SS1	SS0
<b>ADCSAC</b> , type R/W, offset 0x030, reset 0x0000.0000 (see page 359)																AVG	
<b>ADCSSMUX0</b> , type R/W, offset 0x040, reset 0x0000.0000 (see page 360)																MUX4	
				MUX7			MUX6			MUX5						MUX0	
				MUX3			MUX2			MUX1							
<b>ADCSSCTL0</b> , type R/W, offset 0x044, reset 0x0000.0000 (see page 362)																D4	
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</b> , type RO, offset 0x048, reset - (see page 365)																DATA	
<b>ADCSSFIFO1</b> , type RO, offset 0x068, reset - (see page 365)																DATA	
<b>ADCSSFIFO2</b> , type RO, offset 0x088, reset - (see page 365)																DATA	
<b>ADCSSFIFO3</b> , type RO, offset 0xA8, reset - (see page 365)																DATA	
<b>ADCSSFSTAT0</b> , type RO, offset 0x04C, reset 0x0000.0100 (see page 366)																TPTR	
				FULL			EMPTY			HPTR							
<b>ADCSSFSTAT1</b> , type RO, offset 0x06C, reset 0x0000.0100 (see page 366)																TPTR	
				FULL			EMPTY			HPTR							
<b>ADCSSFSTAT2</b> , type RO, offset 0x08C, reset 0x0000.0100 (see page 366)																TPTR	
				FULL			EMPTY			HPTR							
<b>ADCSSFSTAT3</b> , type RO, offset 0x0AC, reset 0x0000.0100 (see page 366)																TPTR	
				FULL			EMPTY			HPTR							
<b>ADCSSMUX1</b> , type R/W, offset 0x060, reset 0x0000.0000 (see page 367)																MUX0	
				MUX3			MUX2			MUX1							
<b>ADCSSMUX2</b> , type R/W, offset 0x080, reset 0x0000.0000 (see page 367)																MUX0	
				MUX3			MUX2			MUX1							
<b>ADCSSCTL1</b> , type R/W, offset 0x064, reset 0x0000.0000 (see page 368)																	
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0		
<b>ADCSSCTL2</b> , type R/W, offset 0x084, reset 0x0000.0000 (see page 368)																	
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0		
<b>ADCSSMUX3</b> , type R/W, offset 0xA0, reset 0x0000.0000 (see page 370)																MUX0	

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>ADCSSCTL3, type R/W, offset 0x0A4, reset 0x0000.0002 (see page 371)</b>															
														TS0	IE0
														END0	D0
<b>ADCTMLB, type R/W, offset 0x100, reset 0x0000.0000 (see page 372)</b>															
															LB
<b>Universal Asynchronous Receivers/Transmitters (UARTs)</b>															
UART0 base: 0x4000.C000															
UART1 base: 0x4000.D000															
<b>UARTDR, type R/W, offset 0x000, reset 0x0000.0000 (see page 381)</b>															
				OE	BE	PE	FE								DATA
<b>UARTRSR/UARTECR, type RO, offset 0x004, reset 0x0000.0000 (Reads) (see page 383)</b>															
														OE	BE
														PE	FE
<b>UARTRSR/UARTECR, type WO, offset 0x004, reset 0x0000.0000 (Writes) (see page 383)</b>															
															DATA
<b>UARTFR, type RO, offset 0x018, reset 0x0000.0090 (see page 385)</b>															
														TXFE	RXFF
														TXFF	RXFE
														BUSY	
<b>UARTIBRD, type R/W, offset 0x024, reset 0x0000.0000 (see page 387)</b>															
															DIVINT
<b>UARTFBRD, type R/W, offset 0x028, reset 0x0000.0000 (see page 388)</b>															
															DIVFRAC
<b>UARTLCRH, type R/W, offset 0x02C, reset 0x0000.0000 (see page 389)</b>															
														SPS	WLEN
														FEN	STP2
														EPS	PEN
														BRK	
<b>UARTCTL, type R/W, offset 0x030, reset 0x0000.0300 (see page 391)</b>															
														RXE	TXE
														LBE	
															UARTEN
<b>UARTIFLS, type R/W, offset 0x034, reset 0x0000.0012 (see page 393)</b>															
															RXIFLSEL
<b>UARTIM, type R/W, offset 0x038, reset 0x0000.0000 (see page 395)</b>															
														OEIM	BEIM
														PEIM	FEIM
														RTIM	TXIM
														RXIM	
<b>UARTRIS, type RO, offset 0x03C, reset 0x0000.0000 (see page 397)</b>															
														OERIS	BERIS
														PERIS	FERIS
														RTRIS	TXRIS
														RXRIS	
<b>UARTMIS, type RO, offset 0x040, reset 0x0000.0000 (see page 398)</b>															
														OEMIS	BEMIS
														PEMIS	FEMIS
														RTMIS	TXMIS
														RXMIS	
<b>UARTICR, type W1C, offset 0x044, reset 0x0000.0000 (see page 399)</b>															
														OEIC	BEIC
														PEIC	FEIC
														RTIC	TXIC
														RXIC	
<b>UARTPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000 (see page 401)</b>															
															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>UARTPeriphID5</b> , type RO, offset 0xFD4, reset 0x0000.0000 (see page 402)															
															PID5
<b>UARTPeriphID6</b> , type RO, offset 0xFD8, reset 0x0000.0000 (see page 403)															
															PID6
<b>UARTPeriphID7</b> , type RO, offset 0xFDC, reset 0x0000.0000 (see page 404)															
															PID7
<b>UARTPeriphID0</b> , type RO, offset 0xFE0, reset 0x0000.0011 (see page 405)															
															PID0
<b>UARTPeriphID1</b> , type RO, offset 0xFE4, reset 0x0000.0000 (see page 406)															
															PID1
<b>UARTPeriphID2</b> , type RO, offset 0xFE8, reset 0x0000.0018 (see page 407)															
															PID2
<b>UARTPeriphID3</b> , type RO, offset 0xFEC, reset 0x0000.0001 (see page 408)															
															PID3
<b>UARTPCellID0</b> , type RO, offset 0xFF0, reset 0x0000.000D (see page 409)															
															CID0
<b>UARTPCellID1</b> , type RO, offset 0xFF4, reset 0x0000.00F0 (see page 410)															
															CID1
<b>UARTPCellID2</b> , type RO, offset 0xFF8, reset 0x0000.0005 (see page 411)															
															CID2
<b>UARTPCellID3</b> , type RO, offset 0xFFC, reset 0x0000.00B1 (see page 412)															
															CID3
<b>Synchronous Serial Interface (SSI)</b>															
SSI0 base: 0x4000.8000															
<b>SSICR0</b> , type R/W, offset 0x000, reset 0x0000.0000 (see page 426)															
															SCR      SPH      SPO      FRF      DSS
<b>SSICR1</b> , type R/W, offset 0x004, reset 0x0000.0000 (see page 428)															
															SOD      MS      SSE      LBM
<b>SSIDR</b> , type R/W, offset 0x008, reset 0x0000.0000 (see page 430)															
															DATA
<b>SSISR</b> , type RO, offset 0x00C, reset 0x0000.0003 (see page 431)															
															BSY      RFF      RNE      TNF      TFE
<b>SSICPSR</b> , type R/W, offset 0x010, reset 0x0000.0000 (see page 433)															
															CPSDVS
<b>SSIIM</b> , type R/W, offset 0x014, reset 0x0000.0000 (see page 434)															
															TXIM      RXIM      RTIM      RORIM



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>I2CMCS, type RO, offset 0x004, reset 0x0000.0000 (Reads)</b>															
									BUSBSY	IDLE	ARBLST	DATACK	ADRACK	ERROR	BUSY
<b>I2CMCS, type WO, offset 0x004, reset 0x0000.0000 (Writes)</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 0 base: 0x4002.0000															
<b>I2CSOAR, type R/W, offset 0x800, reset 0x0000.0000</b>															
												OAR			
<b>I2CSCSR, type RO, offset 0x804, reset 0x0000.0000 (Reads)</b>															
												FBR	TREQ	RREQ	
<b>I2CSCSR, type WO, offset 0x804, reset 0x0000.0000 (Writes)</b>															
												DA			
<b>I2CSDR, type R/W, offset 0x808, reset 0x0000.0000</b>															
												DATA			
<b>I2CSIMR, type R/W, offset 0x80C, reset 0x0000.0000</b>															
												DATAIM			
<b>I2CSRIS, type RO, offset 0x810, reset 0x0000.0000</b>															
												DATARIS			
<b>I2CSMIS, type RO, offset 0x814, reset 0x0000.0000</b>															
												DATAMIS			

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>I2CSICR</b> , type WO, offset 0x818, reset 0x0000.0000															
DATAIC															
<b>Analog Comparator</b>															
Base 0x4003.C000															
<b>ACMIS</b> , type R/W1C, offset 0x000, reset 0x0000.0000 (see page 493)															
IN0															
<b>ACRIS</b> , type RO, offset 0x004, reset 0x0000.0000 (see page 494)															
IN0															
<b>ACINTEN</b> , type R/W, offset 0x008, reset 0x0000.0000 (see page 495)															
IN0															
<b>ACREFCTL</b> , type R/W, offset 0x010, reset 0x0000.0000 (see page 496)															
EN      RNG      VREF															
<b>ACSTAT0</b> , type RO, offset 0x020, reset 0x0000.0000 (see page 497)															
OVAL															
<b>ACCTL0</b> , type R/W, offset 0x024, reset 0x0000.0000 (see page 498)															
TOEN      ASRCP      TSLVAL      TSEN      ISLVAL      ISEN      CINV															
<b>Pulse Width Modulator (PWM)</b>															
Base 0x4002.8000															
<b>PWMCTL</b> , type R/W, offset 0x000, reset 0x0000.0000 (see page 510)															
GlobalSync2      GlobalSync1      GlobalSync0															
<b>PWMSYNC</b> , type R/W, offset 0x004, reset 0x0000.0000 (see page 511)															
Sync2      Sync1      Sync0															
<b>PWMENABLE</b> , type R/W, offset 0x008, reset 0x0000.0000 (see page 512)															
PWM5En      PWM4En      PWM3En      PWM2En      PWM1En      PWM0En															
<b>PWMINVERT</b> , type R/W, offset 0x00C, reset 0x0000.0000 (see page 513)															
PWM5Inv      PWM4Inv      PWM3Inv      PWM2Inv      PWM1Inv      PWM0Inv															
<b>PWMFAULT</b> , type R/W, offset 0x010, reset 0x0000.0000 (see page 514)															
Fault5      Fault4      Fault3      Fault2      Fault1      Fault0															
<b>PWMINTEN</b> , type R/W, offset 0x014, reset 0x0000.0000 (see page 515)															
IntFault      IntPWM2      IntPWM1      IntPWM0															
<b>PWMRIS</b> , type RO, offset 0x018, reset 0x0000.0000 (see page 516)															
IntFault      IntPWM2      IntPWM1      IntPWM0															
<b>PWMISC</b> , type R/W1C, offset 0x01C, reset 0x0000.0000 (see page 517)															
IntFault      IntPWM2      IntPWM1      IntPWM0															

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>PWMSTATUS</b> , type RO, offset 0x020, reset 0x0000.0000 (see page 518)															

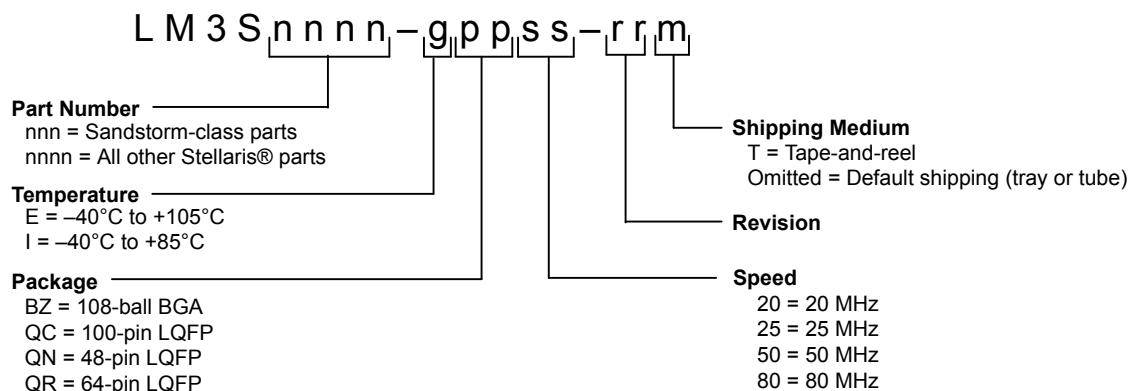
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>PWM1COUNT, type RO, offset 0x094, reset 0x0000.0000 (see page 527)</b>															
Count															
<b>PWM2COUNT, type RO, offset 0x0D4, reset 0x0000.0000 (see page 527)</b>															
Count															
<b>PWM0CMPA, type R/W, offset 0x058, reset 0x0000.0000 (see page 528)</b>															
CompA															
<b>PWM1CMPA, type R/W, offset 0x098, reset 0x0000.0000 (see page 528)</b>															
CompA															
<b>PWM2CMPA, type R/W, offset 0x0D8, reset 0x0000.0000 (see page 528)</b>															
CompA															
<b>PWM0CMPB, type R/W, offset 0x05C, reset 0x0000.0000 (see page 529)</b>															
CompB															
<b>PWM1CMPB, type R/W, offset 0x09C, reset 0x0000.0000 (see page 529)</b>															
CompB															
<b>PWM2CMPB, type R/W, offset 0x0DC, reset 0x0000.0000 (see page 529)</b>															
CompB															
<b>PWM0GENA, type R/W, offset 0x060, reset 0x0000.0000 (see page 530)</b>															
ActCmpBD      ActCmpBU      ActCmpAD      ActCmpAU      ActLoad      ActZero															
<b>PWM1GENA, type R/W, offset 0x0A0, reset 0x0000.0000 (see page 530)</b>															
ActCmpBD      ActCmpBU      ActCmpAD      ActCmpAU      ActLoad      ActZero															
<b>PWM2GENA, type R/W, offset 0xE0, reset 0x0000.0000 (see page 530)</b>															
ActCmpBD      ActCmpBU      ActCmpAD      ActCmpAU      ActLoad      ActZero															
<b>PWM0GENB, type R/W, offset 0x064, reset 0x0000.0000 (see page 533)</b>															
ActCmpBD      ActCmpBU      ActCmpAD      ActCmpAU      ActLoad      ActZero															
<b>PWM1GENB, type R/W, offset 0x0A4, reset 0x0000.0000 (see page 533)</b>															
ActCmpBD      ActCmpBU      ActCmpAD      ActCmpAU      ActLoad      ActZero															
<b>PWM2GENB, type R/W, offset 0xE4, reset 0x0000.0000 (see page 533)</b>															
ActCmpBD      ActCmpBU      ActCmpAD      ActCmpAU      ActLoad      ActZero															
<b>PWM0DBCTL, type R/W, offset 0x068, reset 0x0000.0000 (see page 536)</b>															
Enable															
<b>PWM1DBCTL, type R/W, offset 0xA8, reset 0x0000.0000 (see page 536)</b>															
Enable															
<b>PWM2DBCTL, type R/W, offset 0xE8, reset 0x0000.0000 (see page 536)</b>															
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>PWM0DBRISE</b> , type R/W, offset 0x06C, reset 0x0000.0000 (see page 537)															
RiseDelay															
<b>PWM1DBRISE</b> , type R/W, offset 0x0AC, reset 0x0000.0000 (see page 537)															
RiseDelay															
<b>PWM2DBRISE</b> , type R/W, offset 0x0EC, reset 0x0000.0000 (see page 537)															
RiseDelay															
<b>PWM0DBFALL</b> , type R/W, offset 0x070, reset 0x0000.0000 (see page 538)															
FallDelay															
<b>PWM1DBFALL</b> , type R/W, offset 0x0B0, reset 0x0000.0000 (see page 538)															
FallDelay															
<b>PWM2DBFALL</b> , type R/W, offset 0x0F0, reset 0x0000.0000 (see page 538)															
FallDelay															

## C Ordering and Contact Information

### C.1 Ordering Information

The figure below defines the full set of potential orderable part numbers for all the Stellaris® LM3S microcontrollers. See the Package Option Addendum for the valid orderable part numbers for the LM3S811 microcontroller.



### C.2 Part Markings

The Stellaris microcontrollers are marked with an identifying number. This code contains the following information:

- The first line indicates the part number, for example, LM3S9B90.
- In the second line, the first eight characters indicate the temperature, package, speed, revision, and product status. For example in the figure below, IQC80C0X indicates an Industrial temperature (I), 100-pin LQFP package (QC), 80-MHz (80), revision C0 (C0) device. The letter immediately following the revision indicates product status. An X indicates experimental and requires a waiver; an S indicates the part is fully qualified and released to production.
- The remaining characters contain internal tracking numbers.



### C.3 Kits

The 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
- Evaluation Kits provide a low-cost and effective means of evaluating Stellaris microcontrollers before purchase
- Development Kits provide you with all the tools you need to develop and prototype embedded applications right out of the box

See the website at [www.ti.com/stellaris](http://www.ti.com/stellaris) for the latest tools available, or ask your distributor.

## C.4 Support Information

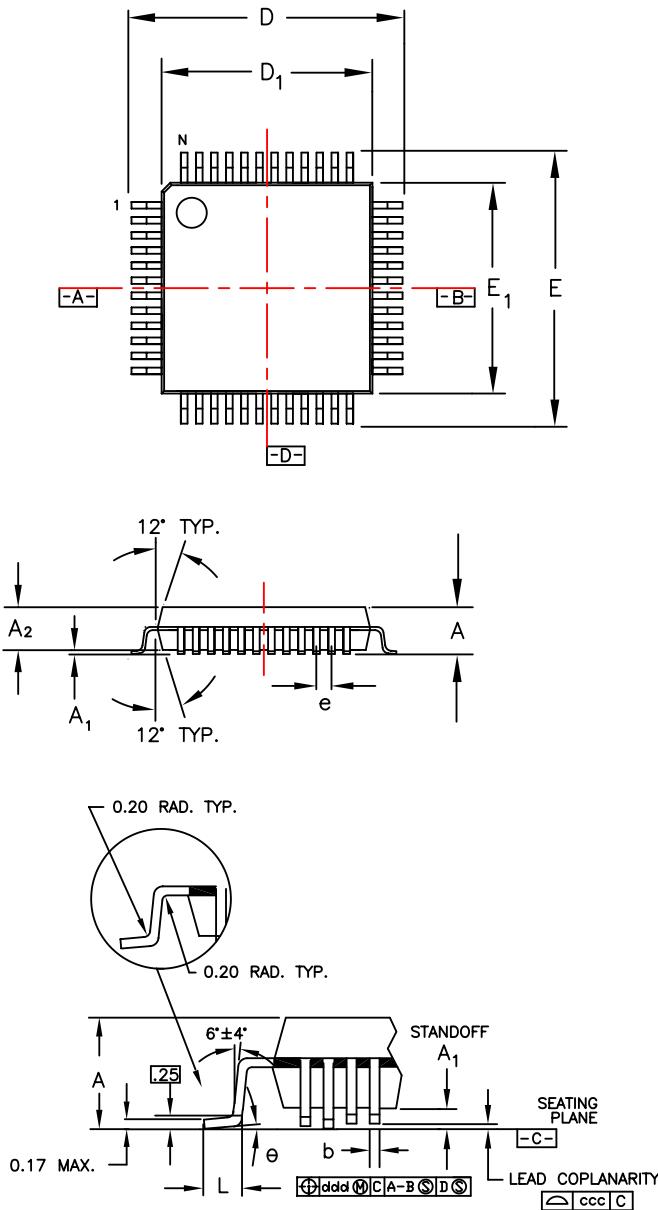
For support on Stellaris products, contact the TI Worldwide Product Information Center nearest you:  
<http://www-k.ext.ti.com/sc/technical-support/product-information-centers.htm>.

## D Package Information

### D.1 48-Pin LQFP Package

#### D.1.1 Package Dimensions

Figure D-1. Stellaris LM3S811 48-Pin LQFP Package



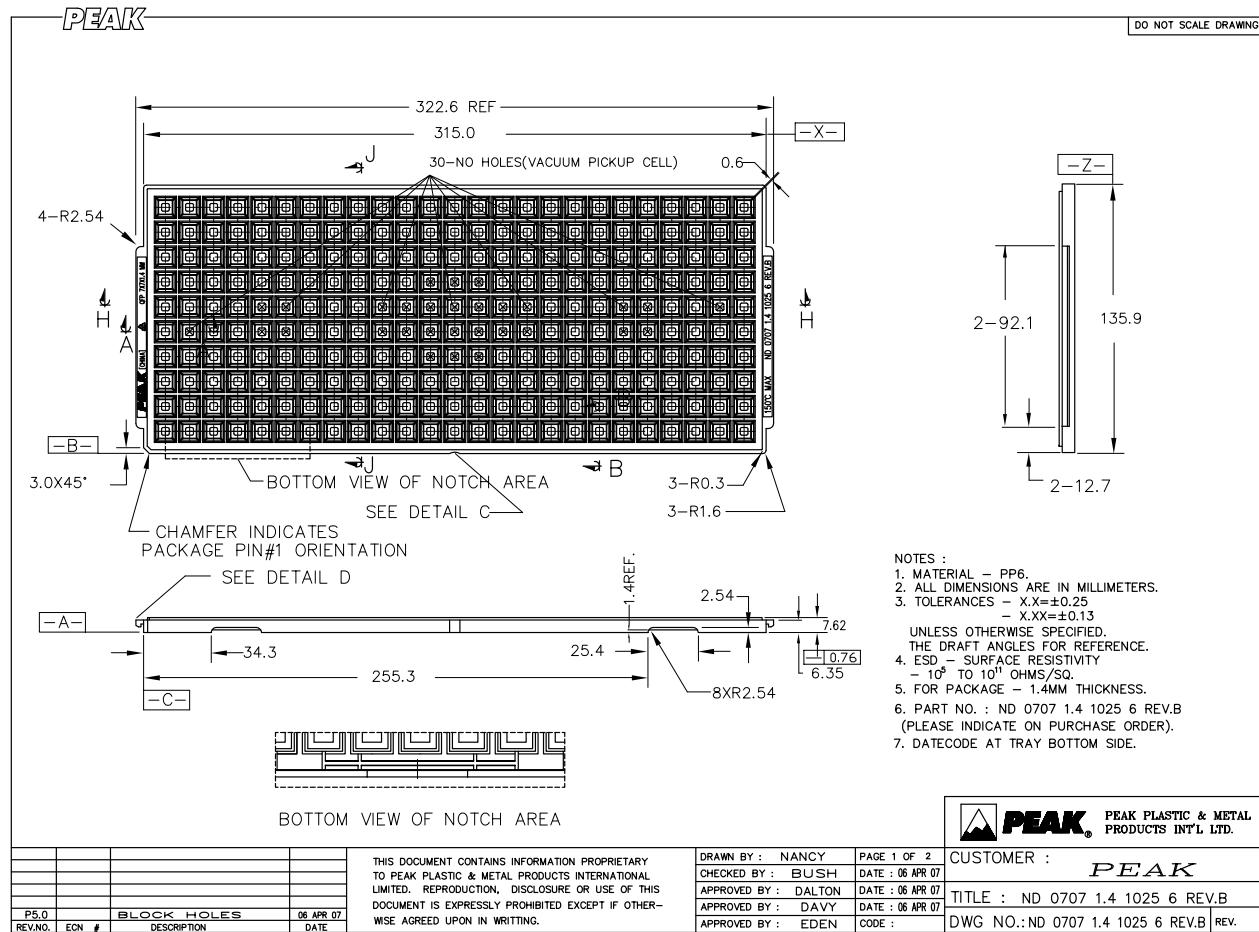
**Note:** The following notes apply to the package drawing.

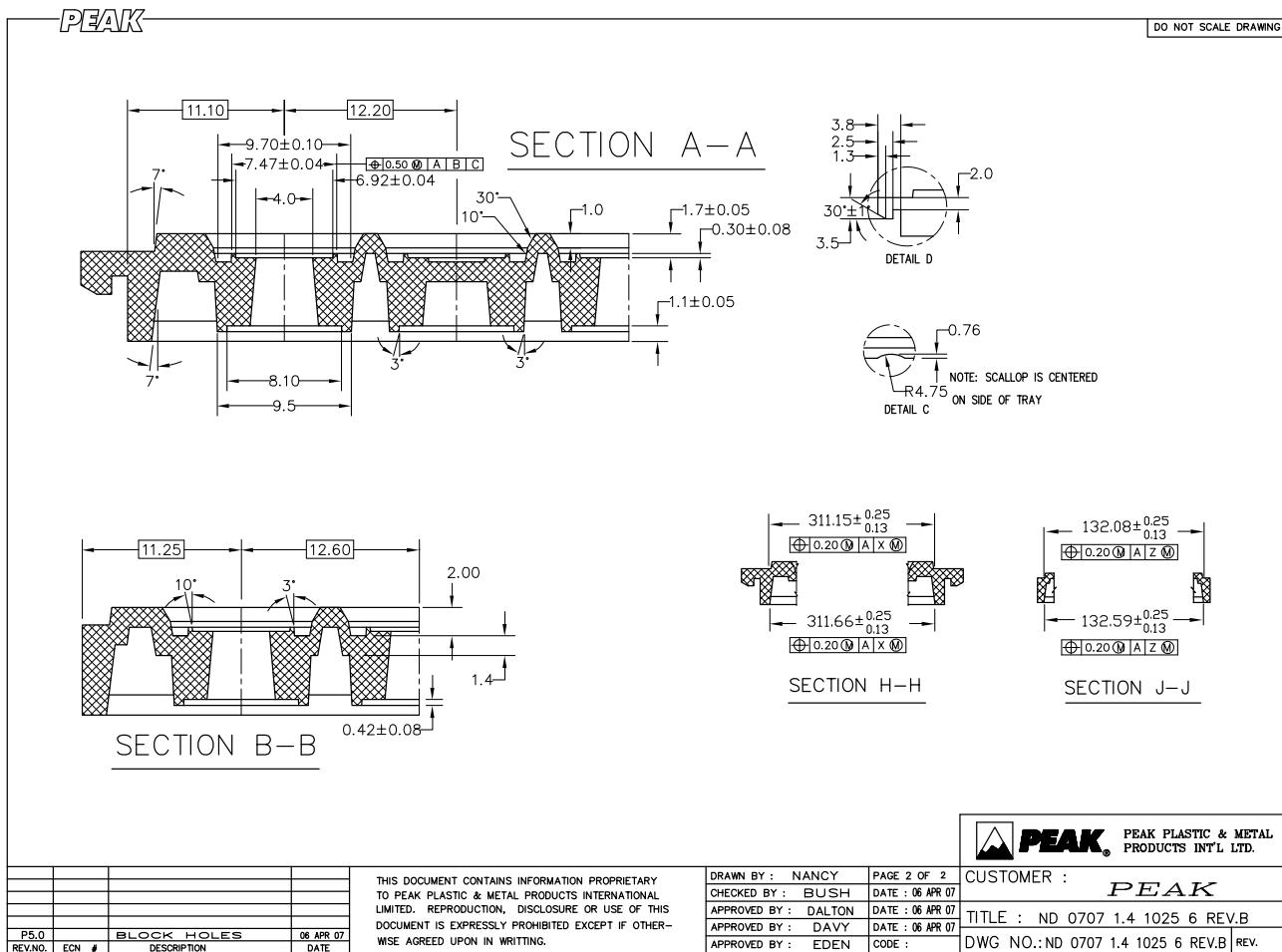
1. All dimensions are in mm.
2. Dimensions shown are nominal with tolerances indicated.
3. Foot length "L" is measured at gage plane 0.25 mm above seating plane.
4. L/F: Eftec 64T Cu or equivalent, 0.127 mm (0.005") thick.

Symbol	Package Type		Note	
	48LD LQFP			
	MIN	MAX		
A	-	1.60		
A <sub>1</sub>	0.05	0.15		
A <sub>2</sub>	-	1.40		
D		9.00		
D <sub>1</sub>		7.00		
E		9.00		
E <sub>1</sub>		7.00		
L		0.60		
e		0.50		
b		0.22		
theta		0° - 7°		
ddd		0.08		
ccc		0.08		
JEDEC Reference Drawing			MS-026	
Variation Designator			BBC	

## D.1.2 Tray Dimensions

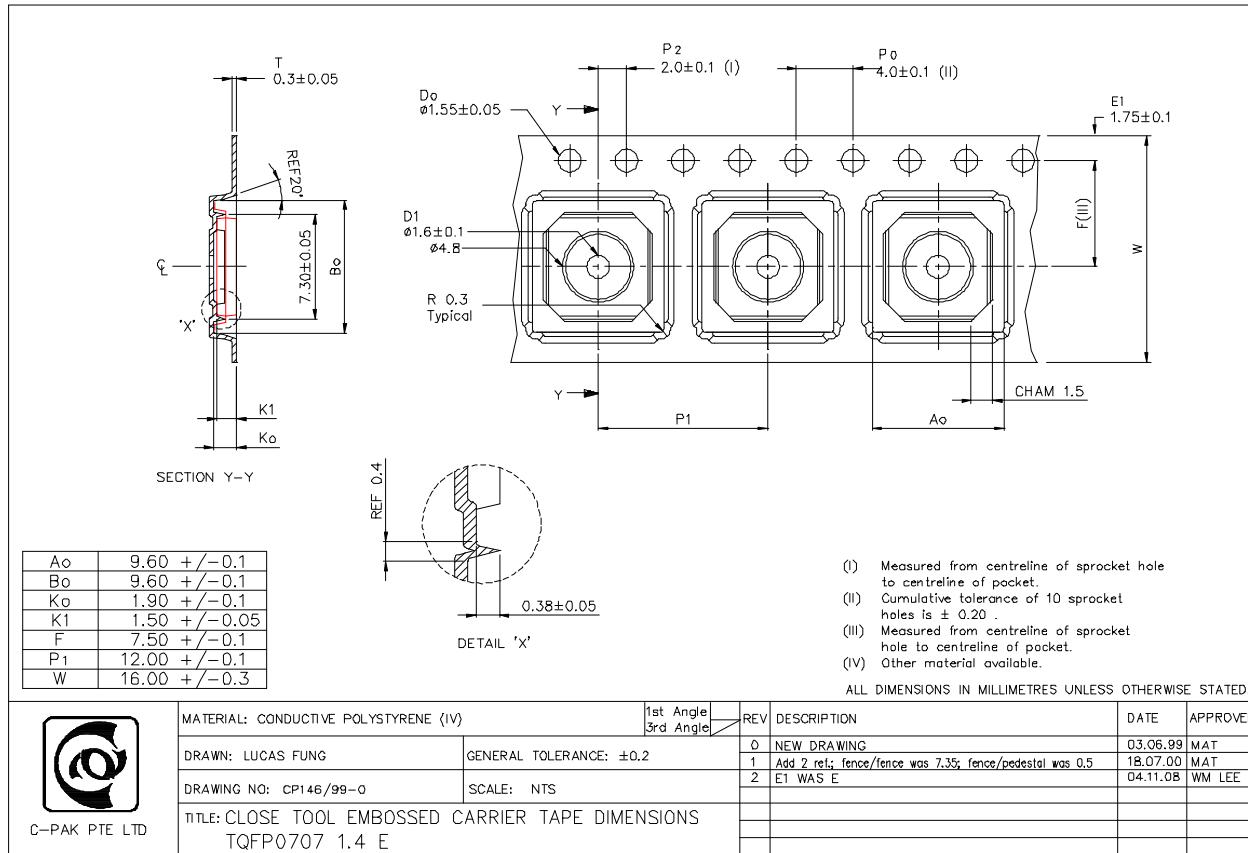
Figure D-2. 48-Pin LQFP Tray Dimensions





### D.1.3 Tape and Reel Dimensions

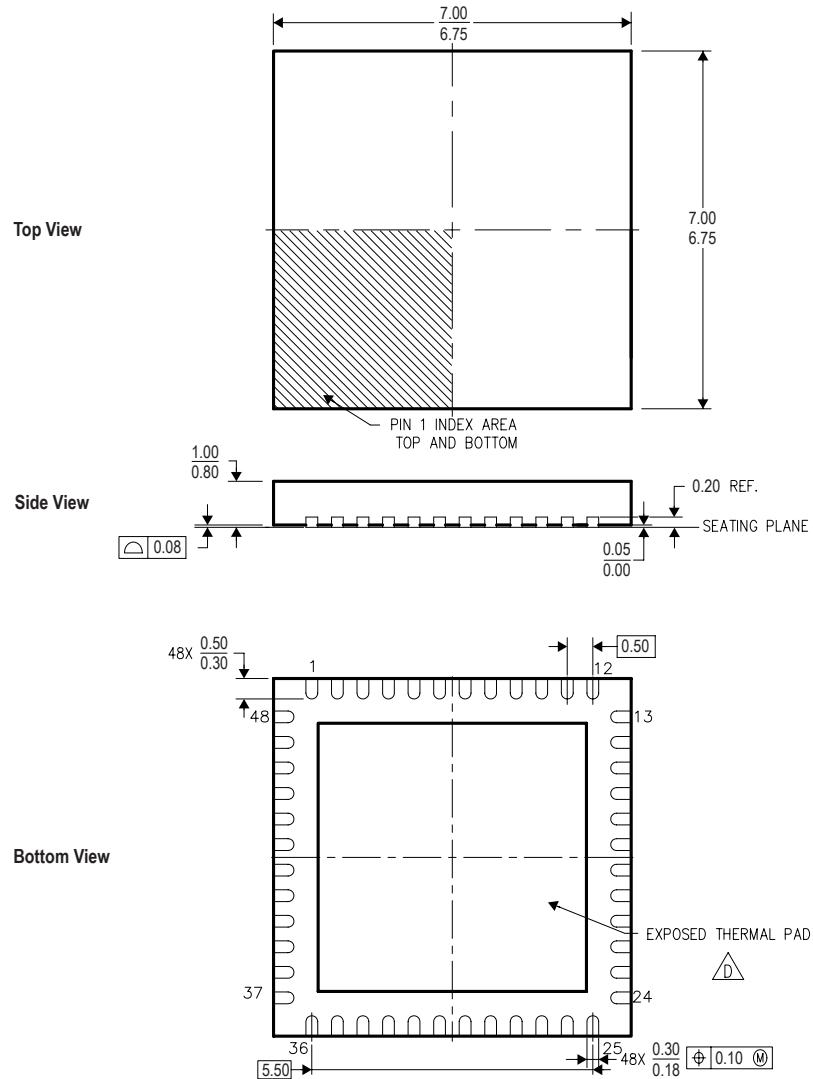
Figure D-3. 48-Pin LQFP Tape and Reel Dimensions



## D.2 48-Pin QFN Package

### D.2.1 Package Dimensions

Figure D-4. Stellaris LM3S811 48-Pin QFN Package<sup>1</sup>



NOTES: A. All linear dimensions are in millimeters. Dimensioning and tolerancing per ASME Y14.5M-1994.

B. This drawing is subject to change without notice.

C. Quad Flatpack, No-leads (QFN) package configuration.

D. The package thermal pad must be soldered to the board for thermal and mechanical performance. In addition, the pad must be connected to GND.  
E. Falls within JEDEC MO-220.

<sup>1</sup>OBSOLETE: TI has discontinued production of this device.

**PACKAGING INFORMATION**

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish (6)	MSL Peak Temp (3)	Op Temp (°C)	Device Marking (4/5)	Samples
LM3S811-IQN50-C2	NRND	LQFP	PT	48	250	Green (RoHS & no Sb/Br)	SN   Call TI	Level-3-260C-168 HR	-40 to 85	LM3S811 IQN50 PT	
LM3S811-IQN50-C2T	NRND	LQFP	PT	48	2000	Green (RoHS & no Sb/Br)	SN   Call TI	Level-3-260C-168 HR	-40 to 85	LM3S811 IQN50 PT	

(1) The marketing status values are defined as follows:

**ACTIVE:** Product device recommended for new designs.

**LIFEBUY:** TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.

**NRND:** Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.

**PREVIEW:** Device has been announced but is not in production. Samples may or may not be available.

**OBSOLETE:** TI has discontinued the production of the device.

(2) Eco Plan - The planned eco-friendly classification: Pb-Free (RoHS), Pb-Free (RoHS Exempt), or Green (RoHS & no Sb/Br) - please check <http://www.ti.com/productcontent> for the latest availability information and additional product content details.

**TBD:** The Pb-Free/Green conversion plan has not been defined.

**Pb-Free (RoHS):** TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.

**Pb-Free (RoHS Exempt):** This component has a RoHS exemption for either 1) lead-based flip-chip solder bumps used between the die and package, or 2) lead-based die adhesive used between the die and leadframe. The component is otherwise considered Pb-Free (RoHS compatible) as defined above.

**Green (RoHS & no Sb/Br):** TI defines "Green" to mean Pb-Free (RoHS compatible), and free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material)

(3) MSL, Peak Temp. - The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

(4) There may be additional marking, which relates to the logo, the lot trace code information, or the environmental category on the device.

(5) Multiple Device Markings will be inside parentheses. Only one Device Marking contained in parentheses and separated by a "~" will appear on a device. If a line is indented then it is a continuation of the previous line and the two combined represent the entire Device Marking for that device.

(6) Lead/Ball Finish - Orderable Devices may have multiple material finish options. Finish options are separated by a vertical ruled line. Lead/Ball Finish values may wrap to two lines if the finish value exceeds the maximum column width.

**Important Information and Disclaimer:** The information provided on this page represents TI's knowledge and belief as of the date that it is provided. TI bases its knowledge and belief on information provided by third parties, and makes no representation or warranty as to the accuracy of such information. Efforts are underway to better integrate information from third parties. TI has taken and



www.ti.com

## PACKAGE OPTION ADDENDUM

17-Aug-2014

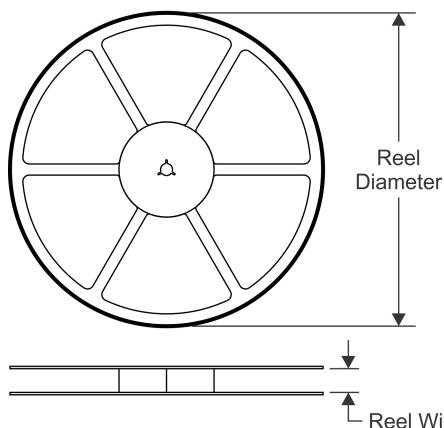
---

continues to take reasonable steps to provide representative and accurate information but may not have conducted destructive testing or chemical analysis on incoming materials and chemicals. TI and TI suppliers consider certain information to be proprietary, and thus CAS numbers and other limited information may not be available for release.

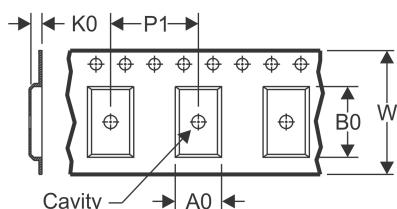
In no event shall TI's liability arising out of such information exceed the total purchase price of the TI part(s) at issue in this document sold by TI to Customer on an annual basis.

## TAPE AND REEL INFORMATION

### REEL DIMENSIONS

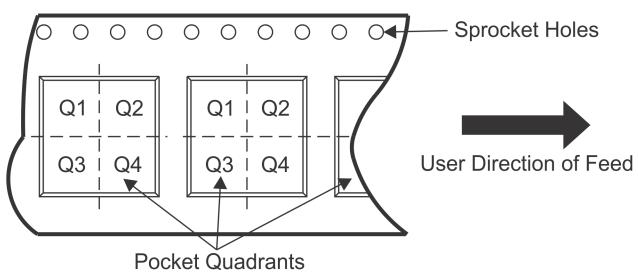


### TAPE DIMENSIONS



A0	Dimension designed to accommodate the component width
B0	Dimension designed to accommodate the component length
K0	Dimension designed to accommodate the component thickness
W	Overall width of the carrier tape
P1	Pitch between successive cavity centers

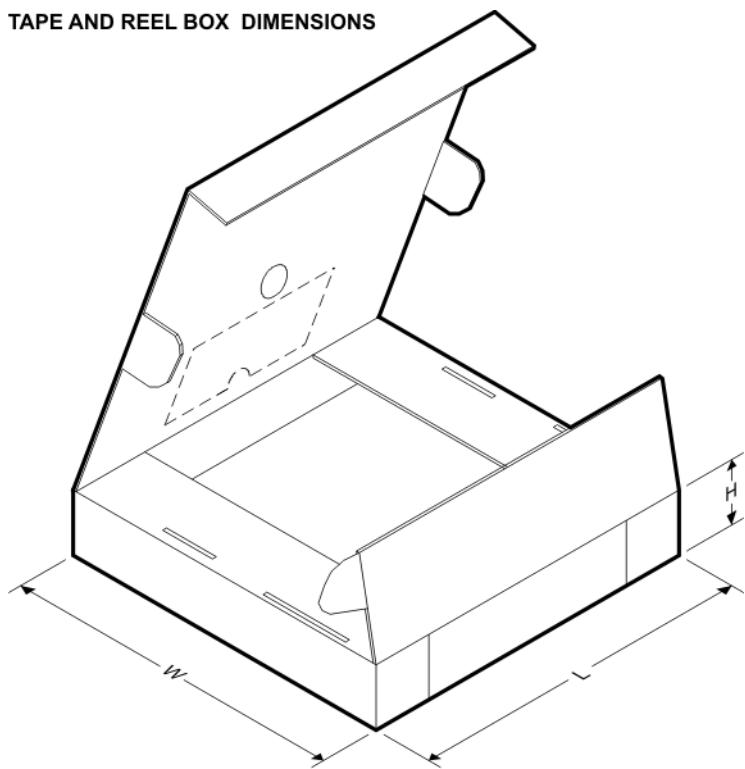
### QUADRANT ASSIGNMENTS FOR PIN 1 ORIENTATION IN TAPE



\*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Reel Diameter (mm)	Reel Width W1 (mm)	A0 (mm)	B0 (mm)	K0 (mm)	P1 (mm)	W (mm)	Pin1 Quadrant
LM3S811-IQN50-C2T	LQFP	PT	48	2000	330.0	16.4	9.6	9.6	1.9	12.0	16.0	Q2

## TAPE AND REEL BOX DIMENSIONS



\*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Length (mm)	Width (mm)	Height (mm)
LM3S811-IQN50-C2T	LQFP	PT	48	2000	367.0	367.0	38.0

## IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as "components") are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have **not** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

<b>Products</b>	<b>Applications</b>		
Audio	<a href="http://www.ti.com/audio">www.ti.com/audio</a>	Automotive and Transportation	<a href="http://www.ti.com/automotive">www.ti.com/automotive</a>
Amplifiers	<a href="http://amplifier.ti.com">amplifier.ti.com</a>	Communications and Telecom	<a href="http://www.ti.com/communications">www.ti.com/communications</a>
Data Converters	<a href="http://dataconverter.ti.com">dataconverter.ti.com</a>	Computers and Peripherals	<a href="http://www.ti.com/computers">www.ti.com/computers</a>
DLP® Products	<a href="http://www.dlp.com">www.dlp.com</a>	Consumer Electronics	<a href="http://www.ti.com/consumer-apps">www.ti.com/consumer-apps</a>
DSP	<a href="http://dsp.ti.com">dsp.ti.com</a>	Energy and Lighting	<a href="http://www.ti.com/energy">www.ti.com/energy</a>
Clocks and Timers	<a href="http://www.ti.com/clocks">www.ti.com/clocks</a>	Industrial	<a href="http://www.ti.com/industrial">www.ti.com/industrial</a>
Interface	<a href="http://interface.ti.com">interface.ti.com</a>	Medical	<a href="http://www.ti.com/medical">www.ti.com/medical</a>
Logic	<a href="http://logic.ti.com">logic.ti.com</a>	Security	<a href="http://www.ti.com/security">www.ti.com/security</a>
Power Mgmt	<a href="http://power.ti.com">power.ti.com</a>	Space, Avionics and Defense	<a href="http://www.ti.com/space-avionics-defense">www.ti.com/space-avionics-defense</a>
Microcontrollers	<a href="http://microcontroller.ti.com">microcontroller.ti.com</a>	Video and Imaging	<a href="http://www.ti.com/video">www.ti.com/video</a>
RFID	<a href="http://www.ti-rfid.com">www.ti-rfid.com</a>	<b>TI E2E Community</b>	
OMAP Applications Processors	<a href="http://www.ti.com/omap">www.ti.com/omap</a>	<a href="http://e2e.ti.com">e2e.ti.com</a>	
Wireless Connectivity	<a href="http://www.ti.com/wirelessconnectivity">www.ti.com/wirelessconnectivity</a>		