



LUMINARY MICRO™

---

# LM3S8962 Microcontroller

DATA SHEET

---

## Legal Disclaimers and Trademark Information

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

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

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

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

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



# Table of Contents

<b>About This Document .....</b>	<b>21</b>
Audience .....	21
About This Manual .....	21
Related Documents .....	21
Documentation Conventions .....	21
<b>1      Architectural Overview .....</b>	<b>23</b>
1.1    Product Features .....	23
1.2    Target Applications .....	29
1.3    High-Level Block Diagram .....	30
1.4    Functional Overview .....	31
1.4.1    ARM Cortex™-M3 .....	32
1.4.2    Motor Control Peripherals .....	32
1.4.3    Analog Peripherals .....	33
1.4.4    Serial Communications Peripherals .....	34
1.4.5    System Peripherals .....	36
1.4.6    Memory Peripherals .....	36
1.4.7    Additional Features .....	37
1.4.8    Hardware Details .....	38
<b>2      ARM Cortex-M3 Processor Core .....</b>	<b>39</b>
2.1    Block Diagram .....	40
2.2    Functional Description .....	40
2.2.1    Serial Wire and JTAG Debug .....	40
2.2.2    Embedded Trace Macrocell (ETM) .....	41
2.2.3    Trace Port Interface Unit (TPIU) .....	41
2.2.4    ROM Table .....	41
2.2.5    Memory Protection Unit (MPU) .....	41
2.2.6    Nested Vectored Interrupt Controller (NVIC) .....	41
<b>3      Memory Map .....</b>	<b>45</b>
<b>4      Interrupts .....</b>	<b>47</b>
<b>5      JTAG Interface .....</b>	<b>50</b>
5.1    Block Diagram .....	51
5.2    Functional Description .....	51
5.2.1    JTAG Interface Pins .....	52
5.2.2    JTAG TAP Controller .....	53
5.2.3    Shift Registers .....	54
5.2.4    Operational Considerations .....	54
5.3    Initialization and Configuration .....	57
5.4    Register Descriptions .....	57
5.4.1    Instruction Register (IR) .....	57
5.4.2    Data Registers .....	59
<b>6      System Control .....</b>	<b>61</b>
6.1    Functional Description .....	61
6.1.1    Device Identification .....	61
6.1.2    Reset Control .....	61

6.1.3	Power Control .....	64
6.1.4	Clock Control .....	64
6.1.5	System Control .....	66
6.2	Initialization and Configuration .....	67
6.3	Register Map .....	67
6.4	Register Descriptions .....	68
<b>7</b>	<b>Hibernation Module .....</b>	<b>122</b>
7.1	Block Diagram .....	123
7.2	Functional Description .....	123
7.2.1	Register Access Timing .....	123
7.2.2	Clock Source .....	124
7.2.3	Battery Management .....	124
7.2.4	Real-Time Clock .....	124
7.2.5	Non-Volatile Memory .....	125
7.2.6	Power Control .....	125
7.2.7	Interrupts and Status .....	125
7.3	Initialization and Configuration .....	126
7.3.1	Initialization .....	126
7.3.2	RTC Match Functionality (No Hibernation) .....	126
7.3.3	RTC Match/Wake-Up from Hibernation .....	126
7.3.4	External Wake-Up from Hibernation .....	127
7.3.5	RTC/External Wake-Up from Hibernation .....	127
7.4	Register Map .....	127
7.5	Register Descriptions .....	128
<b>8</b>	<b>Internal Memory .....</b>	<b>141</b>
8.1	Block Diagram .....	141
8.2	Functional Description .....	141
8.2.1	SRAM Memory .....	141
8.2.2	Flash Memory .....	142
8.3	Flash Memory Initialization and Configuration .....	143
8.3.1	Flash Programming .....	143
8.3.2	Nonvolatile Register Programming .....	144
8.4	Register Map .....	144
8.5	Flash Register Descriptions (Flash Control Offset) .....	145
8.6	Flash Register Descriptions (System Control Offset) .....	152
<b>9</b>	<b>General-Purpose Input/Outputs (GPIOs) .....</b>	<b>165</b>
9.1	Functional Description .....	165
9.1.1	Data Control .....	166
9.1.2	Interrupt Control .....	167
9.1.3	Mode Control .....	168
9.1.4	Commit Control .....	168
9.1.5	Pad Control .....	168
9.1.6	Identification .....	168
9.2	Initialization and Configuration .....	168
9.3	Register Map .....	170
9.4	Register Descriptions .....	171

<b>10</b>	<b>General-Purpose Timers .....</b>	<b>206</b>
10.1	Block Diagram .....	206
10.2	Functional Description .....	207
10.2.1	GPTM Reset Conditions .....	208
10.2.2	32-Bit Timer Operating Modes .....	208
10.2.3	16-Bit Timer Operating Modes .....	209
10.3	Initialization and Configuration .....	213
10.3.1	32-Bit One-Shot/Periodic Timer Mode .....	213
10.3.2	32-Bit Real-Time Clock (RTC) Mode .....	214
10.3.3	16-Bit One-Shot/Periodic Timer Mode .....	214
10.3.4	16-Bit Input Edge Count Mode .....	215
10.3.5	16-Bit Input Edge Timing Mode .....	215
10.3.6	16-Bit PWM Mode .....	216
10.4	Register Map .....	216
10.5	Register Descriptions .....	217
<b>11</b>	<b>Watchdog Timer .....</b>	<b>242</b>
11.1	Block Diagram .....	242
11.2	Functional Description .....	242
11.3	Initialization and Configuration .....	243
11.4	Register Map .....	243
11.5	Register Descriptions .....	244
<b>12</b>	<b>Analog-to-Digital Converter (ADC) .....</b>	<b>265</b>
12.1	Block Diagram .....	266
12.2	Functional Description .....	266
12.2.1	Sample Sequencers .....	266
12.2.2	Module Control .....	267
12.2.3	Hardware Sample Averaging Circuit .....	268
12.2.4	Analog-to-Digital Converter .....	268
12.2.5	Test Modes .....	268
12.2.6	Internal Temperature Sensor .....	268
12.3	Initialization and Configuration .....	269
12.3.1	Module Initialization .....	269
12.3.2	Sample Sequencer Configuration .....	269
12.4	Register Map .....	270
12.5	Register Descriptions .....	271
<b>13</b>	<b>Universal Asynchronous Receivers/Transmitters (UARTs) .....</b>	<b>298</b>
13.1	Block Diagram .....	299
13.2	Functional Description .....	299
13.2.1	Transmit/Receive Logic .....	299
13.2.2	Baud-Rate Generation .....	300
13.2.3	Data Transmission .....	301
13.2.4	Serial IR (SIR) .....	301
13.2.5	FIFO Operation .....	302
13.2.6	Interrupts .....	302
13.2.7	Loopback Operation .....	303
13.2.8	IrDA SIR block .....	303
13.3	Initialization and Configuration .....	303
13.4	Register Map .....	304

13.5	Register Descriptions .....	305
<b>14</b>	<b>Synchronous Serial Interface (SSI) .....</b>	<b>339</b>
14.1	Block Diagram .....	339
14.2	Functional Description .....	339
14.2.1	Bit Rate Generation .....	340
14.2.2	FIFO Operation .....	340
14.2.3	Interrupts .....	340
14.2.4	Frame Formats .....	341
14.3	Initialization and Configuration .....	348
14.4	Register Map .....	349
14.5	Register Descriptions .....	350
<b>15</b>	<b>Inter-Integrated Circuit (I<sup>2</sup>C) Interface .....</b>	<b>376</b>
15.1	Block Diagram .....	376
15.2	Functional Description .....	376
15.2.1	I <sup>2</sup> C Bus Functional Overview .....	377
15.2.2	Available Speed Modes .....	379
15.2.3	Interrupts .....	380
15.2.4	Loopback Operation .....	380
15.2.5	Command Sequence Flow Charts .....	381
15.3	Initialization and Configuration .....	387
15.4	I <sup>2</sup> C Register Map .....	388
15.5	Register Descriptions (I <sup>2</sup> C Master) .....	389
15.6	Register Descriptions (I <sup>2</sup> C Slave) .....	402
<b>16</b>	<b>Controller Area Network (CAN) Module .....</b>	<b>411</b>
16.1	Controller Area Network Overview .....	411
16.2	Controller Area Network Features .....	411
16.3	Controller Area Network Block Diagram .....	412
16.4	Controller Area Network Functional Description .....	413
16.4.1	Initialization .....	413
16.4.2	Operation .....	414
16.4.3	Transmitting Message Objects .....	414
16.4.4	Configuring a Transmit Message Object .....	414
16.4.5	Updating a Transmit Message Object .....	415
16.4.6	Accepting Received Message Objects .....	415
16.4.7	Receiving a Data Frame .....	416
16.4.8	Receiving a Remote Frame .....	416
16.4.9	Receive/Transmit Priority .....	416
16.4.10	Configuring a Receive Message Object .....	416
16.4.11	Handling of Received Message Objects .....	417
16.4.12	Handling of Interrupts .....	417
16.4.13	Bit Timing Configuration Error Considerations .....	418
16.4.14	Bit Time and Bit Rate .....	418
16.4.15	Calculating the Bit Timing Parameters .....	420
16.5	Controller Area Network Register Map .....	422
16.6	Register Descriptions .....	424
<b>17</b>	<b>Ethernet Controller .....</b>	<b>452</b>
17.1	Block Diagram .....	453

17.2	Functional Description .....	453
17.2.1	Internal MII Operation .....	453
17.2.2	PHY Configuration/Operation .....	454
17.2.3	MAC Configuration/Operation .....	455
17.2.4	Interrupts .....	458
17.3	Initialization and Configuration .....	458
17.4	Ethernet Register Map .....	459
17.5	Ethernet MAC Register Descriptions .....	460
17.6	MII Management Register Descriptions .....	478
<b>18</b>	<b>Analog Comparator .....</b>	<b>497</b>
18.1	Block Diagram .....	497
18.2	Functional Description .....	497
18.2.1	Internal Reference Programming .....	498
18.3	Initialization and Configuration .....	499
18.4	Register Map .....	500
18.5	Register Descriptions .....	500
<b>19</b>	<b>Pulse Width Modulator (PWM) .....</b>	<b>508</b>
19.1	Block Diagram .....	508
19.2	Functional Description .....	508
19.2.1	PWM Timer .....	508
19.2.2	PWM Comparators .....	509
19.2.3	PWM Signal Generator .....	510
19.2.4	Dead-Band Generator .....	511
19.2.5	Interrupt/ADC-Trigger Selector .....	511
19.2.6	Synchronization Methods .....	511
19.2.7	Fault Conditions .....	512
19.2.8	Output Control Block .....	512
19.3	Initialization and Configuration .....	512
19.4	Register Map .....	513
19.5	Register Descriptions .....	515
<b>20</b>	<b>Quadrature Encoder Interface (QEI) .....</b>	<b>544</b>
20.1	Block Diagram .....	544
20.2	Functional Description .....	545
20.3	Initialization and Configuration .....	547
20.4	Register Map .....	548
20.5	Register Descriptions .....	548
<b>21</b>	<b>Pin Diagram .....</b>	<b>561</b>
<b>22</b>	<b>Signal Tables .....</b>	<b>562</b>
<b>23</b>	<b>Operating Characteristics .....</b>	<b>576</b>
<b>24</b>	<b>Electrical Characteristics .....</b>	<b>577</b>
24.1	DC Characteristics .....	577
24.1.1	Maximum Ratings .....	577
24.1.2	Recommended DC Operating Conditions .....	577
24.1.3	On-Chip Low Drop-Out (LDO) Regulator Characteristics .....	578
24.1.4	Power Specifications .....	578
24.1.5	Flash Memory Characteristics .....	580

24.2	AC Characteristics .....	580
24.2.1	Load Conditions .....	580
24.2.2	Clocks .....	580
24.2.3	Analog-to-Digital Converter .....	581
24.2.4	Analog Comparator .....	582
24.2.5	I <sup>2</sup> C .....	582
24.2.6	Ethernet Controller .....	583
24.2.7	Hibernation Module .....	586
24.2.8	Synchronous Serial Interface (SSI) .....	586
24.2.9	JTAG and Boundary Scan .....	588
24.2.10	General-Purpose I/O .....	589
24.2.11	Reset .....	590
<b>25</b>	<b>Package Information .....</b>	<b>592</b>
<b>A</b>	<b>Serial Flash Loader .....</b>	<b>594</b>
A.1	Serial Flash Loader .....	594
A.2	Interfaces .....	594
A.2.1	UART .....	594
A.2.2	SSI .....	594
A.3	Packet Handling .....	595
A.3.1	Packet Format .....	595
A.3.2	Sending Packets .....	595
A.3.3	Receiving Packets .....	595
A.4	Commands .....	596
A.4.1	COMMAND_PING (0X20) .....	596
A.4.2	COMMAND_GET_STATUS (0x23) .....	596
A.4.3	COMMAND_DOWNLOAD (0x21) .....	596
A.4.4	COMMAND_SEND_DATA (0x24) .....	597
A.4.5	COMMAND_RUN (0x22) .....	597
A.4.6	COMMAND_RESET (0x25) .....	597
<b>B</b>	<b>Register Quick Reference .....</b>	<b>599</b>
<b>C</b>	<b>Ordering and Contact Information .....</b>	<b>621</b>
C.1	Ordering Information .....	621
C.2	Kits .....	621
C.3	Company Information .....	621
C.4	Support Information .....	622

## List of Figures

Figure 1-1.	Stellaris® 8000 Series High-Level Block Diagram .....	31
Figure 2-1.	CPU Block Diagram .....	40
Figure 2-2.	TPIU Block Diagram .....	41
Figure 5-1.	JTAG Module Block Diagram .....	51
Figure 5-2.	Test Access Port State Machine .....	54
Figure 5-3.	IDCODE Register Format .....	59
Figure 5-4.	BYPASS Register Format .....	60
Figure 5-5.	Boundary Scan Register Format .....	60
Figure 6-1.	External Circuitry to Extend Reset .....	62
Figure 7-1.	Hibernation Module Block Diagram .....	123
Figure 8-1.	Flash Block Diagram .....	141
Figure 9-1.	GPIO Port Block Diagram .....	166
Figure 9-2.	GPIODATA Write Example .....	167
Figure 9-3.	GPIODATA Read Example .....	167
Figure 10-1.	GPTM Module Block Diagram .....	207
Figure 10-2.	16-Bit Input Edge Count Mode Example .....	211
Figure 10-3.	16-Bit Input Edge Time Mode Example .....	212
Figure 10-4.	16-Bit PWM Mode Example .....	213
Figure 11-1.	WDT Module Block Diagram .....	242
Figure 12-1.	ADC Module Block Diagram .....	266
Figure 12-2.	Internal Temperature Sensor Characteristic .....	269
Figure 13-1.	UART Module Block Diagram .....	299
Figure 13-2.	UART Character Frame .....	300
Figure 13-3.	IrDA Data Modulation .....	302
Figure 14-1.	SSI Module Block Diagram .....	339
Figure 14-2.	TI Synchronous Serial Frame Format (Single Transfer) .....	341
Figure 14-3.	TI Synchronous Serial Frame Format (Continuous Transfer) .....	342
Figure 14-4.	Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0 .....	343
Figure 14-5.	Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0 .....	343
Figure 14-6.	Freescale SPI Frame Format with SPO=0 and SPH=1 .....	344
Figure 14-7.	Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0 .....	345
Figure 14-8.	Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0 .....	345
Figure 14-9.	Freescale SPI Frame Format with SPO=1 and SPH=1 .....	346
Figure 14-10.	MICROWIRE Frame Format (Single Frame) .....	347
Figure 14-11.	MICROWIRE Frame Format (Continuous Transfer) .....	348
Figure 14-12.	MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements .....	348
Figure 15-1.	I <sup>2</sup> C Block Diagram .....	376
Figure 15-2.	I <sup>2</sup> C Bus Configuration .....	377
Figure 15-3.	START and STOP Conditions .....	377
Figure 15-4.	Complete Data Transfer with a 7-Bit Address .....	378
Figure 15-5.	R/S Bit in First Byte .....	378
Figure 15-6.	Data Validity During Bit Transfer on the I <sup>2</sup> C Bus .....	378
Figure 15-7.	Master Single SEND .....	381
Figure 15-8.	Master Single RECEIVE .....	382
Figure 15-9.	Master Burst SEND .....	383

Figure 15-10. Master Burst RECEIVE .....	384
Figure 15-11. Master Burst RECEIVE after Burst SEND .....	385
Figure 15-12. Master Burst SEND after Burst RECEIVE .....	386
Figure 15-13. Slave Command Sequence .....	387
Figure 16-1. CAN Module Block Diagram .....	412
Figure 16-2. CAN Bit Time .....	419
Figure 17-1. Ethernet Controller Block Diagram .....	453
Figure 17-2. Ethernet Controller .....	453
Figure 17-3. Ethernet Frame .....	455
Figure 18-1. Analog Comparator Module Block Diagram .....	497
Figure 18-2. Structure of Comparator Unit .....	498
Figure 18-3. Comparator Internal Reference Structure .....	499
Figure 19-1. PWM Module Block Diagram .....	508
Figure 19-2. PWM Count-Down Mode .....	509
Figure 19-3. PWM Count-Up/Down Mode .....	510
Figure 19-4. PWM Generation Example In Count-Up/Down Mode .....	510
Figure 19-5. PWM Dead-Band Generator .....	511
Figure 20-1. QEI Block Diagram .....	545
Figure 20-2. Quadrature Encoder and Velocity Predivider Operation .....	546
Figure 21-1. Pin Connection Diagram .....	561
Figure 24-1. Load Conditions .....	580
Figure 24-2. I <sup>2</sup> C Timing .....	583
Figure 24-3. External XTLP Oscillator Characteristics .....	585
Figure 24-4. Hibernation Module Timing .....	586
Figure 24-5. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement .....	587
Figure 24-6. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer .....	587
Figure 24-7. SSI Timing for SPI Frame Format (FRF=00), with SPH=1 .....	588
Figure 24-8. JTAG Test Clock Input Timing .....	589
Figure 24-9. JTAG Test Access Port (TAP) Timing .....	589
Figure 24-10. JTAG TRST Timing .....	589
Figure 24-11. External Reset Timing ( $\overline{RST}$ ) .....	590
Figure 24-12. Power-On Reset Timing .....	591
Figure 24-13. Brown-Out Reset Timing .....	591
Figure 24-14. Software Reset Timing .....	591
Figure 24-15. Watchdog Reset Timing .....	591
Figure 25-1. 100-Pin LQFP Package .....	592

## List of Tables

Table 1.	Documentation Conventions .....	21
Table 3-1.	Memory Map .....	45
Table 4-1.	Exception Types .....	47
Table 4-2.	Interrupts .....	48
Table 5-1.	JTAG Port Pins Reset State .....	52
Table 5-2.	JTAG Instruction Register Commands .....	57
Table 6-1.	System Control Register Map .....	67
Table 7-1.	Hibernation Module Register Map .....	127
Table 8-1.	Flash Protection Policy Combinations .....	143
Table 8-2.	Flash Resident Registers .....	144
Table 8-3.	Flash Register Map .....	144
Table 9-1.	GPIO Pad Configuration Examples .....	169
Table 9-2.	GPIO Interrupt Configuration Example .....	169
Table 9-3.	GPIO Register Map .....	170
Table 10-1.	Available CCP Pins .....	207
Table 10-2.	16-Bit Timer With Prescaler Configurations .....	210
Table 10-3.	Timers Register Map .....	216
Table 11-1.	Watchdog Timer Register Map .....	243
Table 12-1.	Samples and FIFO Depth of Sequencers .....	266
Table 12-2.	ADC Register Map .....	270
Table 13-1.	UART Register Map .....	304
Table 14-1.	SSI Register Map .....	349
Table 15-1.	Examples of I <sup>2</sup> C Master Timer Period versus Speed Mode .....	379
Table 15-2.	Inter-Integrated Circuit (I <sup>2</sup> C) Interface Register Map .....	388
Table 15-3.	Write Field Decoding for I2CMCS[3:0] Field (Sheet 1 of 3) .....	393
Table 16-1.	Transmit Message Object Bit Settings .....	415
Table 16-2.	Receive Message Object Bit Settings .....	417
Table 16-3.	CAN Protocol Ranges .....	419
Table 16-4.	CAN Register Map .....	422
Table 17-1.	TX & RX FIFO Organization .....	456
Table 17-2.	Ethernet Register Map .....	459
Table 18-1.	Comparator 0 Operating Modes .....	498
Table 18-2.	Internal Reference Voltage and ACREFCTL Field Values .....	499
Table 18-3.	Analog Comparators Register Map .....	500
Table 19-1.	PWM Register Map .....	513
Table 20-1.	QEI Register Map .....	548
Table 22-1.	Signals by Pin Number .....	562
Table 22-2.	Signals by Signal Name .....	566
Table 22-3.	Signals by Function, Except for GPIO .....	570
Table 22-4.	GPIO Pins and Alternate Functions .....	574
Table 23-1.	Temperature Characteristics .....	576
Table 23-2.	Thermal Characteristics .....	576
Table 24-1.	Maximum Ratings .....	577
Table 24-2.	Recommended DC Operating Conditions .....	577
Table 24-3.	LDO Regulator Characteristics .....	578
Table 24-4.	Detailed Power Specifications .....	579

Table 24-5.	Flash Memory Characteristics .....	580
Table 24-6.	Phase Locked Loop (PLL) Characteristics .....	580
Table 24-7.	Clock Characteristics .....	580
Table 24-8.	Crystal Characteristics .....	581
Table 24-9.	ADC Characteristics .....	581
Table 24-10.	Analog Comparator Characteristics .....	582
Table 24-11.	Analog Comparator Voltage Reference Characteristics .....	582
Table 24-12.	I <sup>2</sup> C Characteristics .....	582
Table 24-13.	100BASE-TX Transmitter Characteristics .....	583
Table 24-14.	100BASE-TX Transmitter Characteristics (informative) .....	583
Table 24-15.	100BASE-TX Receiver Characteristics .....	583
Table 24-16.	10BASE-T Transmitter Characteristics .....	583
Table 24-17.	10BASE-T Transmitter Characteristics (informative) .....	584
Table 24-18.	10BASE-T Receiver Characteristics .....	584
Table 24-19.	Isolation Transformers .....	584
Table 24-20.	Ethernet Reference Crystal .....	585
Table 24-21.	External XTLP Oscillator Characteristics .....	585
Table 24-22.	Hibernation Module Characteristics .....	586
Table 24-23.	SSI Characteristics .....	586
Table 24-24.	JTAG Characteristics .....	588
Table 24-25.	GPIO Characteristics .....	590
Table 24-26.	Reset Characteristics .....	590
Table C-1.	Part Ordering Information .....	621

# List of Registers

<b>System Control .....</b>	<b>61</b>
Register 1: Device Identification 0 (DID0), offset 0x000 .....	69
Register 2: Brown-Out Reset Control (PBORCTL), offset 0x030 .....	71
Register 3: LDO Power Control (LDOPCTL), offset 0x034 .....	72
Register 4: Raw Interrupt Status (RIS), offset 0x050 .....	73
Register 5: Interrupt Mask Control (IMC), offset 0x054 .....	74
Register 6: Masked Interrupt Status and Clear (MISC), offset 0x058 .....	75
Register 7: Reset Cause (RESC), offset 0x05C .....	76
Register 8: Run-Mode Clock Configuration (RCC), offset 0x060 .....	77
Register 9: XTAL to PLL Translation (PLLCFG), offset 0x064 .....	81
Register 10: Run-Mode Clock Configuration 2 (RCC2), offset 0x070 .....	82
Register 11: Deep Sleep Clock Configuration (DSLPCLKCFG), offset 0x144 .....	84
Register 12: Device Identification 1 (DID1), offset 0x004 .....	85
Register 13: Device Capabilities 0 (DC0), offset 0x008 .....	87
Register 14: Device Capabilities 1 (DC1), offset 0x010 .....	88
Register 15: Device Capabilities 2 (DC2), offset 0x014 .....	90
Register 16: Device Capabilities 3 (DC3), offset 0x018 .....	92
Register 17: Device Capabilities 4 (DC4), offset 0x01C .....	94
Register 18: Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100 .....	96
Register 19: Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110 .....	98
Register 20: Deep Sleep Mode Clock Gating Control Register 0 (DCGC0), offset 0x120 .....	100
Register 21: Run Mode Clock Gating Control Register 1 (RCGC1), offset 0x104 .....	102
Register 22: Sleep Mode Clock Gating Control Register 1 (SCGC1), offset 0x114 .....	105
Register 23: Deep Sleep Mode Clock Gating Control Register 1 (DCGC1), offset 0x124 .....	108
Register 24: Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108 .....	111
Register 25: Sleep Mode Clock Gating Control Register 2 (SCGC2), offset 0x118 .....	113
Register 26: Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128 .....	115
Register 27: Software Reset Control 0 (SRCR0), offset 0x040 .....	117
Register 28: Software Reset Control 1 (SRCR1), offset 0x044 .....	118
Register 29: Software Reset Control 2 (SRCR2), offset 0x048 .....	120
<b>Hibernation Module .....</b>	<b>122</b>
Register 1: Hibernation RTC Counter (HIBRTCC), offset 0x000 .....	129
Register 2: Hibernation RTC Match 0 (HIBRTCM0), offset 0x004 .....	130
Register 3: Hibernation RTC Match 1 (HIBRTCM1), offset 0x008 .....	131
Register 4: Hibernation RTC Load (HIBRTCLD), offset 0x00C .....	132
Register 5: Hibernation Control (HIBCTL), offset 0x010 .....	133
Register 6: Hibernation Interrupt Mask (HIBIM), offset 0x014 .....	135
Register 7: Hibernation Raw Interrupt Status (HIBRIS), offset 0x018 .....	136
Register 8: Hibernation Masked Interrupt Status (HIBMIS), offset 0x01C .....	137
Register 9: Hibernation Interrupt Clear (HIBIC), offset 0x020 .....	138
Register 10: Hibernation RTC Trim (HIBRTCT), offset 0x024 .....	139
Register 11: Hibernation Data (HIBDATA), offset 0x030-0x12C .....	140
<b>Internal Memory .....</b>	<b>141</b>
Register 1: Flash Memory Address (FMA), offset 0x000 .....	146
Register 2: Flash Memory Data (FMD), offset 0x004 .....	147

Register 3:	Flash Memory Control (FMC), offset 0x008 .....	148
Register 4:	Flash Controller Raw Interrupt Status (FCRIS), offset 0x00C .....	150
Register 5:	Flash Controller Interrupt Mask (FCIM), offset 0x010 .....	151
Register 6:	Flash Controller Masked Interrupt Status and Clear (FCMISC), offset 0x014 .....	152
Register 7:	USec Reload (USECRL), offset 0x140 .....	153
Register 8:	Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x130 and 0x200 .....	154
Register 9:	Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x134 and 0x400 .....	155
Register 10:	User Debug (USER_DBG), offset 0x1D0 .....	156
Register 11:	User Register 0 (USER_REG0), offset 0x1E0 .....	157
Register 12:	User Register 1 (USER_REG1), offset 0x1E4 .....	158
Register 13:	Flash Memory Protection Read Enable 1 (FMPRE1), offset 0x204 .....	159
Register 14:	Flash Memory Protection Read Enable 2 (FMPRE2), offset 0x208 .....	160
Register 15:	Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C .....	161
Register 16:	Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404 .....	162
Register 17:	Flash Memory Protection Program Enable 2 (FMPPE2), offset 0x408 .....	163
Register 18:	Flash Memory Protection Program Enable 3 (FMPPE3), offset 0x40C .....	164
<b>General-Purpose Input/Outputs (GPIOs) .....</b>	<b>165</b>	
Register 1:	GPIO Data (GPIODATA), offset 0x000 .....	172
Register 2:	GPIO Direction (GPIODIR), offset 0x400 .....	173
Register 3:	GPIO Interrupt Sense (GPIOIS), offset 0x404 .....	174
Register 4:	GPIO Interrupt Both Edges (GPIOIBE), offset 0x408 .....	175
Register 5:	GPIO Interrupt Event (GPIOEV), offset 0x40C .....	176
Register 6:	GPIO Interrupt Mask (GPIOIM), offset 0x410 .....	177
Register 7:	GPIO Raw Interrupt Status (GPIORIS), offset 0x414 .....	178
Register 8:	GPIO Masked Interrupt Status (GPIOVIS), offset 0x418 .....	179
Register 9:	GPIO Interrupt Clear (GPIOICR), offset 0x41C .....	180
Register 10:	GPIO Alternate Function Select (GPIOAFSEL), offset 0x420 .....	181
Register 11:	GPIO 2-mA Drive Select (GPIODR2R), offset 0x500 .....	183
Register 12:	GPIO 4-mA Drive Select (GPIODR4R), offset 0x504 .....	184
Register 13:	GPIO 8-mA Drive Select (GPIODR8R), offset 0x508 .....	185
Register 14:	GPIO Open Drain Select (GPIOODR), offset 0x50C .....	186
Register 15:	GPIO Pull-Up Select (GPIOPUR), offset 0x510 .....	187
Register 16:	GPIO Pull-Down Select (GPIOPDR), offset 0x514 .....	188
Register 17:	GPIO Slew Rate Control Select (GPIOSLR), offset 0x518 .....	189
Register 18:	GPIO Digital Enable (GPIODEN), offset 0x51C .....	190
Register 19:	GPIO Lock (GPIOLOCK), offset 0x520 .....	191
Register 20:	GPIO Commit (GPIOCR), offset 0x524 .....	192
Register 21:	GPIO Peripheral Identification 4 (GPIOPeriphID4), offset 0xFD0 .....	194
Register 22:	GPIO Peripheral Identification 5 (GPIOPeriphID5), offset 0xFD4 .....	195
Register 23:	GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8 .....	196
Register 24:	GPIO Peripheral Identification 7 (GPIOPeriphID7), offset 0xFDC .....	197
Register 25:	GPIO Peripheral Identification 0 (GPIOPeriphID0), offset 0xFE0 .....	198
Register 26:	GPIO Peripheral Identification 1 (GPIOPeriphID1), offset 0xFE4 .....	199
Register 27:	GPIO Peripheral Identification 2 (GPIOPeriphID2), offset 0xFE8 .....	200
Register 28:	GPIO Peripheral Identification 3 (GPIOPeriphID3), offset 0xFEC .....	201
Register 29:	GPIO PrimeCell Identification 0 (GPIOPCellID0), offset 0xFF0 .....	202
Register 30:	GPIO PrimeCell Identification 1 (GPIOPCellID1), offset 0xFF4 .....	203
Register 31:	GPIO PrimeCell Identification 2 (GPIOPCellID2), offset 0xFF8 .....	204

Register 32:	GPIO PrimeCell Identification 3 (GPIOPCellID3), offset 0xFFC .....	205
<b>General-Purpose Timers .....</b>		<b>206</b>
Register 1:	GPTM Configuration (GPTMCFG), offset 0x000 .....	218
Register 2:	GPTM TimerA Mode (GPTMTAMR), offset 0x004 .....	219
Register 3:	GPTM TimerB Mode (GPTMTBMR), offset 0x008 .....	221
Register 4:	GPTM Control (GPTMCTL), offset 0x00C .....	223
Register 5:	GPTM Interrupt Mask (GPTMIMR), offset 0x018 .....	226
Register 6:	GPTM Raw Interrupt Status (GPTMRIS), offset 0x01C .....	228
Register 7:	GPTM Masked Interrupt Status (GPTMMIS), offset 0x020 .....	229
Register 8:	GPTM Interrupt Clear (GPTMICR), offset 0x024 .....	230
Register 9:	GPTM TimerA Interval Load (GPTMTAILR), offset 0x028 .....	232
Register 10:	GPTM TimerB Interval Load (GPTMTBILR), offset 0x02C .....	233
Register 11:	GPTM TimerA Match (GPTMTAMATCHR), offset 0x030 .....	234
Register 12:	GPTM TimerB Match (GPTMTBMATCHR), offset 0x034 .....	235
Register 13:	GPTM TimerA Prescale (GPTMTAPR), offset 0x038 .....	236
Register 14:	GPTM TimerB Prescale (GPTMTBPR), offset 0x03C .....	237
Register 15:	GPTM TimerA Prescale Match (GPTMTAPMR), offset 0x040 .....	238
Register 16:	GPTM TimerB Prescale Match (GPTMTBPMR), offset 0x044 .....	239
Register 17:	GPTM TimerA (GPTMTAR), offset 0x048 .....	240
Register 18:	GPTM TimerB (GPTMTBR), offset 0x04C .....	241
<b>Watchdog Timer .....</b>		<b>242</b>
Register 1:	Watchdog Load (WDTLOAD), offset 0x000 .....	245
Register 2:	Watchdog Value (WDTVALUE), offset 0x004 .....	246
Register 3:	Watchdog Control (WDTCTL), offset 0x008 .....	247
Register 4:	Watchdog Interrupt Clear (WDTICR), offset 0x00C .....	248
Register 5:	Watchdog Raw Interrupt Status (WDTRIS), offset 0x010 .....	249
Register 6:	Watchdog Masked Interrupt Status (WDTMIS), offset 0x014 .....	250
Register 7:	Watchdog Test (WDTTEST), offset 0x418 .....	251
Register 8:	Watchdog Lock (WDTLOCK), offset 0xC00 .....	252
Register 9:	Watchdog Peripheral Identification 4 (WDTPeriphID4), offset 0xFD0 .....	253
Register 10:	Watchdog Peripheral Identification 5 (WDTPeriphID5), offset 0xFD4 .....	254
Register 11:	Watchdog Peripheral Identification 6 (WDTPeriphID6), offset 0xFD8 .....	255
Register 12:	Watchdog Peripheral Identification 7 (WDTPeriphID7), offset 0xFDC .....	256
Register 13:	Watchdog Peripheral Identification 0 (WDTPeriphID0), offset 0xFE0 .....	257
Register 14:	Watchdog Peripheral Identification 1 (WDTPeriphID1), offset 0xFE4 .....	258
Register 15:	Watchdog Peripheral Identification 2 (WDTPeriphID2), offset 0xFE8 .....	259
Register 16:	Watchdog Peripheral Identification 3 (WDTPeriphID3), offset 0xFEC .....	260
Register 17:	Watchdog PrimeCell Identification 0 (WDTPCellID0), offset 0xFF0 .....	261
Register 18:	Watchdog PrimeCell Identification 1 (WDTPCellID1), offset 0xFF4 .....	262
Register 19:	Watchdog PrimeCell Identification 2 (WDTPCellID2), offset 0xFF8 .....	263
Register 20:	Watchdog PrimeCell Identification 3 (WDTPCellID3 ), offset 0xFFC .....	264
<b>Analog-to-Digital Converter (ADC) .....</b>		<b>265</b>
Register 1:	ADC Active Sample Sequencer (ADCACTSS), offset 0x000 .....	272
Register 2:	ADC Raw Interrupt Status (ADCRIS), offset 0x004 .....	273
Register 3:	ADC Interrupt Mask (ADCIM), offset 0x008 .....	274
Register 4:	ADC Interrupt Status and Clear (ADCISC), offset 0x00C .....	275
Register 5:	ADC Overflow Status (ADCOSTAT), offset 0x010 .....	276
Register 6:	ADC Event Multiplexer Select (ADCEMUX), offset 0x014 .....	277

Register 7:	ADC Underflow Status (ADCUSTAT), offset 0x018 .....	280
Register 8:	ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020 .....	281
Register 9:	ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028 .....	282
Register 10:	ADC Sample Averaging Control (ADCSAC), offset 0x030 .....	283
Register 11:	ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040 .....	284
Register 12:	ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044 .....	286
Register 13:	ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048 .....	289
Register 14:	ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068 .....	289
Register 15:	ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088 .....	289
Register 16:	ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8 .....	289
Register 17:	ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C .....	290
Register 18:	ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C .....	290
Register 19:	ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C .....	290
Register 20:	ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC .....	290
Register 21:	ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060 .....	291
Register 22:	ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080 .....	291
Register 23:	ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064 .....	292
Register 24:	ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084 .....	292
Register 25:	ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0 .....	294
Register 26:	ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4 .....	295
Register 27:	ADC Test Mode Loopback (ADCTMLB), offset 0x100 .....	296
<b>Universal Asynchronous Receivers/Transmitters (UARTs) .....</b>		<b>298</b>
Register 1:	UART Data (UARTDR), offset 0x000 .....	306
Register 2:	UART Receive Status/Error Clear (UARTRSR/UARTECR), offset 0x004 .....	308
Register 3:	UART Flag (UARTFR), offset 0x018 .....	310
Register 4:	UART IrDA Low-Power Register (UARTILPR), offset 0x020 .....	312
Register 5:	UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024 .....	313
Register 6:	UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028 .....	314
Register 7:	UART Line Control (UARTLCRH), offset 0x02C .....	315
Register 8:	UART Control (UARTCTL), offset 0x030 .....	317
Register 9:	UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034 .....	319
Register 10:	UART Interrupt Mask (UARTIM), offset 0x038 .....	321
Register 11:	UART Raw Interrupt Status (UARTRIS), offset 0x03C .....	323
Register 12:	UART Masked Interrupt Status (UARTMIS), offset 0x040 .....	324
Register 13:	UART Interrupt Clear (UARTICR), offset 0x044 .....	325
Register 14:	UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0 .....	327
Register 15:	UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4 .....	328
Register 16:	UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8 .....	329
Register 17:	UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC .....	330
Register 18:	UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0 .....	331
Register 19:	UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4 .....	332
Register 20:	UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8 .....	333
Register 21:	UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC .....	334
Register 22:	UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0 .....	335
Register 23:	UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4 .....	336
Register 24:	UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8 .....	337
Register 25:	UART PrimeCell Identification 3 (UARTPCellID3), offset 0xFFC .....	338

<b>Synchronous Serial Interface (SSI) .....</b>	<b>339</b>
Register 1: SSI Control 0 (SSICR0), offset 0x000 .....	351
Register 2: SSI Control 1 (SSICR1), offset 0x004 .....	353
Register 3: SSI Data (SSIDR), offset 0x008 .....	355
Register 4: SSI Status (SSISR), offset 0x00C .....	356
Register 5: SSI Clock Prescale (SSICPSR), offset 0x010 .....	358
Register 6: SSI Interrupt Mask (SSIIM), offset 0x014 .....	359
Register 7: SSI Raw Interrupt Status (SSIRIS), offset 0x018 .....	361
Register 8: SSI Masked Interrupt Status (SSIMIS), offset 0x01C .....	362
Register 9: SSI Interrupt Clear (SSIICR), offset 0x020 .....	363
Register 10: SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0 .....	364
Register 11: SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4 .....	365
Register 12: SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8 .....	366
Register 13: SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC .....	367
Register 14: SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0 .....	368
Register 15: SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4 .....	369
Register 16: SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8 .....	370
Register 17: SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC .....	371
Register 18: SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0 .....	372
Register 19: SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4 .....	373
Register 20: SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8 .....	374
Register 21: SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC .....	375
<b>Inter-Integrated Circuit (I<sup>2</sup>C) Interface .....</b>	<b>376</b>
Register 1: I <sup>2</sup> C Master Slave Address (I2CMSA), offset 0x000 .....	390
Register 2: I <sup>2</sup> C Master Control/Status (I2CMCS), offset 0x004 .....	391
Register 3: I <sup>2</sup> C Master Data (I2CMDR), offset 0x008 .....	395
Register 4: I <sup>2</sup> C Master Timer Period (I2CMTPR), offset 0x00C .....	396
Register 5: I <sup>2</sup> C Master Interrupt Mask (I2CMIMR), offset 0x010 .....	397
Register 6: I <sup>2</sup> C Master Raw Interrupt Status (I2CMRIS), offset 0x014 .....	398
Register 7: I <sup>2</sup> C Master Masked Interrupt Status (I2CMMIS), offset 0x018 .....	399
Register 8: I <sup>2</sup> C Master Interrupt Clear (I2CMICR), offset 0x01C .....	400
Register 9: I <sup>2</sup> C Master Configuration (I2CMCR), offset 0x020 .....	401
Register 10: I <sup>2</sup> C Slave Own Address (I2CSOAR), offset 0x000 .....	403
Register 11: I <sup>2</sup> C Slave Control/Status (I2CSCSR), offset 0x004 .....	404
Register 12: I <sup>2</sup> C Slave Data (I2CSDR), offset 0x008 .....	406
Register 13: I <sup>2</sup> C Slave Interrupt Mask (I2CSIMR), offset 0x00C .....	407
Register 14: I <sup>2</sup> C Slave Raw Interrupt Status (I2CSRIS), offset 0x010 .....	408
Register 15: I <sup>2</sup> C Slave Masked Interrupt Status (I2CSMIS), offset 0x014 .....	409
Register 16: I <sup>2</sup> C Slave Interrupt Clear (I2CSICR), offset 0x018 .....	410
<b>Controller Area Network (CAN) Module .....</b>	<b>411</b>
Register 1: CAN Control (CANCTL), offset 0x000 .....	425
Register 2: CAN Status (CANSTS), offset 0x004 .....	427
Register 3: CAN Error Counter (CANERR), offset 0x008 .....	430
Register 4: CAN Bit Timing (CANBIT), offset 0x00C .....	431
Register 5: CAN Interrupt (CANINT), offset 0x010 .....	433
Register 6: CAN Test (CANTST), offset 0x014 .....	434
Register 7: CAN Baud Rate Prescalar Extension (CANBRPE), offset 0x018 .....	436

Register 8:	CAN IF1 Command Request (CANIF1CRQ), offset 0x020 .....	437
Register 9:	CAN IF2 Command Request (CANIF2CRQ), offset 0x080 .....	437
Register 10:	CAN IF1 Command Mask (CANIF1CMSK), offset 0x024 .....	438
Register 11:	CAN IF2 Command Mask (CANIF2CMSK), offset 0x084 .....	438
Register 12:	CAN IF1 Mask 1 (CANIF1MSK1), offset 0x028 .....	441
Register 13:	CAN IF2 Mask 1 (CANIF2MSK1), offset 0x088 .....	441
Register 14:	CAN IF1 Mask 2 (CANIF1MSK2), offset 0x02C .....	442
Register 15:	CAN IF2 Mask 2 (CANIF2MSK2), offset 0x08C .....	442
Register 16:	CAN IF1 Arbitration 1 (CANIF1ARB1), offset 0x030 .....	443
Register 17:	CAN IF2 Arbitration 1 (CANIF2ARB1), offset 0x090 .....	443
Register 18:	CAN IF1 Arbitration 2 (CANIF1ARB2), offset 0x034 .....	444
Register 19:	CAN IF2 Arbitration 2 (CANIF2ARB2), offset 0x094 .....	444
Register 20:	CAN IF1 Message Control (CANIF1MCTL), offset 0x038 .....	445
Register 21:	CAN IF2 Message Control (CANIF2MCTL), offset 0x098 .....	445
Register 22:	CAN IF1 Data A1 (CANIF1DA1), offset 0x03C .....	447
Register 23:	CAN IF1 Data A2 (CANIF1DA2), offset 0x040 .....	447
Register 24:	CAN IF1 Data B1 (CANIF1DB1), offset 0x044 .....	447
Register 25:	CAN IF1 Data B2 (CANIF1DB2), offset 0x048 .....	447
Register 26:	CAN IF2 Data A1 (CANIF2DA1), offset 0x09C .....	447
Register 27:	CAN IF2 Data A2 (CANIF2DA2), offset 0x0A0 .....	447
Register 28:	CAN IF2 Data B1 (CANIF2DB1), offset 0x0A4 .....	447
Register 29:	CAN IF2 Data B2 (CANIF2DB2), offset 0x0A8 .....	447
Register 30:	CAN Transmission Request 1 (CANTXRQ1), offset 0x100 .....	448
Register 31:	CAN Transmission Request 2 (CANTXRQ2), offset 0x104 .....	448
Register 32:	CAN New Data 1 (CANNWDA1), offset 0x120 .....	449
Register 33:	CAN New Data 2 (CANNWDA2), offset 0x124 .....	449
Register 34:	CAN Message 1 Interrupt Pending (CANMSG1INT), offset 0x140 .....	450
Register 35:	CAN Message 2 Interrupt Pending (CANMSG2INT), offset 0x144 .....	450
Register 36:	CAN Message 1 Valid (CANMSG1VAL), offset 0x160 .....	451
Register 37:	CAN Message 2 Valid (CANMSG2VAL), offset 0x164 .....	451
<b>Ethernet Controller .....</b>	<b>452</b>	
Register 1:	Ethernet MAC Raw Interrupt Status (MACRIS), offset 0x000 .....	461
Register 2:	Ethernet MAC Interrupt Acknowledge (MACIACK), offset 0x000 .....	463
Register 3:	Ethernet MAC Interrupt Mask (MACIM), offset 0x004 .....	464
Register 4:	Ethernet MAC Receive Control (MACRCTL), offset 0x008 .....	465
Register 5:	Ethernet MAC Transmit Control (MACTCTL), offset 0x00C .....	466
Register 6:	Ethernet MAC Data (MACDATA), offset 0x010 .....	467
Register 7:	Ethernet MAC Individual Address 0 (MACIA0), offset 0x014 .....	469
Register 8:	Ethernet MAC Individual Address 1 (MACIA1), offset 0x018 .....	470
Register 9:	Ethernet MAC Threshold (MACTR), offset 0x01C .....	471
Register 10:	Ethernet MAC Management Control (MACMCTL), offset 0x020 .....	472
Register 11:	Ethernet MAC Management Divider (MACMDV), offset 0x024 .....	473
Register 12:	Ethernet MAC Management Transmit Data (MACMTXD), offset 0x02C .....	474
Register 13:	Ethernet MAC Management Receive Data (MACMRXD), offset 0x030 .....	475
Register 14:	Ethernet MAC Number of Packets (MACNP), offset 0x034 .....	476
Register 15:	Ethernet MAC Transmission Request (MACTR), offset 0x038 .....	477
Register 16:	Ethernet MAC Timer Support (MACTS), offset 0x03C .....	478
Register 17:	Ethernet PHY Management Register 0 – Control (MR0), address 0x00 .....	479

Register 18:	Ethernet PHY Management Register 1 – Status (MR1), address 0x01 .....	481
Register 19:	Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2), address 0x02 .....	483
Register 20:	Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3), address 0x03 .....	484
Register 21:	Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4), address 0x04 .....	485
Register 22:	Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5), address 0x05 .....	487
Register 23:	Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6), address 0x06 .....	488
Register 24:	Ethernet PHY Management Register 16 – Vendor-Specific (MR16), address 0x10 .....	489
Register 25:	Ethernet PHY Management Register 17 – Interrupt Control/Status (MR17), address 0x11 .....	491
Register 26:	Ethernet PHY Management Register 18 – Diagnostic (MR18), address 0x12 .....	493
Register 27:	Ethernet PHY Management Register 19 – Transceiver Control (MR19), address 0x13 .....	494
Register 28:	Ethernet PHY Management Register 23 – LED Configuration (MR23), address 0x17 .....	495
Register 29:	Ethernet PHY Management Register 24 –MDI/MDIX Control (MR24), address 0x18 .....	496
<b>Analog Comparator .....</b>	<b>497</b>	
Register 1:	Analog Comparator Masked Interrupt Status (ACMIS), offset 0x00 .....	501
Register 2:	Analog Comparator Raw Interrupt Status (ACRIS), offset 0x04 .....	502
Register 3:	Analog Comparator Interrupt Enable (ACINTEN), offset 0x08 .....	503
Register 4:	Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x10 .....	504
Register 5:	Analog Comparator Status 0 (ACSTAT0), offset 0x20 .....	505
Register 6:	Analog Comparator Control 0 (ACCTL0), offset 0x24 .....	506
<b>Pulse Width Modulator (PWM) .....</b>	<b>508</b>	
Register 1:	PWM Master Control (PWMCTL), offset 0x000 .....	516
Register 2:	PWM Time Base Sync (PWMSYNC), offset 0x004 .....	517
Register 3:	PWM Output Enable (PWMCENABLE), offset 0x008 .....	518
Register 4:	PWM Output Inversion (PWMINVERT), offset 0x00C .....	519
Register 5:	PWM Output Fault (PWMFAULT), offset 0x010 .....	520
Register 6:	PWM Interrupt Enable (PWMINTEN), offset 0x014 .....	521
Register 7:	PWM Raw Interrupt Status (PWMRIS), offset 0x018 .....	522
Register 8:	PWM Interrupt Status and Clear (PWMIISC), offset 0x01C .....	523
Register 9:	PWM Status (PWMSSTATUS), offset 0x020 .....	524
Register 10:	PWM0 Control (PWMOCTL), offset 0x040 .....	525
Register 11:	PWM1 Control (PWMI1CTL), offset 0x080 .....	525
Register 12:	PWM2 Control (PWMI2CTL), offset 0x0C0 .....	525
Register 13:	PWM0 Interrupt and Trigger Enable (PWMOINTEN), offset 0x044 .....	527
Register 14:	PWM1 Interrupt and Trigger Enable (PWMIINTEN), offset 0x084 .....	527
Register 15:	PWM2 Interrupt and Trigger Enable (PWMI2INTEN), offset 0x0C4 .....	527
Register 16:	PWM0 Raw Interrupt Status (PWMORIS), offset 0x048 .....	529
Register 17:	PWM1 Raw Interrupt Status (PWMI1RIS), offset 0x088 .....	529
Register 18:	PWM2 Raw Interrupt Status (PWMI2RIS), offset 0x0C8 .....	529
Register 19:	PWM0 Interrupt Status and Clear (PWMOISC), offset 0x04C .....	530
Register 20:	PWM1 Interrupt Status and Clear (PWMI1ISC), offset 0x08C .....	530
Register 21:	PWM2 Interrupt Status and Clear (PWMI2ISC), offset 0x0CC .....	530
Register 22:	PWM0 Load (PWMOLOAD), offset 0x050 .....	531
Register 23:	PWM1 Load (PWMI1LOAD), offset 0x090 .....	531
Register 24:	PWM2 Load (PWMI2LOAD), offset 0x0D0 .....	531

Register 25:	PWM0 Counter (PWM0COUNT), offset 0x054 .....	532
Register 26:	PWM1 Counter (PWM1COUNT), offset 0x094 .....	532
Register 27:	PWM2 Counter (PWM2COUNT), offset 0x0D4 .....	532
Register 28:	PWM0 Compare A (PWM0CMPA), offset 0x058 .....	533
Register 29:	PWM1 Compare A (PWM1CMPA), offset 0x098 .....	533
Register 30:	PWM2 Compare A (PWM2CMPA), offset 0x0D8 .....	533
Register 31:	PWM0 Compare B (PWM0CMPB), offset 0x05C .....	534
Register 32:	PWM1 Compare B (PWM1CMPB), offset 0x09C .....	534
Register 33:	PWM2 Compare B (PWM2CMPB), offset 0x0DC .....	534
Register 34:	PWM0 Generator A Control (PWM0GENA), offset 0x060 .....	535
Register 35:	PWM1 Generator A Control (PWM1GENA), offset 0x0A0 .....	535
Register 36:	PWM2 Generator A Control (PWM2GENA), offset 0x0E0 .....	535
Register 37:	PWM0 Generator B Control (PWM0GENB), offset 0x064 .....	538
Register 38:	PWM1 Generator B Control (PWM1GENB), offset 0x0A4 .....	538
Register 39:	PWM2 Generator B Control (PWM2GENB), offset 0x0E4 .....	538
Register 40:	PWM0 Dead-Band Control (PWM0DBCTL), offset 0x068 .....	541
Register 41:	PWM1 Dead-Band Control (PWM1DBCTL), offset 0x0A8 .....	541
Register 42:	PWM2 Dead-Band Control (PWM2DBCTL), offset 0x0E8 .....	541
Register 43:	PWM0 Dead-Band Rising-Edge Delay (PWM0DBRISE), offset 0x06C .....	542
Register 44:	PWM1 Dead-Band Rising-Edge Delay (PWM1DBRISE), offset 0x0AC .....	542
Register 45:	PWM2 Dead-Band Rising-Edge Delay (PWM2DBRISE), offset 0x0EC .....	542
Register 46:	PWM0 Dead-Band Falling-Edge-Delay (PWM0DBFALL), offset 0x070 .....	543
Register 47:	PWM1 Dead-Band Falling-Edge-Delay (PWM1DBFALL), offset 0x0B0 .....	543
Register 48:	PWM2 Dead-Band Falling-Edge-Delay (PWM2DBFALL), offset 0x0F0 .....	543
<b>Quadrature Encoder Interface (QEI) .....</b>	<b>544</b>	
Register 1:	QEI Control (QEICTL), offset 0x000 .....	549
Register 2:	QEI Status (QEISTAT), offset 0x004 .....	551
Register 3:	QEI Position (QEIPOS), offset 0x008 .....	552
Register 4:	QEI Maximum Position (QEIMAXPOS), offset 0x00C .....	553
Register 5:	QEI Timer Load (QEILOAD), offset 0x010 .....	554
Register 6:	QEI Timer (QEITIME), offset 0x014 .....	555
Register 7:	QEI Velocity Counter (QEICOUNT), offset 0x018 .....	556
Register 8:	QEI Velocity (QEISPEED), offset 0x01C .....	557
Register 9:	QEI Interrupt Enable (QEINTEN), offset 0x020 .....	558
Register 10:	QEI Raw Interrupt Status (QEIRIS), offset 0x024 .....	559
Register 11:	QEI Interrupt Status and Clear (QEISC), offset 0x028 .....	560

# About This Document

This data sheet provides reference information for the LM3S8962 microcontroller, describing the functional blocks of the system-on-chip (SoC) device designed around the ARM® Cortex™-M3 core.

## Audience

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

## About This Manual

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

## Related Documents

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

- *ARM® Cortex™-M3 Technical Reference Manual*
- *ARM® CoreSight Technical Reference Manual*
- *ARM® v7-M Architecture Application Level Reference Manual*

The following related documents are also referenced:

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

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

## Documentation Conventions

This document uses the conventions shown in Table 1 on page 21.

**Table 1. Documentation Conventions**

Notation	Meaning
<b>General Register Notation</b>	
<b>REGISTER</b>	APB registers are indicated in uppercase bold. For example, <b>PBORCTL</b> is the Power-On and Brown-Out Reset Control register. If a register name contains a lowercase n, it represents more than one register. For example, <b>SRCRn</b> represents any (or all) of the three Software Reset Control registers: <b>SRCR0</b> , <b>SRCR1</b> , and <b>SRCR2</b> .
bit	A single bit in a register.
bit field	Two or more consecutive and related bits.
offset 0xnnn	A hexadecimal increment to a register's address, relative to that module's base address as specified in "Memory Map" on page 45.
Register N	Registers are numbered consecutively throughout the document to aid in referencing them. The register number has no meaning to software.

Notation	Meaning
reserved	Register bits marked <i>reserved</i> are reserved for future use. In most cases, reserved bits are set to 0; however, user software should not rely on the value of a reserved bit. To provide software compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
yy:xx	The range of register bits inclusive from xx to yy. For example, 31:15 means bits 15 through 31 in that register.
<b>Register Bit/Field Types</b>	This value in the register bit diagram indicates whether software running on the controller can change the value of the bit field.
RC	Software can read this field. The bit or field is cleared by hardware after reading the bit/field.
RO	Software can read this field. Always write the chip reset value.
R/W	Software can read or write this field.
R/W1C	Software can read or write this field. A write of a 0 to a W1C bit does not affect the bit value in the register. A write of a 1 clears the value of the bit in the register; the remaining bits remain unchanged.  This register type is primarily used for clearing interrupt status bits where the read operation provides the interrupt status and the write of the read value clears only the interrupts being reported at the time the register was read.
W1C	Software can write this field. A write of a 0 to a W1C bit does not affect the bit value in the register. A write of a 1 clears the value of the bit in the register; the remaining bits remain unchanged. A read of the register returns no meaningful data.  This register is typically used to clear the corresponding bit in an interrupt register.
WO	Only a write by software is valid; a read of the register returns no meaningful data.
<b>Register Bit/Field Reset Value</b>	This value in the register bit diagram shows the bit/field value after any reset, unless noted.
0	Bit cleared to 0 on chip reset.
1	Bit set to 1 on chip reset.
-	Nondeterministic.
<b>Pin/Signal Notation</b>	
[]	Pin alternate function; a pin defaults to the signal without the brackets.
pin	Refers to the physical connection on the package.
signal	Refers to the electrical signal encoding of a pin.
assert a signal	Change the value of the signal from the logically False state to the logically True state. For active High signals, the asserted signal value is 1 (High); for active Low signals, the asserted signal value is 0 (Low). The active polarity (High or Low) is defined by the signal name (see 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.
SIGNAL	Signal names are in uppercase and in the Courier font. An overbar on a signal name indicates that it is active Low. To assert SIGNAL is to drive it Low; to deassert SIGNAL 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 Luminary Micro Stellaris® family of microcontrollers—the first ARM® Cortex™-M3 based controllers—brings high-performance 32-bit computing to cost-sensitive embedded microcontroller applications. These pioneering parts deliver customers 32-bit performance at a cost equivalent to legacy 8- and 16-bit devices, all in a package with a small footprint.

The Stellaris® family offers efficient performance and extensive integration, favorably positioning the device into cost-conscious applications requiring significant control-processing and connectivity capabilities. The Stellaris® LM3S1000 series extends the Stellaris® family with larger on-chip memories, enhanced power management, and expanded I/O and control capabilities. The Stellaris® LM3S2000 series, designed for Controller Area Network (CAN) applications, extends the Stellaris family with Bosch CAN networking technology, the golden standard in short-haul industrial networks. The Stellaris® LM3S2000 series also marks the first integration of CAN capabilities with the revolutionary Cortex-M3 core. The Stellaris® LM3S6000 series combines both a 10/100 Ethernet Media Access Control (MAC) and Physical (PHY) layer, marking the first time that integrated connectivity is available with an ARM Cortex-M3 MCU and the only integrated 10/100 Ethernet MAC and PHY available in an ARM architecture MCU. The Stellaris® LM3S8000 series combines Bosch Controller Area Network technology with both a 10/100 Ethernet Media Access Control (MAC) and Physical (PHY) layer.

The LM3S8962 microcontroller is targeted for industrial applications, including remote monitoring, electronic point-of-sale machines, test and measurement equipment, network appliances and switches, factory automation, HVAC and building control, gaming equipment, motion control, medical instrumentation, and fire and security.

For applications requiring extreme conservation of power, the LM3S8962 microcontroller features a Battery-backed Hibernation module to efficiently power down the LM3S8962 to a low-power state during extended periods of inactivity. With a power-up/power-down sequencer, a continuous time counter (RTC), a pair of match registers, an APB interface to the system bus, and dedicated non-volatile memory, the Hibernation module positions the LM3S8962 microcontroller perfectly for battery applications.

In addition, the LM3S8962 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 LM3S8962 microcontroller is code-compatible to all members of the extensive Stellaris® family; providing flexibility to fit our customers' precise needs.

Luminary Micro offers a complete solution to get to market quickly, with evaluation and development boards, white papers and application notes, an easy-to-use peripheral driver library, and a strong support, sales, and distributor network.

## 1.1 Product Features

The LM3S8962 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
  - 36 interrupts with eight priority levels
- Memory protection unit (MPU), providing a privileged mode for protected operating system functionality
- Unaligned data access, enabling data to be efficiently packed into memory
- Atomic bit manipulation (bit-banding), delivering maximum memory utilization and streamlined peripheral control
- Internal Memory
  - 256 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
  - 64 KB single-cycle SRAM
- General-Purpose Timers
  - Four General-Purpose Timer Modules (GPTM), each of which provides two 16-bit timers. Each GPTM can be configured to operate independently:
    - As a single 32-bit timer
    - As one 32-bit Real-Time Clock (RTC) to event capture
    - For Pulse Width Modulation (PWM)
    - To trigger analog-to-digital conversions
  - 32-bit Timer modes
    - Programmable one-shot timer
    - Programmable periodic timer
    - Real-Time Clock when using an external 32.768-KHz clock as the input

- User-enabled stalling in periodic and one-shot mode when the controller asserts the CPU Halt flag during debug
- ADC event trigger
- 16-bit Timer modes
  - General-purpose timer function with an 8-bit prescaler
  - Programmable one-shot timer
  - Programmable periodic timer
  - User-enabled stalling when the controller asserts CPU Halt flag during debug
  - ADC event trigger
- 16-bit Input Capture modes
  - Input edge count capture
  - Input edge time capture
- 16-bit PWM mode
  - Simple PWM mode with software-programmable output inversion of the PWM signal
- ARM FiRM-compliant Watchdog Timer
  - 32-bit down counter with a programmable load register
  - Separate watchdog clock with an enable
  - Programmable interrupt generation logic with interrupt masking
  - Lock register protection from runaway software
  - Reset generation logic with an enable/disable
  - User-enabled stalling when the controller asserts the CPU Halt flag during debug
- Controller Area Network (CAN)
  - Supports CAN protocol version 2.0 part A/B
  - Bit rates up to 1Mb/s
  - 32 message objects, each with its own identifier mask
  - Maskable interrupt
  - Disable automatic retransmission mode for TTCAN
  - Programmable loop-back mode for self-test operation
- 10/100 Ethernet Controller

- Conforms to the IEEE 802.3-2002 Specification
- IEEE 1588-2002 Precision Time Protocol (PTP) compliant
- Full- and half-duplex for both 100 Mbps and 10 Mbps operation
- Integrated 10/100 Mbps Transceiver (PHY)
- Automatic MDI/MDI-X cross-over correction
- Programmable MAC address
- Power-saving and power-down modes
- 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
- UART
  - Two fully programmable 16C550-type UARTs with IrDA support
  - Separate 16x8 transmit (TX) and 16x12 receive (RX) FIFOs to reduce CPU interrupt service loading
  - Programmable baud-rate generator with fractional divider
  - Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
  - FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
  - Standard asynchronous communication bits for start, stop, and parity
  - False-start-bit detection
  - Line-break generation and detection
- ADC
  - Single- and differential-input configurations
  - Four 10-bit channels (inputs) when used as single-ended inputs
  - Sample rate of 500 thousand samples/second

- Flexible, configurable analog-to-digital conversion
- Four programmable sample conversion sequences from one to eight entries long, with corresponding conversion result FIFOs
- Each sequence triggered by software or internal event (timers, analog comparators, PWM or GPIO)
- On-chip temperature sensor
- 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
- I<sup>2</sup>C
  - Master and slave receive and transmit operation with transmission speed up to 100 Kbps in Standard mode and 400 Kbps in Fast mode
  - Interrupt generation
  - Master with arbitration and clock synchronization, multimaster support, and 7-bit addressing mode
- PWM
  - Three PWM generator blocks, each with one 16-bit counter, two comparators, a PWM generator, and a dead-band generator
  - 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
  - Synchronization of timer/comparator updates across the PWM generator blocks
  - Interrupt status summary of the PWM generator blocks
- Can initiate an ADC sample sequence
- QEI
  - Two QEI modules
  - Hardware position integrator tracks the encoder position
  - Velocity capture using built-in timer
  - Interrupt generation on index pulse, velocity-timer expiration, direction change, and quadrature error detection
- GPIOs
  - 5-42 GPIOs, depending on configuration
  - 5-V-tolerant input/outputs
  - Programmable interrupt generation as either edge-triggered or level-sensitive
  - Bit masking in both read and write operations through address lines
  - Can initiate an ADC sample sequence
  - Programmable control for GPIO pad configuration:
    - Weak pull-up or pull-down resistors
    - 2-mA, 4-mA, and 8-mA pad drive
    - Slew rate control for the 8-mA drive
    - Open drain enables
    - Digital input enables

- Power
  - On-chip Low Drop-Out (LDO) voltage regulator, with programmable output user-adjustable from 2.25 V to 2.75 V
  - Hibernation module handles the power-up/down 3.3 V sequencing and control for the core digital logic and analog circuits
  - Low-power options on controller: Sleep and Deep-sleep modes
  - Low-power options for peripherals: software controls shutdown of individual peripherals
  - User-enabled LDO unregulated voltage detection and automatic reset
  - 3.3-V supply brown-out detection and reporting via interrupt or reset
- Flexible Reset Sources
  - Power-on reset (POR)
  - Reset pin assertion
  - Brown-out (BOR) detector alerts to system power drops
  - Software reset
  - Watchdog timer reset
  - Internal low drop-out (LDO) regulator output goes unregulated
- Additional Features
  - Six reset sources
  - Programmable clock source control
  - Clock gating to individual peripherals for power savings
  - IEEE 1149.1-1990 compliant Test Access Port (TAP) controller
  - Debug access via JTAG and Serial Wire interfaces
  - Full JTAG boundary scan
- Industrial-range 100-pin RoHS-compliant LQFP package

## 1.2 Target Applications

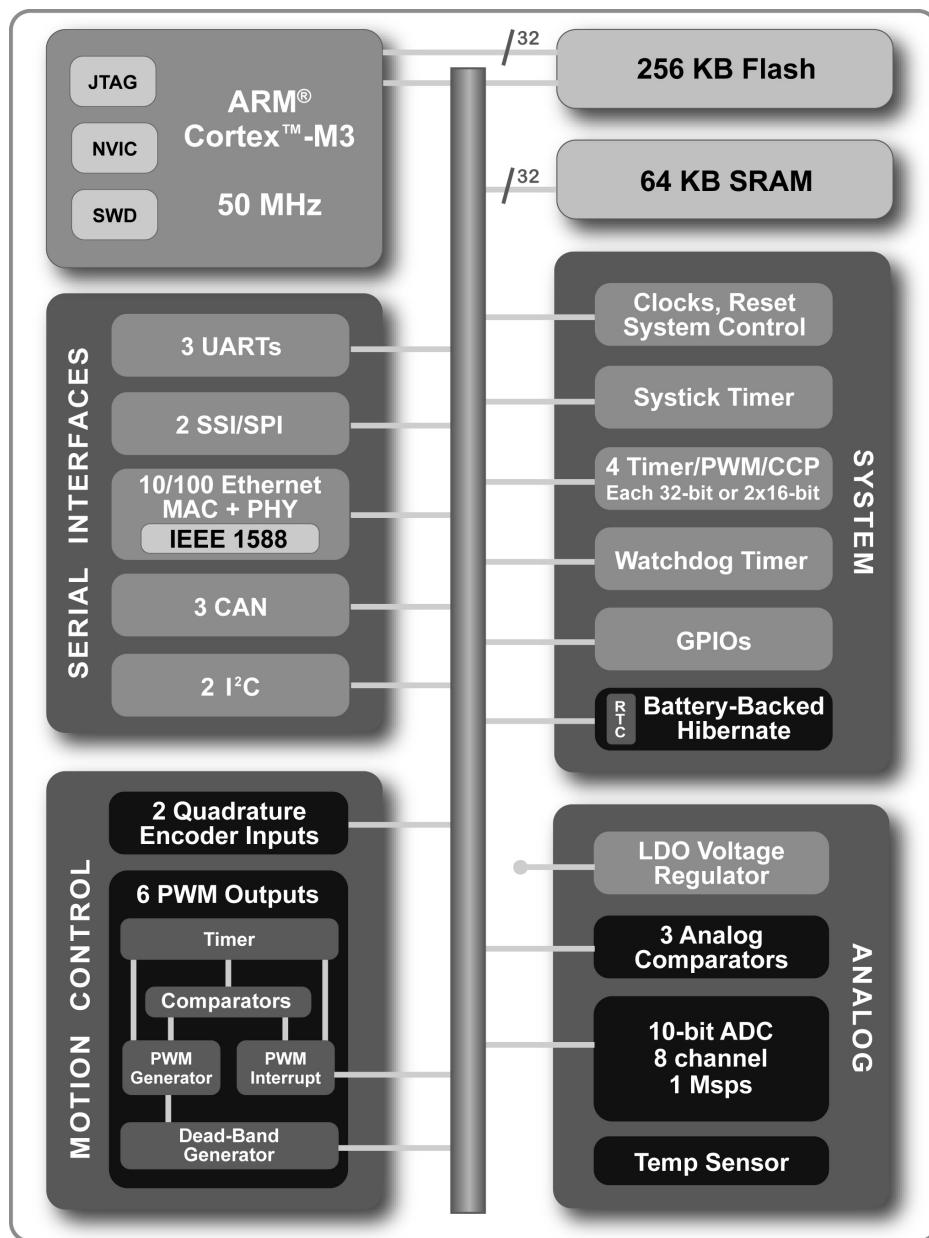
- Remote monitoring
- Electronic point-of-sale (POS) machines
- Test and measurement equipment
- Network appliances and switches

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

### **1.3 High-Level Block Diagram**

Figure 1-1 on page 31 represents the full set of features in the Stellaris® 8000 series of devices; not all features may be available on the LM3S8962 microcontroller.

**Figure 1-1. Stellaris® 8000 Series High-Level Block Diagram**



## 1.4 Functional Overview

The following sections provide an overview of the features of the LM3S8962 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 621.

## 1.4.1 ARM Cortex™-M3

### 1.4.1.1 Processor Core (see page 39)

All members of the Stellaris® product family, including the LM3S8962 microcontroller, are designed around an ARM Cortex™-M3 processor core. The ARM Cortex-M3 processor provides the core for a high-performance, low-cost platform that meets the needs of minimal memory implementation, reduced pin count, and low-power consumption, while delivering outstanding computational performance and exceptional system response to interrupts.

“ARM Cortex-M3 Processor Core” on page 39 provides an overview of the ARM core; the core is detailed in the *ARM® Cortex™-M3 Technical Reference Manual*.

### 1.4.1.2 System Timer (SysTick)

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

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

### 1.4.1.3 Nested Vectored Interrupt Controller (NVIC)

The LM3S8962 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 36 interrupts.

“Interrupts” on page 47 provides an overview of the NVIC controller and the interrupt map. Exceptions and interrupts are detailed in the *ARM® Cortex™-M3 Technical Reference Manual*.

## 1.4.2 Motor Control Peripherals

To enhance motor control, the LM3S8962 controller features Pulse Width Modulation (PWM) outputs and the Quadrature Encoder Interface (QEI).

### 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 LM3S8962, 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 508)**

The LM3S8962 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 212)**

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

### **1.4.2.2 QEI (see page 544)**

A quadrature encoder, also known as a 2-channel incremental encoder, converts linear displacement into a pulse signal. By monitoring both the number of pulses and the relative phase of the two signals, you can track the position, direction of rotation, and speed. In addition, a third channel, or index signal, can be used to reset the position counter.

The Stellaris quadrature encoder with index (QEI) module interprets the code produced by a quadrature encoder wheel to integrate position over time and determine direction of rotation. In addition, it can capture a running estimate of the velocity of the encoder wheel. The LM3S8962 microcontroller includes two QEI modules, which enables control of two motors at the same time.

### **1.4.3 Analog Peripherals**

To handle analog signals, the LM3S8962 microcontroller offers an Analog-to-Digital Converter (ADC).

For support of analog signals, the LM3S8962 microcontroller offers one analog comparator.

#### **1.4.3.1 ADC (see page 265)**

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

The LM3S8962 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 497)**

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

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 LM3S8962 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
- One CAN unit
- Ethernet controller

### 1.4.4.1 UART (see page 298)

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 LM3S8962 controller includes two fully programmable 16C550-type UARTs that support data transfer speeds up to 460.8 Kbps. (Although similar in functionality to a 16C550 UART, it is not register-compatible.) In addition, each UART is capable of supporting IrDA.

Separate 16x8 transmit (TX) and 16x12 receive (RX) FIFOs reduce CPU interrupt service loading. The UART can generate individually masked interrupts from the RX, TX, modem status, and error conditions. The module provides a single combined interrupt when any of the interrupts are asserted and are unmasked.

### 1.4.4.2 SSI (see page 339)

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

The LM3S8962 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 376)

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 LM3S8962 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.

#### 1.4.4.4 Controller Area Network (see page 411)

Controller Area Network (CAN) is a multicast shared serial-bus standard for connecting electronic control units (ECUs). CAN was specifically designed to be robust in electromagnetically noisy environments and can utilize a differential balanced line like RS-485 or a more robust twisted-pair wire. Originally created for automotive purposes, now it is used in many embedded control applications (for example, industrial or medical). Bit rates up to 1Mb/s are possible at network lengths below 40 meters. Decreased bit rates allow longer network distances (for example, 125 Kb/s at 500m).

A transmitter sends a message to all CAN nodes (broadcasting). Each node decides on the basis of the identifier received whether it should process the message. The identifier also determines the priority that the message enjoys in competition for bus access. Each CAN message can transmit from 0 to 8 bytes of user information. The LM3S8962 includes one CAN units.

#### 1.4.4.5 Ethernet Controller (see page 452)

Ethernet is a frame-based computer networking technology for local area networks (LANs). Ethernet has been standardized as IEEE 802.3. It defines a number of wiring and signaling standards for the physical layer, two means of network access at the Media Access Control (MAC)/Data Link Layer, and a common addressing format.

The Stellaris® Ethernet Controller consists of a fully integrated media access controller (MAC) and network physical (PHY) interface device. The Ethernet Controller conforms to IEEE 802.3 specifications and fully supports 10BASE-T and 100BASE-TX standards. In addition, the Ethernet Controller supports automatic MDI/MDI-X cross-over correction.

## 1.4.5 System Peripherals

### 1.4.5.1 Programmable GPIOs (see page 165)

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

The Stellaris® GPIO module is composed of seven 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 5-42 programmable input/output pins. The number of GPIOs available depends on the peripherals being used (see “Signal Tables” on page 562 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.

### 1.4.5.2 Four Programmable Timers (see page 206)

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 four GPTM blocks. Each GPTM block provides two 16-bit timers/counters that can be configured to operate independently as timers or event counters, or configured to operate as one 32-bit timer or one 32-bit Real-Time Clock (RTC). Timers can also be used to trigger analog-to-digital (ADC) conversions.

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

### 1.4.5.3 Watchdog Timer (see page 242)

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

The Stellaris® 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 LM3S8962 controller offers both single-cycle SRAM and single-cycle Flash memory.

### 1.4.6.1 SRAM (see page 141)

The LM3S8962 static random access memory (SRAM) controller supports 64 KB SRAM. The internal SRAM of the Stellaris® devices is located at offset 0x0000.0000 of the device memory map. To reduce the number of time-consuming read-modify-write (RMW) operations, ARM has introduced *bit-banding* technology in the new Cortex-M3 processor. With a bit-band-enabled processor, certain regions in the memory map (SRAM and peripheral space) can use address aliases to access individual bits in a single, atomic operation.

### 1.4.6.2 Flash (see page 142)

The LM3S8962 Flash controller supports 256 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 Memory Map (see page 45)

A memory map lists the location of instructions and data in memory. The memory map for the LM3S8962 controller can be found in “Memory Map” on page 45. Register addresses are given as a hexadecimal increment, relative to the module’s base address as shown in the memory map.

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

### 1.4.7.2 JTAG TAP Controller (see page 50)

The Joint Test Action Group (JTAG) port provides a standardized serial interface for controlling the Test Access Port (TAP) and associated test logic. The TAP, JTAG instruction register, and JTAG data registers can be used to test the interconnects of assembled printed circuit boards, obtain manufacturing information on the components, and observe and/or control the inputs and outputs of the controller during normal operation. The JTAG port provides a high degree of testability and chip-level access at a low cost.

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

### 1.4.7.3 System Control and Clocks (see page 61)

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

### 1.4.7.4 Hibernation Module (see page 122)

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

#### **1.4.8     Hardware Details**

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

- “Pin Diagram” on page 561
- “Signal Tables” on page 562
- “Operating Characteristics” on page 576
- “Electrical Characteristics” on page 577
- “Package Information” on page 592

## 2 ARM Cortex-M3 Processor Core

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

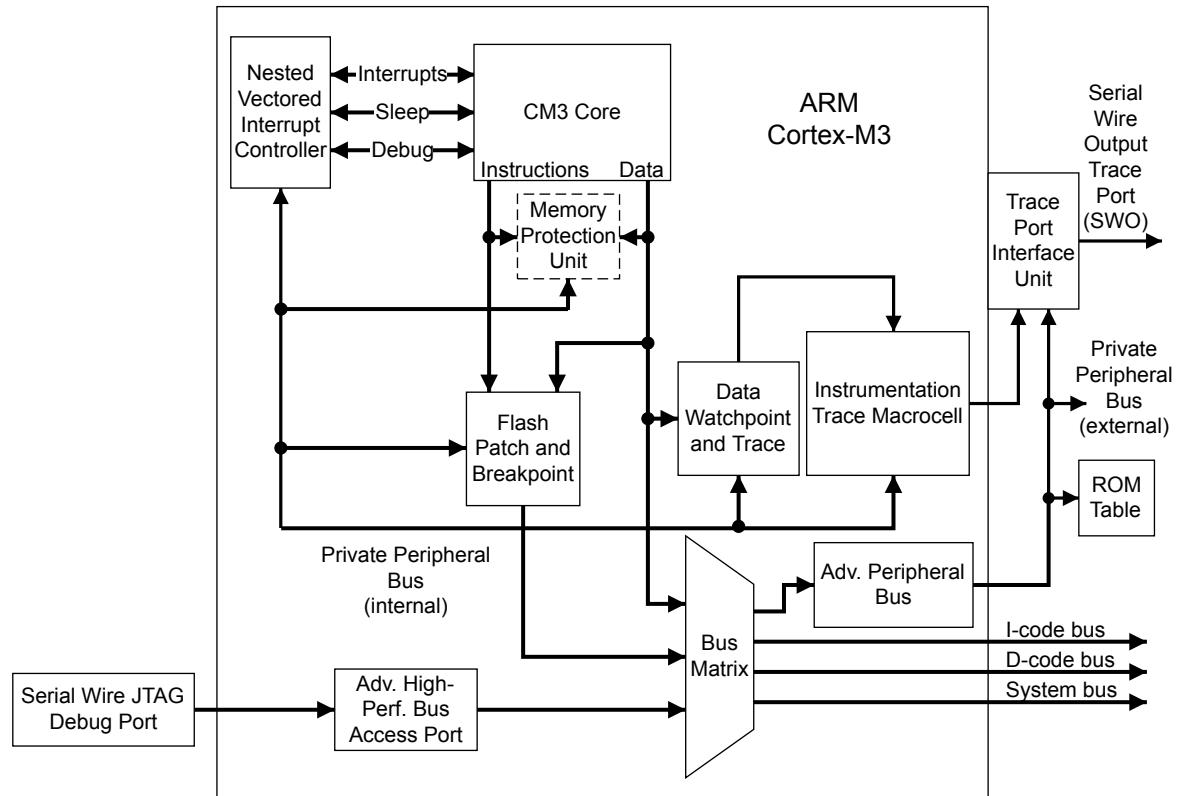
- Compact core.
- Thumb-2 instruction set, delivering the high-performance expected of an ARM core in the memory size usually associated with 8- and 16-bit devices; typically in the range of a few kilobytes of memory for microcontroller class applications.
- Rapid application execution through Harvard architecture characterized by separate buses for instruction and data.
- Exceptional interrupt handling, by implementing the register manipulations required for handling an interrupt in hardware.
- Memory protection unit (MPU) to provide a privileged mode of operation for complex applications.
- Migration from the ARM7™ processor family for better performance and power efficiency.
- Full-featured debug solution with a:
  - Serial Wire JTAG Debug Port (SWJ-DP)
  - Flash Patch and Breakpoint (FPB) unit for implementing breakpoints
  - Data Watchpoint and Trigger (DWT) unit for implementing watchpoints, trigger resources, and system profiling
  - Instrumentation Trace Macrocell (ITM) for support of printf style debugging
  - Trace Port Interface Unit (TPIU) for bridging to a Trace Port Analyzer

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 motors.

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

## 2.1 Block Diagram

Figure 2-1. CPU Block Diagram



## 2.2 Functional Description

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

Luminary Micro has implemented the ARM Cortex-M3 core as shown in Figure 2-1 on page 40. As noted in the *ARM® Cortex™-M3 Technical Reference Manual*, several Cortex-M3 components are flexible in their implementation: SW/JTAG-DP, ETM, TPIU, the ROM table, the MPU, and the Nested Vectored Interrupt Controller (NVIC). Each of these is addressed in the sections that follow.

### 2.2.1 Serial Wire and JTAG Debug

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

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

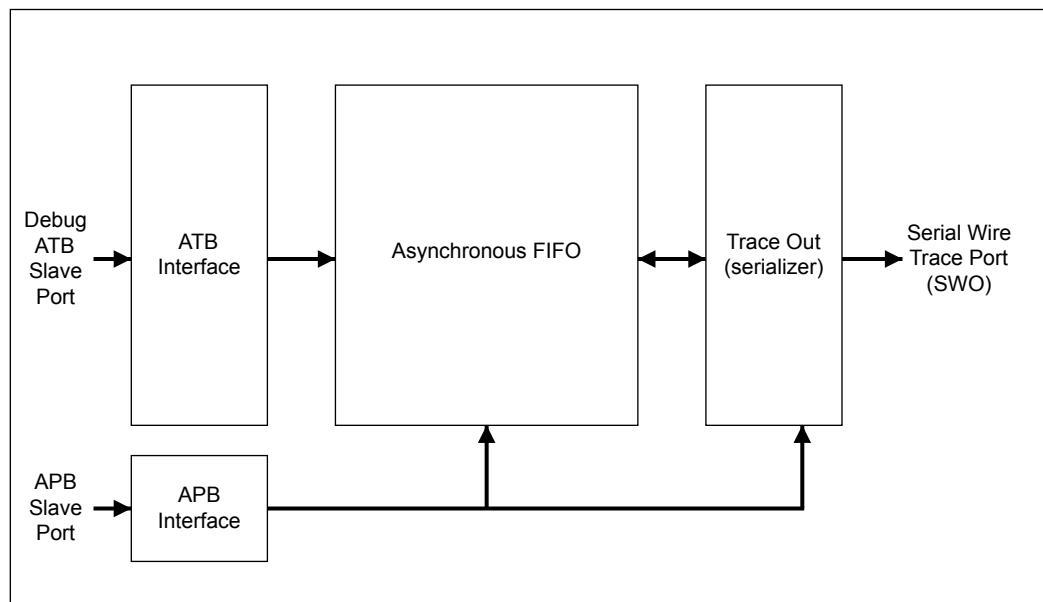
## 2.2.2 Embedded Trace Macrocell (ETM)

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

## 2.2.3 Trace Port Interface Unit (TPIU)

The TPIU acts as a bridge between the Cortex-M3 trace data from the ITM, and an off-chip Trace Port Analyzer. The Stellaris® devices have implemented TPIU as shown in Figure 2-2 on page 41. This is similar to the non-ETM version described in the ARM® Cortex™-M3 Technical Reference Manual, however, SWJ-DP only provides SWV output for the TPIU.

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



## 2.2.4 ROM Table

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

## 2.2.5 Memory Protection Unit (MPU)

The Memory Protection Unit (MPU) is included on the LM3S8962 controller and supports the standard ARMv7 Protected Memory System Architecture (PMSA) model. The MPU provides full support for protection regions, overlapping protection regions, access permissions, and exporting memory attributes to the system.

## 2.2.6 Nested Vectored Interrupt Controller (NVIC)

The Nested Vectored Interrupt Controller (NVIC):

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

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

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

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

All NVIC registers and system debug registers are little endian regardless of the endianness state of the processor.

### 2.2.6.1 Interrupts

The *ARM® Cortex™-M3 Technical Reference Manual* describes the maximum number of interrupts and interrupt priorities. The LM3S8962 microcontroller supports 36 interrupts with eight priority levels.

### 2.2.6.2 System Timer (SysTick)

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

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

#### *Functional Description*

The timer consists of three registers:

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

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

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

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

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

### SysTick Control and Status Register

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

Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	COUNTFLAG	R/W	0	Returns 1 if timer counted to 0 since last time this was read. Clears on read by application. If read by the debugger using the DAP, this bit is cleared on read-only if the MasterType bit in the AHB-AP Control Register is set to 0. Otherwise, the COUNTFLAG bit is not changed by the debugger read.
15:3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	CLKSOURCE	R/W	0	0 = external reference clock. (Not implemented for Stellaris microcontrollers.) 1 = core clock.  If no reference clock is provided, it is held at 1 and so gives the same time as the core clock. The core clock must be at least 2.5 times faster than the reference clock. If it is not, the count values are unpredictable.
1	TICKINT	R/W	0	1 = counting down to 0 pends the SysTick handler. 0 = counting down to 0 does not pend the SysTick handler. Software can use the COUNTFLAG to determine if ever counted to 0.
0	ENABLE	R/W	0	1 = counter operates in a multi-shot way. That is, counter loads with the Reload value and then begins counting down. On reaching 0, it sets the COUNTFLAG to 1 and optionally pends the SysTick handler, based on TICKINT. It then loads the Reload value again, and begins counting. 0 = counter disabled.

### SysTick Reload Value Register

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

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

Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
23:0	RELOAD	W1C	-	Value to load into the SysTick Current Value Register when the counter reaches 0.

### SysTick Current Value Register

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

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

### SysTick Calibration Value Register

The SysTick Calibration Value register is not implemented.

### 3 Memory Map

The memory map for the LM3S8962 controller is provided in Table 3-1 on page 45.

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

**Important:** In Table 3-1 on page 45, addresses not listed are reserved.

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

Start	End	Description	For details on registers, see page ...
<b>Memory</b>			
0x0000.0000	0x0003.FFFF	On-chip flash <sup>b</sup>	145
0x2000.0000	0x2000.FFFF	Bit-banded on-chip SRAM <sup>c</sup>	145
0x2010.0000	0x21FF.FFFF	Reserved non-bit-banded SRAM space	-
0x2200.0000	0x23FF.FFFF	Bit-band alias of 0x2000.0000 through 0x200F.FFFF	141
0x2400.0000	0x3FFF.FFFF	Reserved non-bit-banded SRAM space	-
<b>FiRM Peripherals</b>			
0x4000.0000	0x4000.0FFF	Watchdog timer	244
0x4000.4000	0x4000.4FFF	GPIO Port A	171
0x4000.5000	0x4000.5FFF	GPIO Port B	171
0x4000.6000	0x4000.6FFF	GPIO Port C	171
0x4000.7000	0x4000.7FFF	GPIO Port D	171
0x4000.8000	0x4000.8FFF	SSIO	350
0x4000.C000	0x4000.CFFF	UART0	305
0x4000.D000	0x4000.DFFF	UART1	305
<b>Peripherals</b>			
0x4002.0000	0x4002.07FF	I2C Master 0	389
0x4002.0800	0x4002.0FFF	I2C Slave 0	402
0x4002.4000	0x4002.4FFF	GPIO Port E	171
0x4002.5000	0x4002.5FFF	GPIO Port F	171
0x4002.6000	0x4002.6FFF	GPIO Port G	171
0x4002.8000	0x4002.8FFF	PWM	515
0x4002.C000	0x4002.CFFF	QEIO	548
0x4002.D000	0x4002.DFFF	QEII	548
0x4003.0000	0x4003.0FFF	Timer0	217
0x4003.1000	0x4003.1FFF	Timer1	217
0x4003.2000	0x4003.2FFF	Timer2	217
0x4003.3000	0x4003.3FFF	Timer3	217
0x4003.8000	0x4003.8FFF	ADC	271
0x4003.C000	0x4003.CFFF	Analog Comparators	497
0x4004.0000	0x4004.0FFF	CAN0 Controller	424

Start	End	Description	For details on registers, see page ...
0x4004.8000	0x4004.8FFF	Ethernet Controller	460
0x400F.C000	0x400F.CFFF	Hibernation Module	128
0x400F.D000	0x400F.DFFF	Flash control	145
0x400F.E000	0x400F.EFFF	System control	68
0x4200.0000	0x43FF.FFFF	Bit-banded alias of 0x4000.0000 through 0x400F.FFFF	-
<b>Private Peripheral Bus</b>			
0xE000.0000	0xE000.0FFF	Instrumentation Trace Macrocell (ITM)	ARM® Cortex™-M3 Technical Reference Manual
0xE000.1000	0xE000.1FFF	Data Watchpoint and Trace (DWT)	
0xE000.2000	0xE000.2FFF	Flash Patch and Breakpoint (FPB)	
0xE000.3000	0xE000.DFFF	Reserved	
0xE000.E000	0xE000.EFFF	Nested Vectored Interrupt Controller (NVIC)	
0xE000.F000	0xE003.FFFF	Reserved	
0xE004.0000	0xE004.0FFF	Trace Port Interface Unit (TPIU)	
0xE004.1000	0xE004.1FFF	Reserved	
0xE004.2000	0xE00F.FFFF	Reserved	
0xE010.0000	0xFFFF.FFFF	Reserved for vendor peripherals	

a. All reserved space returns a bus fault when read or written.

b. The unavailable flash will bus fault throughout this range.

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

## 4 Interrupts

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

Table 4-1 on page 47 lists all the exceptions. Software can set eight priority levels on seven of these exceptions (system handlers) as well as on 36 interrupts (listed in Table 4-2 on page 48).

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

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

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

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

**Note:** In Table 4-2 on page 48 interrupts not listed are reserved.

**Table 4-1. Exception Types**

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

Exception Type	Position	Priority <sup>a</sup>	Description
Debug Monitor	12	settable	Debug monitor (when not halting). This is synchronous, but only active when enabled. It does not activate if lower priority than the current activation.
-	13	-	Reserved.
PendSV	14	settable	Pendable request for system service. This is asynchronous and only pended by software.
SysTick	15	settable	System tick timer has fired. This is asynchronous.
Interrupts	16 and above	settable	Asserted from outside the ARM Cortex-M3 core and fed through the NVIC (prioritized). These are all asynchronous. Table 4-2 on page 48 lists the interrupts on the LM3S8962 controller.

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

**Table 4-2. Interrupts**

Interrupt (Bit in Interrupt Registers)	Description
0	GPIO Port A
1	GPIO Port B
2	GPIO Port C
3	GPIO Port D
4	GPIO Port E
5	UART0
6	UART1
7	SSI0
8	I2C0
9	PWM Fault
10	PWM Generator 0
11	PWM Generator 1
12	PWM Generator 2
13	QEI0
14	ADC Sequence 0
15	ADC Sequence 1
16	ADC Sequence 2
17	ADC Sequence 3
18	Watchdog timer
19	Timer0 A
20	Timer0 B
21	Timer1 A
22	Timer1 B
23	Timer2 A
24	Timer2 B
25	Analog Comparator 0
28	System Control
29	Flash Control
30	GPIO Port F
31	GPIO Port G

Interrupt (Bit in Interrupt Registers)	Description
35	Timer3 A
36	Timer3 B
38	QEI1
39	CAN0
42	Ethernet Controller
43	Hibernation Module

## 5 JTAG Interface

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

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

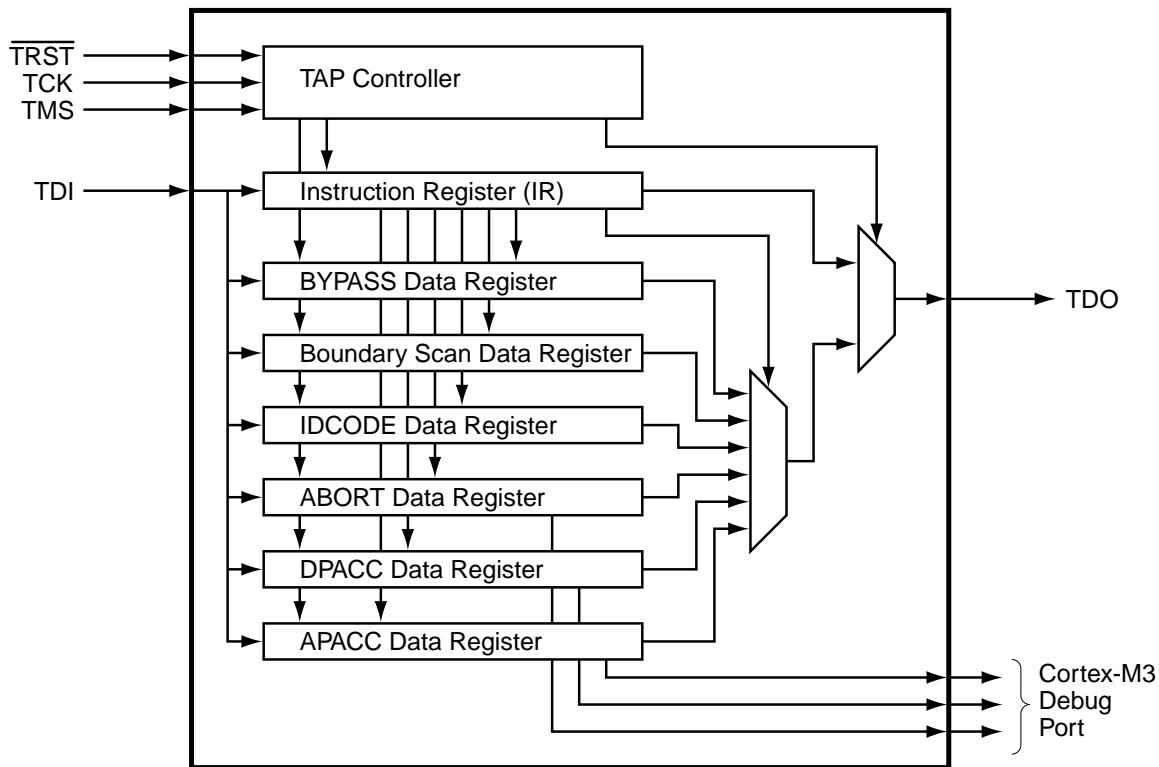
The JTAG module has the following features:

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

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

## 5.1 Block Diagram

Figure 5-1. JTAG Module Block Diagram



## 5.2 Functional Description

A high-level conceptual drawing of the JTAG module is shown in Figure 5-1 on page 51. 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 `TRST`, `TCK` and `TMS` inputs. The current state of the TAP controller depends on the current value of `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 5-2 on page 57 for a list of implemented instructions).

See “JTAG and Boundary Scan” on page 588 for JTAG timing diagrams.

## 5.2.1 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 5-1 on page 52. Detailed information on each pin follows.

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

Pin Name	Data Direction	Internal Pull-Up	Internal Pull-Down	Drive Strength	Drive Value
$\overline{\text{TRST}}$	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

### 5.2.1.1 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.

### 5.2.1.2 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.

### 5.2.1.3 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 5-2 on page 54.

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

#### 5.2.1.4 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.

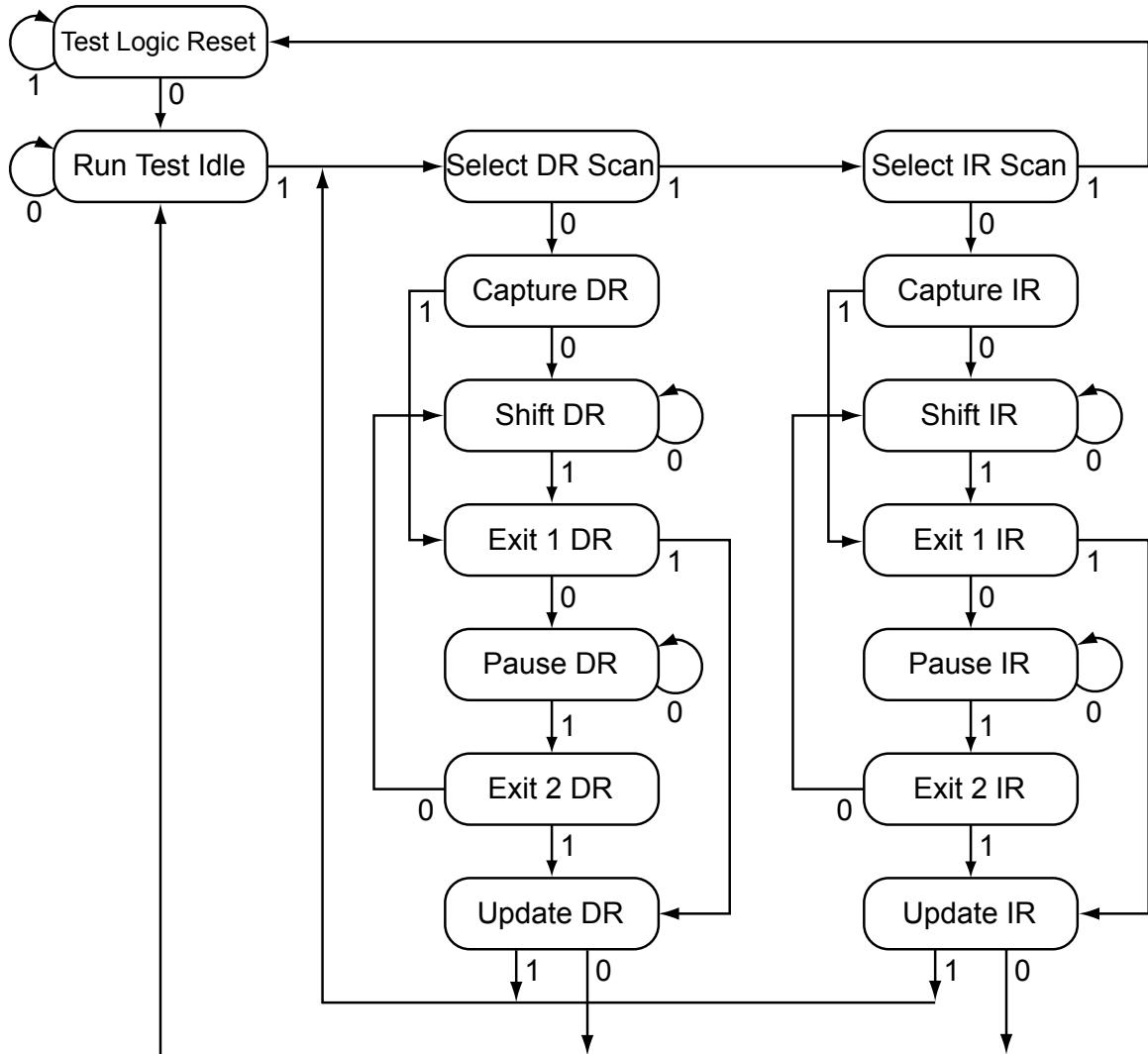
#### 5.2.1.5 Test Data Output (TDO)

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.

### 5.2.2 JTAG TAP Controller

The JTAG TAP controller state machine is shown in Figure 5-2 on page 54. 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 5-2. Test Access Port State Machine**

### 5.2.3 Shift Registers

The Shift Registers consist of a serial shift register chain and a parallel load register. The serial shift register chain samples specific information during the TAP controller's CAPTURE states and allows this information to be shifted out of TDO during the TAP controller's SHIFT states. While the sampled data is being shifted out of the chain on TDO, new data is being shifted into the serial shift register on TDI. This new data is stored in the parallel load register during the TAP controller's UPDATE states. Each of the shift registers is discussed in detail in "Register Descriptions" on page 57.

### 5.2.4 Operational Considerations

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

### 5.2.4.1 GPIO Functionality

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

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

The commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 181) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 191) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 192) have been set to 1.

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

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

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

12. Release the  $\overline{\text{RST}}$  signal.

The JTAG-to-SWD and SWD-to-JTAG switch sequences are described in “ARM Serial Wire Debug (SWD)” on page 56. When performing switch sequences for the purpose of recovering the debug capabilities of the device, only steps 1 and 2 of the switch sequence need to be performed.

### 5.2.4.2 ARM Serial Wire Debug (SWD)

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

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

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

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

#### JTAG-to-SWD Switching

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

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

#### SWD-to-JTAG Switching

To switch the operating mode of the Debug Access Port (DAP) from SWD to JTAG mode, the external debug hardware must send a switch sequence to the device. The 16-bit switch sequence for switching to JTAG mode is defined as b1110011110011110, transmitted LSB first. This can also be represented as 16'hE73C when transmitted LSB first. The complete switch sequence should consist of the following transactions on the TCK/SWCLK and TMS/SWDIO signals:

1. Send at least 50 TCK/SWCLK cycles with TMS/SWDIO set to 1. This ensures that both JTAG and SWD are in their reset/idle states.

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

## 5.3 Initialization and Configuration

After a Power-On-Reset or an external reset ( $\overline{RST}$ ), the JTAG pins are automatically configured for JTAG communication. No user-defined initialization or configuration is needed. However, if the user application changes these pins to their GPIO function, they must be configured back to their JTAG functionality before JTAG communication can be restored. This is done by enabling the five JTAG pins ( $PB7$  and  $PC[3:0]$ ) for their alternate function using the **GPIOAFSEL** register.

## 5.4 Register Descriptions

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

### 5.4.1 Instruction Register (IR)

The JTAG TAP Instruction Register (IR) is a four-bit serial scan chain with a parallel load register connected between the JTAG TDI and TDO pins. When the TAP Controller is placed in the correct states, bits can be shifted into the Instruction Register. Once these bits have been shifted into the chain and updated, they are interpreted as the current instruction. The decode of the Instruction Register bits is shown in Table 5-2 on page 57. A detailed explanation of each instruction, along with its associated Data Register, follows.

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

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

#### 5.4.1.1 EXTEST Instruction

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

tests to be developed that drive known values out of the controller, which can be used to verify connectivity.

#### 5.4.1.2 INTEST Instruction

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

#### 5.4.1.3 SAMPLE/PRELOAD Instruction

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

While these samples of the inputs, outputs, and output enables are being shifted out of the Boundary Scan Data Register, new data is being shifted into the Boundary Scan Data Register from TDI. Once the new data has been shifted into the Boundary Scan Data Register, the data is saved in the parallel load registers when the TAP controller enters the Update DR state. This update of the parallel load register preloads data into the Boundary Scan Data Register that is associated with each input, output, and output enable. This preloaded data can be used with the EXTEST and INTEST instructions to drive data into or out of the controller. Please see “Boundary Scan Data Register” on page 60 for more information.

#### 5.4.1.4 ABORT Instruction

The ABORT instruction connects the associated ABORT Data Register chain between TDI and TDO. This instruction provides read and write access to the ABORT Register of the ARM Debug Access Port (DAP). Shifting the proper data into this Data Register clears various error bits or initiates a DAP abort of a previous request. Please see the “ABORT Data Register” on page 60 for more information.

#### 5.4.1.5 DPACC Instruction

The DPACC instruction connects the associated DPACC Data Register chain between TDI and TDO. This instruction provides read and write access to the DPACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to the ARM debug and status registers. Please see “DPACC Data Register” on page 60 for more information.

#### 5.4.1.6 APACC Instruction

The APACC instruction connects the associated APACC Data Register chain between TDI and TDO. This instruction provides read and write access to the APACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to internal components and buses through the Debug Port. Please see “APACC Data Register” on page 60 for more information.

#### 5.4.1.7 IDCODE Instruction

The IDCODE instruction connects the associated IDCODE Data Register chain between TDI and TDO. This instruction provides information on the manufacturer, part number, and version of the ARM core. This information can be used by testing equipment and debuggers to automatically configure their input and output data streams. IDCODE is the default instruction that is loaded into the JTAG Instruction Register when a power-on-reset (POR) is asserted, TRST is asserted, or the Test-Logic-Reset state is entered. Please see “IDCODE Data Register” on page 59 for more information.

#### 5.4.1.8 BYPASS Instruction

The BYPASS instruction connects the associated BYPASS Data Register chain between TDI and TDO. This instruction is used to create a minimum length serial path between the TDI and TDO ports. The BYPASS Data Register is a single-bit shift register. This instruction improves test efficiency by allowing components that are not needed for a specific test to be bypassed in the JTAG scan chain by loading them with the BYPASS instruction. Please see “BYPASS Data Register” on page 59 for more information.

### 5.4.2 Data Registers

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

#### 5.4.2.1 IDCODE Data Register

The format for the 32-bit IDCODE Data Register defined by the *IEEE Standard 1149.1* is shown in Figure 5-3 on page 59. The standard requires that every JTAG-compliant device implement either the IDCODE instruction or the BYPASS instruction as the default instruction. The LSB of the IDCODE Data Register is defined to be a 1 to distinguish it from the BYPASS instruction, which has an LSB of 0. This allows auto configuration test tools to determine which instruction is the default instruction.

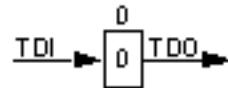
The major uses of the JTAG port are for manufacturer testing of component assembly, and program development and debug. To facilitate the use of auto-configuration debug tools, the IDCODE instruction outputs a value of 0x3BA00477. This value indicates an ARM Cortex-M3, Version 1 processor. This allows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

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



#### 5.4.2.2 BYPASS Data Register

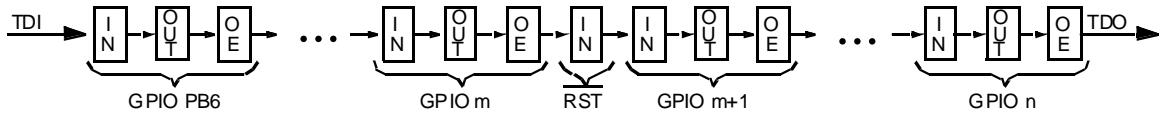
The format for the 1-bit BYPASS Data Register defined by the *IEEE Standard 1149.1* is shown in Figure 5-4 on page 60. The standard requires that every JTAG-compliant device implement either the BYPASS instruction or the IDCODE instruction as the default instruction. The LSB of the BYPASS Data Register is defined to be a 0 to distinguish it from the IDCODE instruction, which has an LSB of 1. This allows auto configuration test tools to determine which instruction is the default instruction.

**Figure 5-4. BYPASS Register Format**

#### 5.4.2.3 Boundary Scan Data Register

The format of the Boundary Scan Data Register is shown in Figure 5-5 on page 60. Each GPIO pin, in a counter-clockwise direction from the JTAG port pins, is included in the Boundary Scan Data Register. Each GPIO pin has three associated digital signals that are included in the chain. These signals are input, output, and output enable, and are arranged in that order as can be seen in the figure. In addition to the GPIO pins, the controller reset pin, RST, is included in the chain. Because the reset pin is always an input, only the input signal is included in the Data Register chain.

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

**Figure 5-5. Boundary Scan Register Format**

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

#### 5.4.2.4 APACC Data Register

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

#### 5.4.2.5 DPACC Data Register

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

#### 5.4.2.6 ABORT Data Register

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

# 6 System Control

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

## 6.1 Functional Description

The System Control module provides the following capabilities:

- Device identification, see “Device Identification” on page 61
- Local control, such as reset (see “Reset Control” on page 61), power (see “Power Control” on page 64) and clock control (see “Clock Control” on page 64)
- System control (Run, Sleep, and Deep-Sleep modes), see “System Control” on page 66

### 6.1.1 Device Identification

Seven read-only registers provide software with information on the microcontroller, such as version, part number, SRAM size, flash size, and other features. See the **DID0**, **DID1**, and **DC0-DC4** registers.

### 6.1.2 Reset Control

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

#### 6.1.2.1 CMOD0 and CMOD1 Test-Mode Control Pins

Two pins, **CMOD0** and **CMOD1**, are defined for use by Luminary Micro for testing the devices during manufacture. They have no end-user function and should not be used. The **CMOD** pins should be connected to ground.

#### 6.1.2.2 Reset Sources

The controller has five sources of reset:

1. External reset input pin ( $\overline{\text{RST}}$ ) assertion, see “ $\overline{\text{RST}}$  Pin Assertion” on page 61.
2. Power-on reset (POR), see “Power-On Reset (POR)” on page 62.
3. Internal brown-out (BOR) detector, see “Brown-Out Reset (BOR)” on page 62.
4. Software-initiated reset (with the software reset registers), see “Software Reset” on page 63.
5. A watchdog timer reset condition violation, see “Watchdog Timer Reset” on page 63.

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

#### 6.1.2.3 $\overline{\text{RST}}$ Pin Assertion

The external reset pin ( $\overline{\text{RST}}$ ) resets the controller. This resets the core and all the peripherals except the JTAG TAP controller (see “JTAG Interface” on page 50). The external reset sequence is as follows:

1. The external reset pin ( $\overline{RST}$ ) is asserted and then de-asserted.
2. The internal reset is released and the core loads from memory the initial stack pointer, the initial program counter, the first instruction designated by the program counter, and begins execution. A few clock cycles from  $\overline{RST}$  de-assertion to the start of the reset sequence is necessary for synchronization.

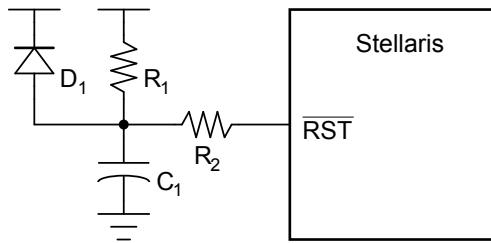
The external reset timing is shown in Figure 24-11 on page 590.

#### 6.1.2.4 Power-On Reset (POR)

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

The device must be operating within the specified operating parameters at the point when the on-chip power-on reset pulse is complete. The 3.3-V power supply to the device must reach 3.0 V within 10 msec of it crossing 2.0 V to guarantee proper operation. For applications that require the use of an external reset to hold the device in reset longer than the internal POR, the  $\overline{RST}$  input may be used with the circuit as shown in Figure 6-1 on page 62.

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



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

The Power-On Reset sequence is as follows:

1. The controller waits for the later of external reset ( $\overline{RST}$ ) or 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, the first instruction designated by the program counter, and begins execution.

The internal POR is only active on the initial power-up of the controller. The Power-On Reset timing is shown in Figure 24-12 on page 591.

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

#### 6.1.2.5 Brown-Out Reset (BOR)

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

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

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

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

The internal Brown-Out Reset timing is shown in Figure 24-13 on page 591.

### 6.1.2.6 Software Reset

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

Peripherals can be individually reset by software via three registers that control reset signals to each peripheral (see the **SRCRn** registers). If the bit position corresponding to a peripheral is set and subsequently cleared, the peripheral is reset. The encoding of the reset registers is consistent with the encoding of the clock gating control for peripherals and on-chip functions (see “System Control” on page 66). 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 24-14 on page 591.

### 6.1.2.7 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 24-15 on page 591.

### 6.1.3 Power Control

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

**Note:** The use of the LDO is optional. The internal logic may be supplied by the on-chip LDO or by an external regulator. If the LDO is used, the LDO output pin is connected to the VDD25 pins on the printed circuit board. The LDO requires decoupling capacitors on the printed circuit board. If an external regulator is used, it is strongly recommended that the external regulator supply the controller only and not be shared with other devices on the printed circuit board.

### 6.1.4 Clock Control

System control determines the control of clocks in this part.

#### 6.1.4.1 Fundamental Clock Sources

There are four clock sources for use in the device:

- **Internal Oscillator (IOSC):** The internal oscillator is an on-chip clock source. It does not require the use of any external components. The frequency of the internal oscillator is  $12\text{ MHz} \pm 30\%$ . Applications that do not depend on accurate clock sources may use this clock source to reduce system cost. The internal oscillator is the clock source the device uses during and following POR. If the main oscillator is required, software must enable the main oscillator following reset and allow the main oscillator to stabilize before changing the clock reference.
- **Main Oscillator:** 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 in the **RCC** register (see page 77).
- **Internal 30-kHz Oscillator:** The internal 30-kHz oscillator is similar to the internal oscillator, except that it provides an operational frequency of  $30\text{ kHz} \pm 30\%$ . It is intended for use during Deep-Sleep power-saving modes. This power-savings mode benefits from reduced internal switching and also allows the main oscillator to be powered down.
- **External Real-Time Oscillator:** The external real-time oscillator provides a low-frequency, accurate clock reference. It is intended to provide the system with a real-time clock source. The real-time oscillator is part of the Hibernation Module (“Hibernation Module” on page 122) and may also provide an accurate source of Deep-Sleep or Hibernate mode power savings.

The internal system clock (sysclk), is derived from any of the four sources plus two others: the output of the internal PLL, and the internal oscillator divided by four ( $3\text{ MHz} \pm 30\%$ ). The frequency of the PLL clock reference must be in the range of 3.579545 MHz to 8.192 MHz (inclusive).

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

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

The main oscillator supports the use of a select number of crystals. If the main oscillator is used by the PLL as a reference clock, the supported range of crystals is 3.579545 to 8.192 MHz, otherwise, the range of supported crystals is 1 to 8.192 MHz.

The **XTAL** bit in the **RCC** register (see page 77) describes the available crystal choices and default programming values.

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

#### 6.1.4.3 PLL Frequency Configuration

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

If the main oscillator provides the clock reference to the 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 81). The internal translation provides a translation within  $\pm 1\%$  of the targeted PLL VCO frequency.

The Crystal Value field (**XTAL**) on page 77 describes the available crystal choices and default programming of the **PLLCFG** register. The crystal number is written into the **XTAL** field of the **Run-Mode Clock Configuration (RCC)** register. Any time the **XTAL** field changes, the new settings are translated and the internal PLL settings are updated.

#### 6.1.4.4 PLL Modes

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

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

The modes are programmed using the **RCC/RCC2** register fields (see page 77 and page 82).

#### 6.1.4.5 PLL Operation

If the 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 24-6 on page 580). During this time, the PLL is not usable as a clock reference.

The PLL is changed by one of the following:

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

A counter is defined to measure the  $T_{READY}$  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, ~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/RCC2** register is switched to use the PLL.

### 6.1.5 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.

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

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

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

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

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

The Cortex-M3 processor core and the memory subsystem are not clocked. Peripherals are clocked that are enabled in the **DCGCn** register when auto-clock gating is enabled (see the **RCC** register) or the **RCGCn** register when auto-clock gating is disabled. The system clock source is the main oscillator by default or the internal oscillator specified in the **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 and override the SYSDIV field of the active **RCC/RCC2** register to be /16 or /64, respectively. When the Deep-Sleep exit event occurs, hardware brings the system clock back to the source and frequency it had at the onset of Deep-Sleep mode before enabling the clocks that had been stopped during the Deep-Sleep duration.

- **Hibernate Mode.** In this mode, the power supplies are turned off to the main part of the device and only the Hibernation module's circuitry is active. An external wake event or RTC event is required to bring the device back to Run mode. The Cortex-M3 processor and peripherals outside

of the Hibernation module see a normal "power on" sequence and the processor starts running code. It can determine that it has been restarted from Hibernate mode by inspecting the Hibernation module registers.

## 6.2 Initialization and Configuration

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

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

## 6.3 Register Map

Table 6-1 on page 67 lists the System Control registers, grouped by function. The offset listed is a hexadecimal increment to the register's address, relative to the System Control base address of 0x400F.E000.

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

**Table 6-1. System Control Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	DID0	RO	-	Device Identification 0	69
0x004	DID1	RO	-	Device Identification 1	85
0x008	DC0	RO	0x00FF.007F	Device Capabilities 0	87
0x010	DC1	RO	0x0111.32FF	Device Capabilities 1	88
0x014	DC2	RO	0x010F.1313	Device Capabilities 2	90
0x018	DC3	RO	0x030F.81FF	Device Capabilities 3	92
0x01C	DC4	RO	0x5100.007F	Device Capabilities 4	94
0x030	PBORCTL	R/W	0x0000.7FFD	Brown-Out Reset Control	71
0x034	LDOCTL	R/W	0x0000.0000	LDO Power Control	72

Offset	Name	Type	Reset	Description	See page
0x040	SRCR0	R/W	0x00000000	Software Reset Control 0	117
0x044	SRCR1	R/W	0x00000000	Software Reset Control 1	118
0x048	SRCR2	R/W	0x00000000	Software Reset Control 2	120
0x050	RIS	RO	0x0000.0000	Raw Interrupt Status	73
0x054	IMC	R/W	0x0000.0000	Interrupt Mask Control	74
0x058	MISC	R/W1C	0x0000.0000	Masked Interrupt Status and Clear	75
0x05C	RESC	R/W	-	Reset Cause	76
0x060	RCC	R/W	0x07AE.3AD1	Run-Mode Clock Configuration	77
0x064	PLLCFG	RO	-	XTAL to PLL Translation	81
0x070	RCC2	R/W	0x0780.2800	Run-Mode Clock Configuration 2	82
0x100	RCGC0	R/W	0x00000040	Run Mode Clock Gating Control Register 0	96
0x104	RCGC1	R/W	0x00000000	Run Mode Clock Gating Control Register 1	102
0x108	RCGC2	R/W	0x00000000	Run Mode Clock Gating Control Register 2	111
0x110	SCGC0	R/W	0x00000040	Sleep Mode Clock Gating Control Register 0	98
0x114	SCGC1	R/W	0x00000000	Sleep Mode Clock Gating Control Register 1	105
0x118	SCGC2	R/W	0x00000000	Sleep Mode Clock Gating Control Register 2	113
0x120	DCGC0	R/W	0x00000040	Deep Sleep Mode Clock Gating Control Register 0	100
0x124	DCGC1	R/W	0x00000000	Deep Sleep Mode Clock Gating Control Register 1	108
0x128	DCGC2	R/W	0x00000000	Deep Sleep Mode Clock Gating Control Register 2	115
0x144	DSLPCLKCFG	R/W	0x0780.0000	Deep Sleep Clock Configuration	84

## 6.4 Register Descriptions

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

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

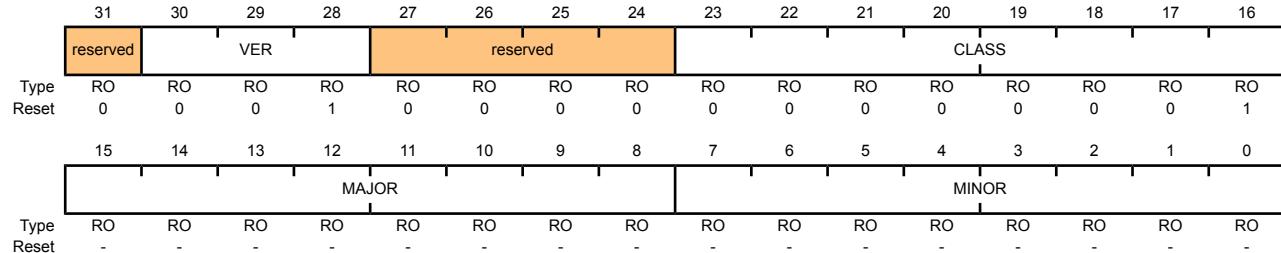
This register identifies the version of the device.

### Device Identification 0 (DID0)

Base 0x400F.E000

Offset 0x000

Type RO, reset -



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

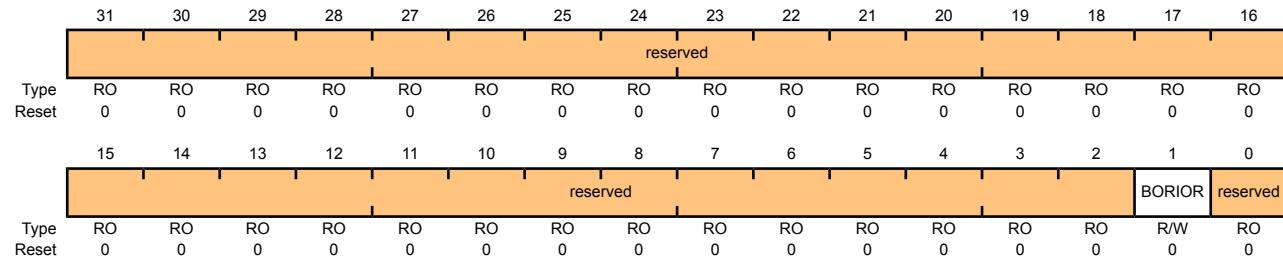
Bit/Field	Name	Type	Reset	Description								
15:8	MAJOR	RO	-	<p>Major Revision</p> <p>This field specifies the major revision number of the device. The major revision reflects changes to base layers of the design. The major revision number is indicated in the part number as a letter (A for first revision,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

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

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

### Brown-Out Reset Control (PBORCTL)

Base 0x400F.E000  
Offset 0x030  
Type R/W, reset 0x0000.7FFD



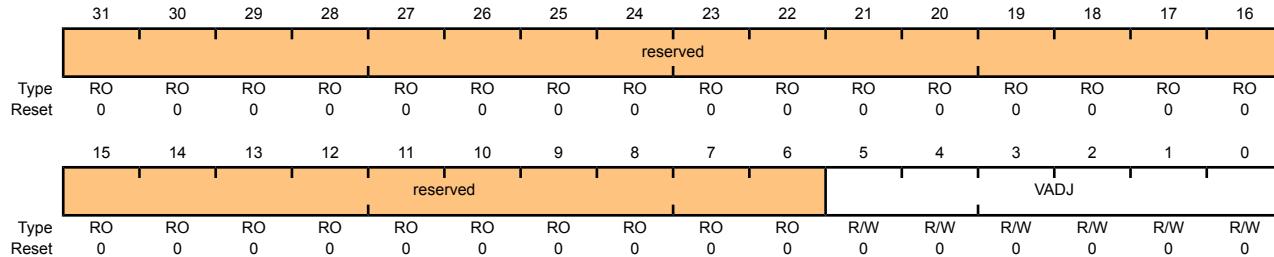
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BORIOR	R/W	0	BOR Interrupt or Reset  This bit controls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

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

The 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	LDO Output Voltage  This field sets the on-chip output voltage. The programming values for the VADJ field are provided below.

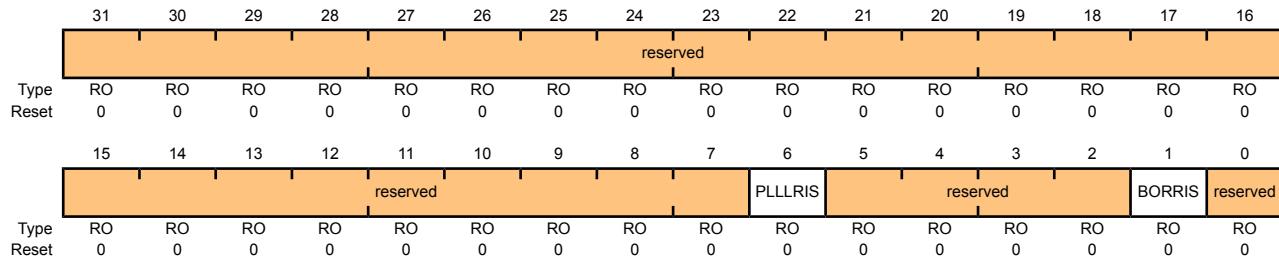
Value	$V_{OUT}$ (V)
0x00	2.50
0x01	2.45
0x02	2.40
0x03	2.35
0x04	2.30
0x05	2.25
0x06-0x3F	Reserved
0x1B	2.75
0x1C	2.70
0x1D	2.65
0x1E	2.60
0x1F	2.55

## 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



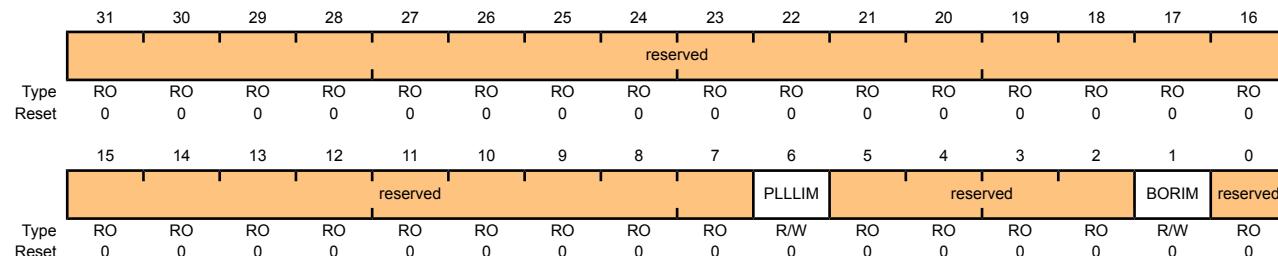
Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PLLRIS	RO	0	PLL Lock Raw Interrupt Status This bit is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

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

Central location for system control interrupt masks.

### Interrupt Mask Control (IMC)

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



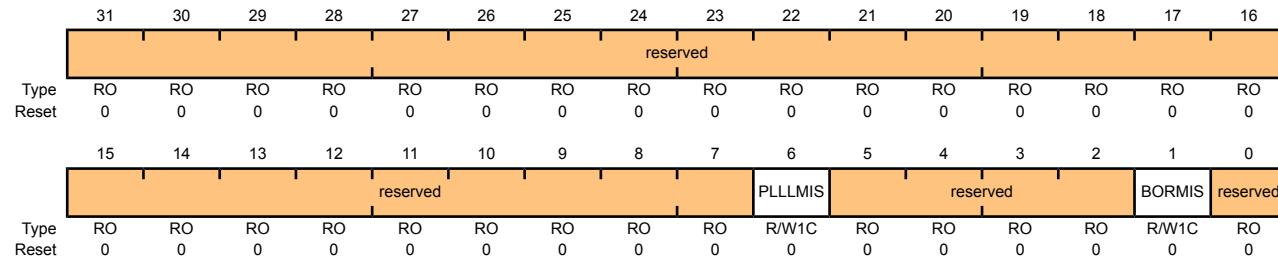
Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PLLLIM	R/W	0	PLL Lock Interrupt Mask  This bit specifies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

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

Central location for system control result of RIS AND IMC to generate an interrupt to the controller. All of the bits are R/W1C and this action also clears the corresponding raw interrupt bit in the **RIS** register (see page 73).

### 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.
5:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BORMIS	R/W1C	0	BOR Masked Interrupt Status The BORMIS is simply the BORRIS ANDed with the mask value, BORIM.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

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

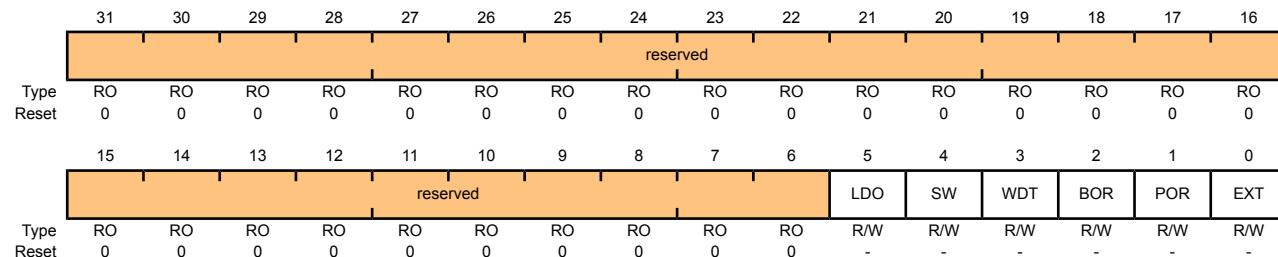
This register is set with the reset cause after reset. The bits in this register are sticky and maintain their state across multiple reset sequences, except when an external reset is the cause, and then all the other bits in the **RESC** register are cleared.

### Reset Cause (RESC)

Base 0x400F.E000

Offset 0x05C

Type R/W, reset -



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 0x07AE.3AD1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved				ACG	SYSDIV				USESYSDIV	reserved	USEPWMDIV	PWMDIV			reserved
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	reserved	BYPASS	reserved	XTAL				OSCSRC		reserved	IOSCDIS	MOSCDIS	
Type	RO	RO	R/W	RO	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO	R/W	R/W
Reset	0	0	1	1	1	0	1	0	1	1	0	1	0	0	0	1

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

Bit/Field	Name	Type	Reset	Description																																																			
26:23	SYSDIV	R/W	0xF	<p>System Clock Divisor</p> <p>Specifies which divisor is used to generate the system clock from the PLL output.</p> <p>The PLL VCO frequency is 400 MHz.</p> <table> <thead> <tr> <th>Value</th> <th>Divisor (BYPASS=1)</th> <th>Frequency (BYPASS=0)</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>reserved</td> <td>reserved</td> </tr> <tr> <td>0x1</td> <td>/2</td> <td>reserved</td> </tr> <tr> <td>0x2</td> <td>/3</td> <td>reserved</td> </tr> <tr> <td>0x3</td> <td>/4</td> <td>50 MHz</td> </tr> <tr> <td>0x4</td> <td>/5</td> <td>40 MHz</td> </tr> <tr> <td>0x5</td> <td>/6</td> <td>33.33 MHz</td> </tr> <tr> <td>0x6</td> <td>/7</td> <td>28.57 MHz</td> </tr> <tr> <td>0x7</td> <td>/8</td> <td>25 MHz</td> </tr> <tr> <td>0x8</td> <td>/9</td> <td>22.22 MHz</td> </tr> <tr> <td>0x9</td> <td>/10</td> <td>20 MHz</td> </tr> <tr> <td>0xA</td> <td>/11</td> <td>18.18 MHz</td> </tr> <tr> <td>0xB</td> <td>/12</td> <td>16.67 MHz</td> </tr> <tr> <td>0xC</td> <td>/13</td> <td>15.38 MHz</td> </tr> <tr> <td>0xD</td> <td>/14</td> <td>14.29 MHz</td> </tr> <tr> <td>0xE</td> <td>/15</td> <td>13.33 MHz</td> </tr> <tr> <td>0xF</td> <td>/16</td> <td>12.5 MHz (default)</td> </tr> </tbody> </table> <p>When reading the <b>Run-Mode Clock Configuration (RCC)</b> register (see page 77), the SYSDIV value is MINSYSDIV if a lower divider was requested and the PLL is being used. This lower value is allowed to divide a non-PLL source.</p>	Value	Divisor (BYPASS=1)	Frequency (BYPASS=0)	0x0	reserved	reserved	0x1	/2	reserved	0x2	/3	reserved	0x3	/4	50 MHz	0x4	/5	40 MHz	0x5	/6	33.33 MHz	0x6	/7	28.57 MHz	0x7	/8	25 MHz	0x8	/9	22.22 MHz	0x9	/10	20 MHz	0xA	/11	18.18 MHz	0xB	/12	16.67 MHz	0xC	/13	15.38 MHz	0xD	/14	14.29 MHz	0xE	/15	13.33 MHz	0xF	/16	12.5 MHz (default)
Value	Divisor (BYPASS=1)	Frequency (BYPASS=0)																																																					
0x0	reserved	reserved																																																					
0x1	/2	reserved																																																					
0x2	/3	reserved																																																					
0x3	/4	50 MHz																																																					
0x4	/5	40 MHz																																																					
0x5	/6	33.33 MHz																																																					
0x6	/7	28.57 MHz																																																					
0x7	/8	25 MHz																																																					
0x8	/9	22.22 MHz																																																					
0x9	/10	20 MHz																																																					
0xA	/11	18.18 MHz																																																					
0xB	/12	16.67 MHz																																																					
0xC	/13	15.38 MHz																																																					
0xD	/14	14.29 MHz																																																					
0xE	/15	13.33 MHz																																																					
0xF	/16	12.5 MHz (default)																																																					
22	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>																																																			
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	<p>Enable PWM Clock Divider</p> <p>Use the PWM clock divider as the source for the PWM clock.</p>																																																			

Bit/Field	Name	Type	Reset	Description																		
19:17	PWMDIV	R/W	0x7	<p>PWM Unit Clock Divisor</p> <p>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.</p> <table> <thead> <tr> <th>Value</th> <th>Divisor</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>/2</td> </tr> <tr> <td>0x1</td> <td>/4</td> </tr> <tr> <td>0x2</td> <td>/8</td> </tr> <tr> <td>0x3</td> <td>/16</td> </tr> <tr> <td>0x4</td> <td>/32</td> </tr> <tr> <td>0x5</td> <td>/64</td> </tr> <tr> <td>0x6</td> <td>/64</td> </tr> <tr> <td>0x7</td> <td>/64 (default)</td> </tr> </tbody> </table>	Value	Divisor	0x0	/2	0x1	/4	0x2	/8	0x3	/16	0x4	/32	0x5	/64	0x6	/64	0x7	/64 (default)
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	<p>PLL Power Down</p> <p>This bit connects to the PLL PWRDN input. The reset value of 1 powers down the PLL.</p>																		
12	reserved	RO	1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.																		
11	BYPASS	R/W	1	<p>PLL Bypass</p> <p>Chooses whether the system clock is derived from the PLL output or the OSC source. If set, the clock that drives the system is the OSC source. Otherwise, the clock that drives the system is the PLL output clock divided by the system divider.</p> <p><b>Note:</b> The ADC must be clocked from the PLL or directly from a 14-MHz to 18-MHz clock source to operate properly. While the ADC works in a 14-18 MHz range, to maintain a 1 M sample/second rate, the ADC must be provided a 16-MHz clock source.</p>																		
10	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																																																			
9:6	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>3.579545 MHz</td><td></td></tr> <tr><td>0x5</td><td>3.6864 MHz</td><td></td></tr> <tr><td>0x6</td><td>4 MHz</td><td></td></tr> <tr><td>0x7</td><td>4.096 MHz</td><td></td></tr> <tr><td>0x8</td><td>4.9152 MHz</td><td></td></tr> <tr><td>0x9</td><td>5 MHz</td><td></td></tr> <tr><td>0xA</td><td>5.12 MHz</td><td></td></tr> <tr><td>0xB</td><td>6 MHz (reset value)</td><td></td></tr> <tr><td>0xC</td><td>6.144 MHz</td><td></td></tr> <tr><td>0xD</td><td>7.3728 MHz</td><td></td></tr> <tr><td>0xE</td><td>8 MHz</td><td></td></tr> <tr><td>0xF</td><td>8.192 MHz</td><td></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	0x1	<p>Oscillator Source</p> <p>Picks among the four input sources 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>Main oscillator (default)</td></tr> <tr><td>0x1</td><td>Internal oscillator (default)</td></tr> <tr><td>0x2</td><td>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	Main oscillator (default)	0x1	Internal oscillator (default)	0x2	Internal oscillator / 4 (this is necessary if used as input to PLL)	0x3	reserved																																									
Value	Input Source																																																						
0x0	Main oscillator (default)																																																						
0x1	Internal oscillator (default)																																																						
0x2	Internal oscillator / 4 (this is necessary if used as input to PLL)																																																						
0x3	reserved																																																						
3:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.																																																			
1	IOSCDIS	R/W	0	<p>Internal Oscillator Disable</p> <p>0: Internal oscillator (IOSC) is enabled. 1: Internal oscillator is disabled.</p>																																																			
0	MOSCDIS	R/W	1	<p>Main Oscillator Disable</p> <p>0: Main oscillator is enabled. 1: Main oscillator is disabled (default).</p>																																																			

## 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 77).

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

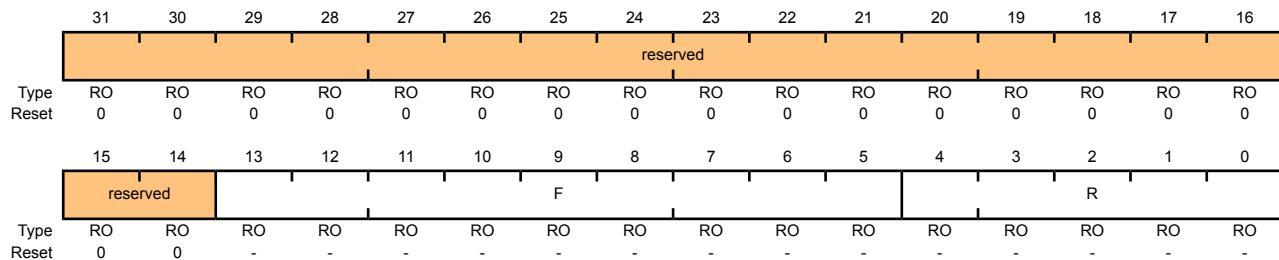
$$\text{PLLFreq} = \text{OSCFreq} * F / (R + 1)$$

### XTAL to PLL Translation (PLLCFG)

Base 0x400F.E000

Offset 0x064

Type RO, reset -



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

## Register 10: Run-Mode Clock Configuration 2 (RCC2), offset 0x070

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

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

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

Base 0x400F.E000  
Offset 0x070  
Type R/W, reset 0x0780.2800

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

Bit/Field	Name	Type	Reset	Description
31	USERCC2	R/W	0	Use RCC2  When set, overrides the <b>RCC</b> register fields.
30:29	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28:23	SYSDIV2	R/W	0x0F	System Clock Divisor  Specifies which divisor is used to generate the system clock from the PLL output.  The PLL VCO frequency is 400 MHz.  This field is wider than the <b>RCC</b> register <b>SYSDIV</b> field in order to provide additional divisor values. This permits the system clock to be run at much lower frequencies during Deep Sleep mode. For example, where the <b>RCC</b> register <b>SYSDIV</b> encoding of 1111 provides /16, the <b>RCC2</b> register <b>SYSDIV2</b> encoding of 111111 provides /64.
22:14	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	PWRDN2	R/W	1	Power-Down PLL  When set, powers down the PLL.
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	BYPASS2	R/W	1	Bypass PLL  When set, bypasses the PLL for the clock source.

---

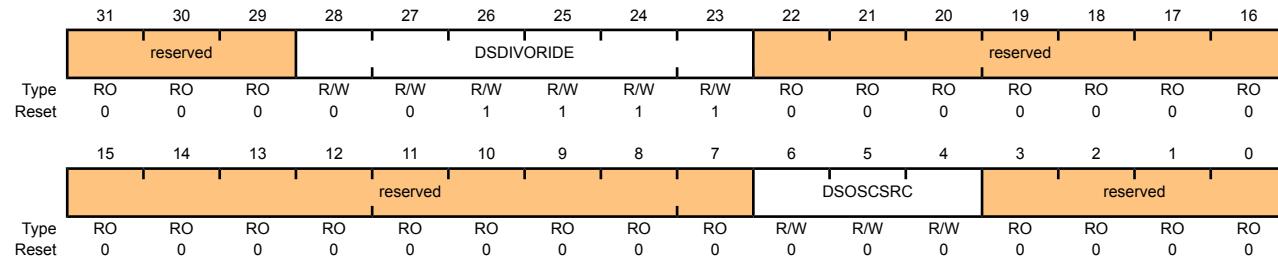
Bit/Field	Name	Type	Reset	Description												
10:7	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.												
6:4	OSCSRC2	R/W	0x0	System Clock Source <table border="0"> <tr> <th style="text-align: center;">Value</th> <th style="text-align: center;">Description</th> </tr> <tr> <td style="text-align: center;">0x0</td> <td>Main oscillator (MOSC)</td> </tr> <tr> <td style="text-align: center;">0x1</td> <td>Internal oscillator (IOSC)</td> </tr> <tr> <td style="text-align: center;">0x2</td> <td>Internal oscillator / 4</td> </tr> <tr> <td style="text-align: center;">0x3</td> <td>30 kHz internal oscillator</td> </tr> <tr> <td style="text-align: center;">0x7</td> <td>32 kHz external oscillator</td> </tr> </table>	Value	Description	0x0	Main oscillator (MOSC)	0x1	Internal oscillator (IOSC)	0x2	Internal oscillator / 4	0x3	30 kHz internal oscillator	0x7	32 kHz external oscillator
Value	Description															
0x0	Main oscillator (MOSC)															
0x1	Internal oscillator (IOSC)															
0x2	Internal oscillator / 4															
0x3	30 kHz internal oscillator															
0x7	32 kHz external oscillator															
3:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.												

## Register 11: Deep Sleep Clock Configuration (DSLPCLKCFG), offset 0x144

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

### Deep Sleep Clock Configuration (DSLPCLKCFG)

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



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

## Register 12: Device Identification 1 (DID1), offset 0x004

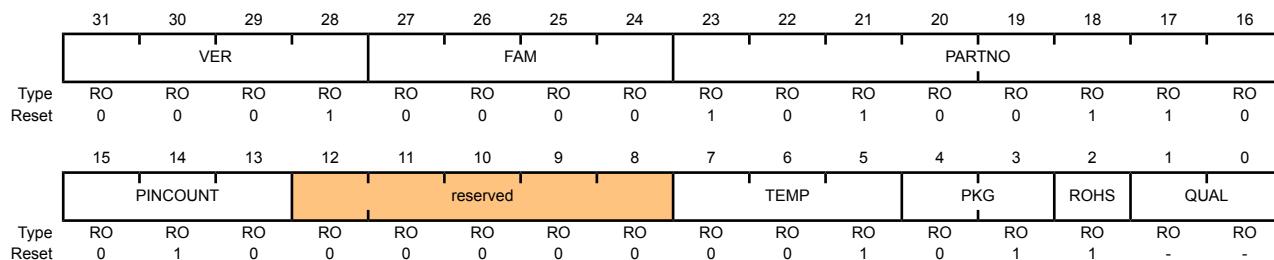
This register identifies the device family, part number, temperature range, pin count, and package type.

### Device Identification 1 (DID1)

Base 0x400F.E000

Offset 0x004

Type RO, reset -



Bit/Field      Name      Type      Reset      Description

31:28      VER      RO      0x1      DID1 Version

This field defines the **DID1** register format version. The version number is numeric. The value of the **VER** field is encoded as follows (all other encodings are reserved):

Value      Description

0x1      First revision of the **DID1** register format, indicating a Stellaris Fury-class device.

27:24      FAM      RO      0x0      Family

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):

Value      Description

0x0      Stellaris family of microcontrollers, that is, all devices with external part numbers starting with LM3S.

23:16      PARTNO      RO      0xA6      Part Number

This field provides the part number of the device within the family. The value is encoded as follows (all other encodings are reserved):

Value      Description

0xA6      LM3S8962

15:13      PINCOUNT      RO      0x2      Package Pin Count

This field specifies the number of pins on the device package. The value is encoded as follows (all other encodings are reserved):

Value      Description

0x2      100-pin package

Bit/Field	Name	Type	Reset	Description								
12:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								
7:5	TEMP	RO	0x1	<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>0x1</td><td>Industrial temperature range (-40°C to 85°C)</td></tr> </tbody> </table>	Value	Description	0x1	Industrial temperature range (-40°C to 85°C)				
Value	Description											
0x1	Industrial temperature range (-40°C to 85°C)											
4:3	PKG	RO	0x1	<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>0x1</td><td>LQFP package</td></tr> </tbody> </table>	Value	Description	0x1	LQFP package				
Value	Description											
0x1	LQFP 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 13: Device Capabilities 0 (DC0), offset 0x008

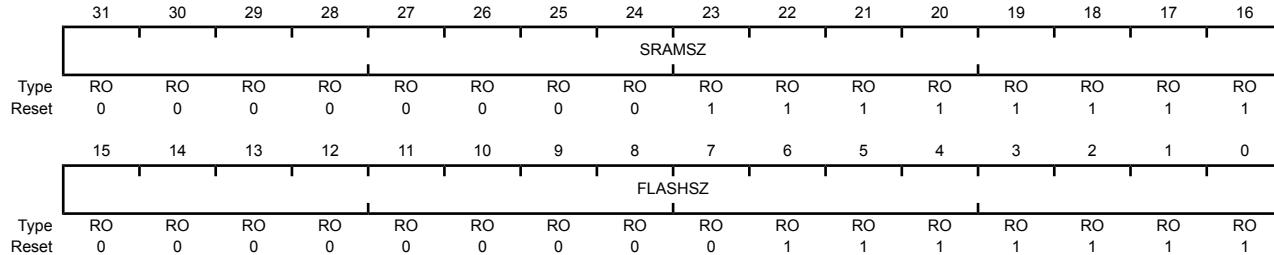
This register is predefined by the part and can be used to verify features.

### Device Capabilities 0 (DC0)

Base 0x400F.E000

Offset 0x008

Type RO, reset 0x00FF.007F



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

31:16	SRAMSZ	RO	0x00FF	SRAM Size Indicates the size of the on-chip SRAM memory. Value Description 0x00FF 64 KB of SRAM
-------	--------	----	--------	--

15:0	FLASHSZ	RO	0x007F	Flash Size Indicates the size of the on-chip flash memory. Value Description 0x007F 256 KB of Flash
------	---------	----	--------	--

## Register 14: 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: CANs, PWM, ADC, Watchdog timer, Hibernation module, 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 **DCGC0** clock control registers and the **SRCR0** software reset control register.

### Device Capabilities 1 (DC1)

Base 0x400F.E000

Offset 0x010

Type RO, reset 0x0111.32FF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	RO	CAN0	reserved	reserved	PWM	reserved	ADC									
Reset	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RO	MPU	HIB	TEMPSNS	PLL	WDT	SWO	SWD	JTAG							
Reset	0	0	1	1	0	0	1	0	1	1	1	1	1	1	1	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	CAN0	RO	1	CAN Module 0 Present When set, indicates that CAN unit 0 is present.
23:21	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.

Bit/Field	Name	Type	Reset	Description				
11:8	MAXADCSPD	RO	0x2	<p>Max ADC Speed</p> <p>Indicates the maximum rate at which the ADC samples data.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x2</td><td>500K samples/second</td></tr> </tbody> </table>	Value	Description	0x2	500K samples/second
Value	Description							
0x2	500K samples/second							
7	MPU	RO	1	<p>MPU Present</p> <p>When set, indicates that the Cortex-M3 Memory Protection Unit (MPU) module is present. See the ARM Cortex-M3 Technical Reference Manual for details on the MPU.</p>				
6	HIB	RO	1	<p>Hibernation Module Present</p> <p>When set, indicates that the Hibernation module is present.</p>				
5	TEMPSNS	RO	1	<p>Temp Sensor Present</p> <p>When set, indicates that the on-chip temperature sensor is present.</p>				
4	PLL	RO	1	<p>PLL Present</p> <p>When set, indicates that the on-chip Phase Locked Loop (PLL) is present.</p>				
3	WDT	RO	1	<p>Watchdog Timer Present</p> <p>When set, indicates that a watchdog timer is present.</p>				
2	SWO	RO	1	<p>SWO Trace Port Present</p> <p>When set, indicates that the Serial Wire Output (SWO) trace port is present.</p>				
1	SWD	RO	1	<p>SWD Present</p> <p>When set, indicates that the Serial Wire Debugger (SWD) is present.</p>				
0	JTAG	RO	1	<p>JTAG Present</p> <p>When set, indicates that the JTAG debugger interface is present.</p>				

## Register 15: 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 **DGCG1** clock control registers and the **SRCR1** software reset control register.

### Device Capabilities 2 (DC2)

Base 0x400F.E000

Offset 0x014

Type RO, reset 0x010F.1313

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	RO	RO	RO	RO	RO	RO	RO	COMPO	reserved	reserved	reserved	TIMER3	TIMER2	TIMER1	TIMER0	
Reset	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	reserved	I2C0	reserved	QEI1	QEI0	reserved	SSI0	reserved	UART1	UART0	reserved	UART1	UART0	UART1	UART0	UART0
Reset	0	0	0	1	0	0	1	0	0	0	0	1	0	1	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.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	RO	1	Timer 3 Present When set, indicates that General-Purpose Timer module 3 is present.
18	TIMER2	RO	1	Timer 2 Present When set, indicates that General-Purpose Timer module 2 is present.
17	TIMER1	RO	1	Timer 1 Present When set, indicates that General-Purpose Timer module 1 is present.
16	TIMER0	RO	1	Timer 0 Present When set, indicates that General-Purpose Timer module 0 is present.
15:13	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.

Bit/Field	Name	Type	Reset	Description
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	QEI1	RO	1	QEI1 Present When set, indicates that QEI module 1 is present.
8	QEI0	RO	1	QEI0 Present When set, indicates that QEI module 0 is present.
7:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SSI0	RO	1	SSI0 Present When set, indicates that SSI module 0 is present.
3:2	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 16: 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 0x030F.81FF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved						CCP1	CCP0	reserved				ADC3	ADC2	ADC1	ADC0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	1	1	0	0	0	0	1	1	1	1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PWMFAULT	reserved						C0O	C0PLUS	C0MINUS	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:26	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	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.
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	PWMFAULT	RO	1	PWM Fault Pin Present When set, indicates that the PWM Fault pin is present.
14:9	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.

---

Bit/Field	Name	Type	Reset	Description
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 17: 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 the following family features in the specific device: Ethernet MAC and PHY, GPIOs, and CCP I/Os. 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 0x5100.007F

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	reserved	EPHY0	reserved	EMAC0	reserved	reserved	E1588	reserved								
Reset	RO 0	RO 1	RO 0	RO 1	RO 0	RO 0	RO 0	RO 1	RO 0							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	reserved								GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA	
Reset	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 1						

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	EPHY0	RO	1	Ethernet PHY0 Present  When set, indicates that Ethernet PHY module 0 is present.
29	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.
28	EMAC0	RO	1	Ethernet MAC0 Present  When set, indicates that Ethernet MAC module 0 is present.
27:25	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	E1588	RO	1	1588 Capable  When set, indicates that that EMAC0 is 1588-capable.
23:7	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	GPIOG	RO	1	GPIO Port G Present  When set, indicates that GPIO Port G is present.
5	GPIOF	RO	1	GPIO Port F Present  When set, indicates that GPIO Port F is present.
4	GPIOE	RO	1	GPIO Port E Present  When set, indicates that GPIO Port E is present.

---

Bit/Field	Name	Type	Reset	Description
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 18: Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100

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

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

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved							CAN0	reserved			PWM	reserved			ADC
Type	RO	RO	RO	RO	RO	RO	RO	R/W	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	HIB	reserved		WDT	reserved		
Type	RO	RO	RO	RO	R/W	R/W	R/W	R/W	RO	R/W	RO	RO	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:25	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	CAN0	R/W	0	CAN0 Clock Gating Control  This bit controls the clock gating for CAN unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.
23:21	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.

Bit/Field	Name	Type	Reset	Description								
15:12	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.								
11:8	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	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								
6	HIB	R/W	0	<p>HIB Clock Gating Control</p> <p>This bit controls the clock gating for the Hibernation module. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.</p>								
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								
3	WDT	R/W	0	<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 19: Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110

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

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

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	RO	RO	RO	RO	RO	RO	RO	R/W	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
Type	RO	RO	RO	RO	R/W	R/W	R/W	R/W	RO	R/W	RO	RO	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<hr/>																
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	CAN0		R/W	0	CAN0 Clock Gating Control  This bit controls the clock gating for CAN unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.											
23:21	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.											

Bit/Field	Name	Type	Reset	Description								
15:12	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.								
11:8	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	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								
6	HIB	R/W	0	<p>HIB Clock Gating Control</p> <p>This bit controls the clock gating for the Hibernation module. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.</p>								
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								
3	WDT	R/W	0	<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: 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
Type	RO	RO	RO	RO	RO	RO	R/W	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
Type	RO	RO	RO	RO	R/W	R/W	R/W	R/W	RO	R/W	RO	RO	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<hr/>																
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	CAN0	R/W	0	CAN0 Clock Gating Control  This bit controls the clock gating for CAN unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.												
23:21	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.												

Bit/Field	Name	Type	Reset	Description								
15:12	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.								
11:8	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	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								
6	HIB	R/W	0	<p>HIB Clock Gating Control</p> <p>This bit controls the clock gating for the Hibernation module. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.</p>								
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								
3	WDT	R/W	0	<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: 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	R/W	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
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RO	RO	RO	R/W	RO	RO	R/W	RO	RO	RO	RO	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
31:25	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.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	R/W	0	Timer 3 Clock Gating Control  This bit controls the clock gating for General-Purpose Timer module 3. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
18	TIMER2	R/W	0	Timer 2 Clock Gating Control  This bit controls the clock gating for General-Purpose Timer module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

Bit/Field	Name	Type	Reset	Description
17	TIMER1	R/W	0	Timer 1 Clock Gating Control  This bit controls the clock gating for General-Purpose Timer module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control  This bit controls the clock gating for General-Purpose Timer module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
15:13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Clock Gating Control  This bit controls the clock gating for I2C module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
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	QEI1	R/W	0	QEI1 Clock Gating Control  This bit controls the clock gating for QEI 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.
8	QEI0	R/W	0	QEI0 Clock Gating Control  This bit controls the clock gating for QEI 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.
7:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SSI0	R/W	0	SSI0 Clock Gating Control  This bit controls the clock gating for SSI module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
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	R/W	0	UART1 Clock Gating Control  This bit controls the clock gating for UART module 1. If set, the unit receives a clock and functions. Otherwise, the unit is 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	UART0	R/W	0	<p>UART0 Clock Gating Control</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 22: 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 0x400F.E000  
Offset 0x114  
Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	RO	RO	RO	RO	RO	RO	R/W	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RO	RO	RO	R/W	RO	RO	R/W	R/W	RO	RO	RO	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
31:25	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.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	R/W	0	Timer 3 Clock Gating Control  This bit controls the clock gating for General-Purpose Timer module 3. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
18	TIMER2	R/W	0	Timer 2 Clock Gating Control  This bit controls the clock gating for General-Purpose Timer module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

Bit/Field	Name	Type	Reset	Description
17	TIMER1	R/W	0	Timer 1 Clock Gating Control  This bit controls the clock gating for General-Purpose Timer module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control  This bit controls the clock gating for General-Purpose Timer module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
15:13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Clock Gating Control  This bit controls the clock gating for I2C module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
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	QEI1	R/W	0	QEI1 Clock Gating Control  This bit controls the clock gating for QEI 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.
8	QEI0	R/W	0	QEI0 Clock Gating Control  This bit controls the clock gating for QEI 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.
7:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SSI0	R/W	0	SSI0 Clock Gating Control  This bit controls the clock gating for SSI module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
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	R/W	0	UART1 Clock Gating Control  This bit controls the clock gating for UART module 1. If set, the unit receives a clock and functions. Otherwise, the unit is 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	UART0	R/W	0	<p>UART0 Clock Gating Control</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: 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 0x400F.E000

Offset 0x124

Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	RO	RO	RO	RO	RO	RO	RO	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
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RO	RO	RO	R/W	RO	RO	R/W	R/W	RO	RO	RO	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
	reserved				I2C0	reserved		QEI1	QEI0	reserved		SSI0	reserved		UART1	UART0

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.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	R/W	0	Timer 3 Clock Gating Control  This bit controls the clock gating for General-Purpose Timer module 3. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
18	TIMER2	R/W	0	Timer 2 Clock Gating Control  This bit controls the clock gating for General-Purpose Timer module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

Bit/Field	Name	Type	Reset	Description
17	TIMER1	R/W	0	Timer 1 Clock Gating Control  This bit controls the clock gating for General-Purpose Timer module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control  This bit controls the clock gating for General-Purpose Timer module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
15:13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Clock Gating Control  This bit controls the clock gating for I2C module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
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	QEI1	R/W	0	QEI1 Clock Gating Control  This bit controls the clock gating for QEI 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.
8	QEI0	R/W	0	QEI0 Clock Gating Control  This bit controls the clock gating for QEI 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.
7:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SSI0	R/W	0	SSI0 Clock Gating Control  This bit controls the clock gating for SSI module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
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	R/W	0	UART1 Clock Gating Control  This bit controls the clock gating for UART module 1. If set, the unit receives a clock and functions. Otherwise, the unit is 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	UART0	R/W	0	<p>UART0 Clock Gating Control</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: 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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	reserved	EPHY0	reserved	EMAC0	reserved											
Type	RO	R/W	RO	R/W	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	reserved												GPIOG	GPIOF	GPIOE	GPIOD
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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	EPHY0	R/W	0	PHY0 Clock Gating Control  This bit controls the clock gating for Ethernet PHY unit 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.
29	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.
28	EMAC0	R/W	0	MAC0 Clock Gating Control  This bit controls the clock gating for Ethernet MAC unit 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.
27:7	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	GPIOG	R/W	0	Port G Clock Gating Control  This bit controls the clock gating for Port G. 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
5	GPIOF	R/W	0	Port F Clock Gating Control  This bit controls the clock gating for Port F. 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.
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 25: 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
Type	reserved	EPHY0	reserved	EMAC0	reserved											
Reset	RO 0	R/W 0	RO 0	R/W 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	reserved												GPIOG	GPIOF	GPIOE	GPIOD
Reset	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	R/W 0						

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPHY0	R/W	0	PHY0 Clock Gating Control  This bit controls the clock gating for Ethernet PHY unit 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.
29	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.
28	EMAC0	R/W	0	MAC0 Clock Gating Control  This bit controls the clock gating for Ethernet MAC unit 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.
27:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
6	GPIOG	R/W	0	Port G Clock Gating Control  This bit controls the clock gating for Port G. 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.
5	GPIOF	R/W	0	Port F Clock Gating Control  This bit controls the clock gating for Port F. 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.
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: Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128

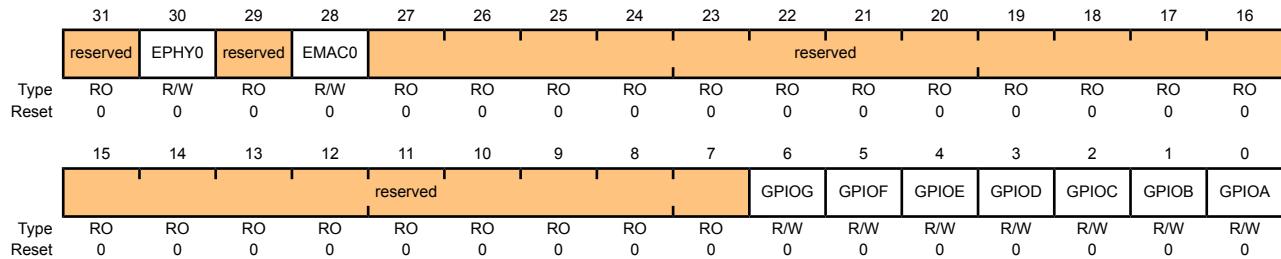
This register controls the clock gating logic. Each bit controls a clock enable for a given interface, function, or unit. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled (saving power). If the unit is unclocked, reads or writes to the unit will generate 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



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	EPHY0	R/W	0	PHY0 Clock Gating Control This bit controls the clock gating for Ethernet PHY unit 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.
29	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.
28	EMAC0	R/W	0	MAC0 Clock Gating Control This bit controls the clock gating for Ethernet MAC unit 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.
27:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

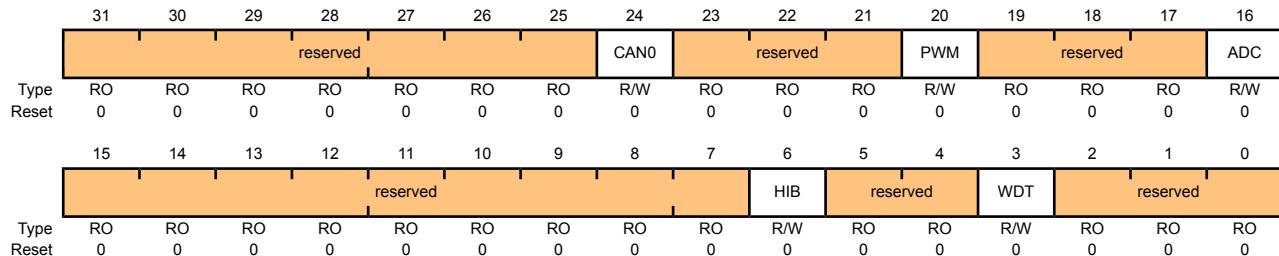
Bit/Field	Name	Type	Reset	Description
6	GPIOG	R/W	0	Port G Clock Gating Control  This bit controls the clock gating for Port G. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
5	GPIOF	R/W	0	Port F Clock Gating Control  This bit controls the clock gating for Port F. 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.
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 27: 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



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	CAN0	R/W	0	CAN0 Reset Control Reset control for CAN unit 0.
23:21	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.
15:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	HIB	R/W	0	HIB Reset Control Reset control for the Hibernation module.
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT	R/W	0	WDT Reset Control Reset control for Watchdog unit.
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 28: Software Reset Control 1 (SRCR1), offset 0x044

Writes to this register are masked by the bits in the **Device Capabilities 2 (DC2)** register.

### Software Reset Control 1 (SRCR1)

Base 0x400F.E000  
Offset 0x044  
Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved																
Type	RO	RO	RO	RO	RO	RO	RO	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
reserved																
Type	RO	RO	RO	R/W	RO	RO	R/W	R/W	RO	RO	RO	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
I2C0																
Type	RO	RO	RO	R/W	RO	RO	R/W	R/W	RO	RO	RO	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
QEI1																
Type	RO	RO	RO	R/W	RO	RO	R/W	R/W	RO	RO	RO	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
QEI0																
Type	RO	RO	RO	R/W	RO	RO	R/W	R/W	RO	RO	RO	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
reserved																
Type	RO	RO	RO	R/W	RO	RO	R/W	R/W	RO	RO	RO	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
SSIO																
Type	RO	RO	RO	R/W	RO	RO	R/W	R/W	RO	RO	RO	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
reserved																
Type	RO	RO	RO	R/W	RO	RO	R/W	R/W	RO	RO	RO	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
UART1																
Type	RO	RO	RO	R/W	RO	RO	R/W	R/W	RO	RO	RO	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
UART0																
Type	RO	RO	RO	R/W	RO	RO	R/W	R/W	RO	RO	RO	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
31:25	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.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	R/W	0	Timer 3 Reset Control Reset control for General-Purpose Timer module 3.
18	TIMER2	R/W	0	Timer 2 Reset Control Reset control for General-Purpose Timer module 2.
17	TIMER1	R/W	0	Timer 1 Reset Control Reset control for General-Purpose Timer module 1.
16	TIMER0	R/W	0	Timer 0 Reset Control Reset control for General-Purpose Timer module 0.
15:13	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.
11:10	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	QEI1	R/W	0	QEI1 Reset Control Reset control for QEI unit 1.

---

Bit/Field	Name	Type	Reset	Description
8	QEI0	R/W	0	QEI0 Reset Control Reset control for QEI unit 0.
7:5	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.
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 29: 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	EPHY0	reserved	EMAC0												
Type	RO	R/W	RO	R/W	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
											GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB
Type	RO	RO	RO	RO	RO	RO	RO	RO	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	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	EPHY0	R/W	0	PHY0 Reset Control Reset control for Ethernet PHY unit 0.
29	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.
28	EMAC0	R/W	0	MAC0 Reset Control Reset control for Ethernet MAC unit 0.
27:7	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	GPIOG	R/W	0	Port G Reset Control Reset control for GPIO Port G.
5	GPIOF	R/W	0	Port F Reset Control Reset control for GPIO Port F.
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.

Bit/Field	Name	Type	Reset	Description
0	GPIOA	R/W	0	Port A Reset Control Reset control for GPIO Port A.

## 7 Hibernation Module

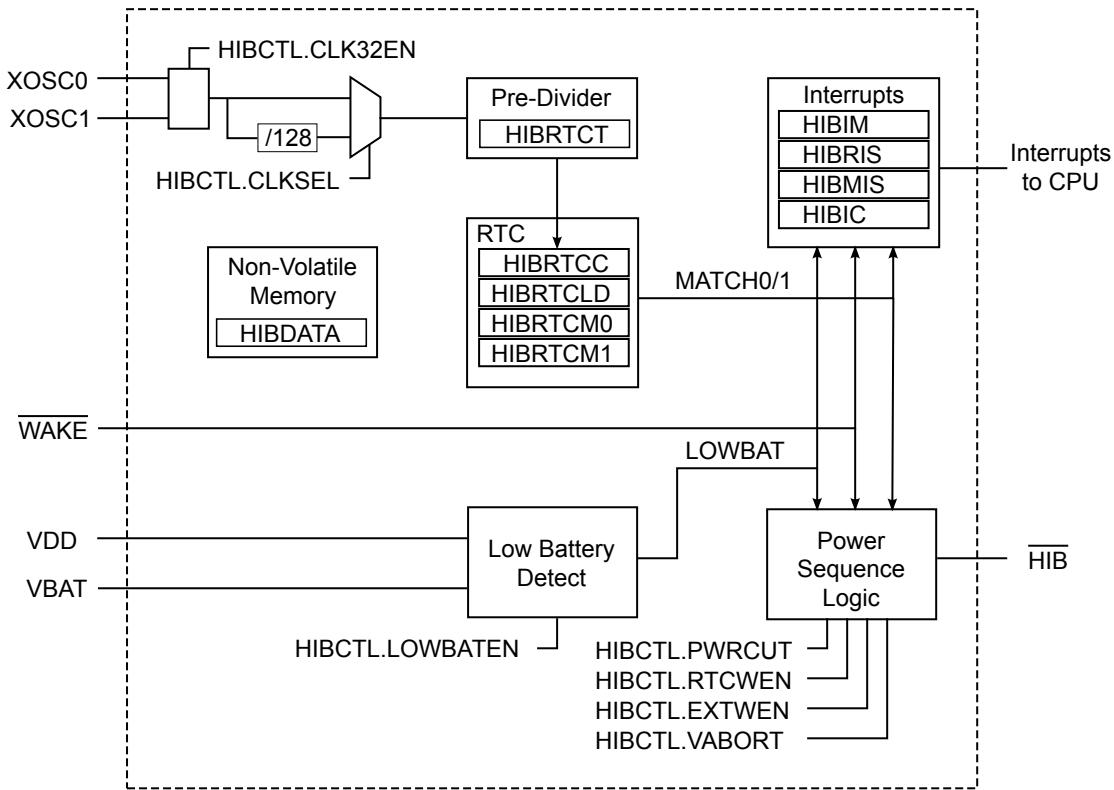
The Hibernation Module manages removal and restoration of power to the rest of the microcontroller to provide a means for reducing power consumption. When the processor and peripherals are idle, power can be completely removed with only the Hibernation Module remaining powered. Power can be restored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

The Hibernation module has the following features:

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

## 7.1 Block Diagram

Figure 7-1. Hibernation Module Block Diagram



## 7.2 Functional Description

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

Power-up from a power cut to code execution is defined as the regulator turn-on time (specified at  $t_{HIB\_TO\_VDD}$  maximum) plus the normal chip POR (see “Hibernation Module” on page 586).

### 7.2.1 Register Access Timing

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

restriction on timing for back-to-back reads from the Hibernation module. Refer to “Register Descriptions” on page 128 for details about which registers are subject to this timing restriction.

### 7.2.2 Clock Source

The Hibernation module must be clocked by an external source, even if the RTC feature will not be used. An external oscillator or crystal can be used for this purpose. To use a crystal, a 4.194304-MHz crystal is connected to the `XOSC0` and `XOSC1` pins. This clock signal is divided by 128 internally to produce the 32.768-kHz clock reference. To use a more precise clock source, a 32.768-kHz oscillator can be connected to the `XOSC0` pin.

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

### 7.2.3 Battery Management

The Hibernation module can be independently powered by a battery or an auxiliary power source. The module can monitor the voltage level of the battery and detect when the voltage becomes too low. When this happens, an interrupt can be generated. The module can also be configured so that it will not go into Hibernate mode if the battery voltage is too low.

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

The Hibernation module can be configured to detect a low battery condition by setting the `LOWBATEN` bit of the **HIBCTL** register. In this configuration, the `LOWBAT` bit of the **HIBRIS** register will be set when the battery level is low. If the `VABORT` bit is also set, then the module is prevented from entering Hibernation mode when a low battery is detected. The module can also be configured to generate an interrupt for the low-battery condition (see “Interrupts and Status” on page 125).

### 7.2.4 Real-Time Clock

The Hibernation module includes a 32-bit counter that increments once per second with a proper clock source and configuration (see “Clock Source” on page 124). The 32.768-kHz clock signal is fed into a predivider register which counts down the 32.768-kHz clock ticks to achieve a once per second clock rate for the RTC. The rate can be adjusted to compensate for inaccuracies in the clock source by using the predivider trim register. This register has a nominal value of `0x7FFF`, and is used for one second out of every 64 seconds to divide the input clock. This allows the software to make fine corrections to the clock rate by adjusting the predivider trim register up or down from `0x7FFF`. The predivider trim should be adjusted up from `0x7FFF` in order to slow down the RTC rate, and down from `0x7FFF` in order to speed up the RTC rate.

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

The RTC must be enabled with the `RTCEN` bit of the **HIBCTL** register. The value of the RTC can be set at any time by writing to the **HIBRTCLD** register. The predivider trim can be adjusted by reading and writing the **HIBRTCT** register. The predivider uses this register once every 64 seconds to adjust

the clock rate. The two match registers can be set by writing to the **HIBRTCM0** and **HIBRTCM1** registers. The RTC can be configured to generate interrupts by using the interrupt registers (see “Interrupts and Status” on page 125).

### 7.2.5 Non-Volatile Memory

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

### 7.2.6 Power Control

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

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

When the Hibernation module wakes, the microcontroller will see a normal power-on reset. It can detect that the power-on was due to a wake from hibernation by examining the raw interrupt status register (see “Interrupts and Status” on page 125) and by looking for state data in the non-volatile memory (see “Non-Volatile Memory” on page 125).

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

### 7.2.7 Interrupts and Status

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

- Assertion of **WAKE** pin
- RTC match
- Low battery detected

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

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

## 7.3 Initialization and Configuration

The Hibernation module can be configured in several different combinations. The following sections show the recommended programming sequence for various scenarios. The examples below assume that a 32.768-kHz oscillator is used, and thus always show bit 2 (CLKSEL) of the **HIBCTL** register set to 1. If a 4.194304-MHz crystal is used instead, then the CLKSEL bit remains cleared. Because the Hibernation module runs at 32 kHz and is asynchronous to the rest of the system, software must allow a delay of  $t_{HIB\_REG\_WRITE}$  after writes to certain registers (see “Register Access Timing” on page 123). The registers that require a delay are denoted with a footnote in Table 7-1 on page 127.

### 7.3.1 Initialization

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

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

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

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

The above is only necessary when the entire system is initialized for the first time. If the processor is powered due to a wake from hibernation, then the Hibernation module has already been powered up and the above steps are not necessary. The software can detect that the Hibernation module and clock are already powered by examining the CLK32EN bit of the **HIBCTL** register.

### 7.3.2 RTC Match Functionality (No Hibernation)

The following steps are needed to use the RTC match functionality of the Hibernation module:

1. Write the required RTC match value to one of the **HIBRTCM<sub>n</sub>** registers at offset 0x004 or 0x008.
2. Write the required RTC load value to the **HIBRTCLD** register at offset 0x00C.
3. Set the required RTC match interrupt mask in the RTCALTO and RTCALT1 bits (bits 1:0) in the **HIBIM** register at offset 0x014.
4. Write 0x0000.0041 to the **HIBCTL** register at offset 0x010 to enable the RTC to begin counting.

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

The following steps are needed to use the RTC match and wake-up functionality of the Hibernation module:

1. Write the required RTC match value to the **HIBRTCM<sub>n</sub>** registers at offset 0x004 or 0x008.
2. Write the required RTC load value to the **HIBRTCLD** register at offset 0x00C.
3. Write any data to be retained during power cut to the **HIBDATA** register at offsets 0x030-0x12C.

4. Set the RTC Match Wake-Up and start the hibernation sequence by writing 0x0000.004F to the **HIBCTL** register at offset 0x010.

#### 7.3.4 External Wake-Up from Hibernation

The following steps are needed to use the Hibernation module with the external **WAKE** pin as the wake-up source for the microcontroller:

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

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

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

### 7.4 Register Map

Table 7-1 on page 127 lists the Hibernation registers. All addresses given are relative to the Hibernation Module base address at 0x400F.C000.

**Note:** **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and require a delay of  $t_{HIB\_REG\_WRITE}$  between write accesses. See “Register Access Timing” on page 123.

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

Offset	Name	Type	Reset	Description	See page
0x000	HIBRTCC	RO	0x0000.0000	Hibernation RTC Counter	129
0x004	HIBRTCM0	R/W	0xFFFF.FFFF	Hibernation RTC Match 0	130
0x008	HIBRTCM1	R/W	0xFFFF.FFFF	Hibernation RTC Match 1	131
0x00C	HIBRTCLD	R/W	0xFFFF.FFFF	Hibernation RTC Load	132
0x010	HIBCTL	R/W	0x0000.0000	Hibernation Control	133
0x014	HIBIM	R/W	0x0000.0000	Hibernation Interrupt Mask	135
0x018	HIBRIS	RO	0x0000.0000	Hibernation Raw Interrupt Status	136
0x01C	HIBMIS	RO	0x0000.0000	Hibernation Masked Interrupt Status	137
0x020	HIBIC	R/W1C	0x0000.0000	Hibernation Interrupt Clear	138
0x024	HIBRTCT	R/W	0x0000.7FFF	Hibernation RTC Trim	139
0x030-0x12C	HIBDATA	R/W	0x0000.0000	Hibernation Data	140

## 7.5 Register Descriptions

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

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

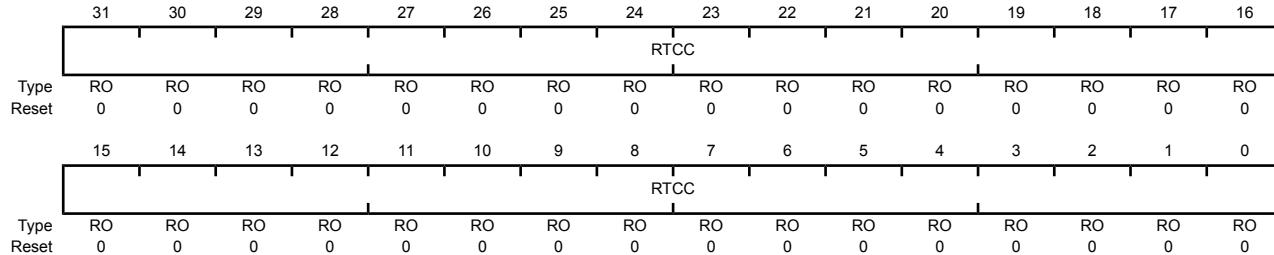
This register is the current 32-bit value of the RTC counter.

### Hibernation RTC Counter (HIBRTCC)

Base 0x400F.C000

Offset 0x000

Type RO, reset 0x0000.0000



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

31:0	RTCC	RO	0x0000.0000	RTC Counter
------	------	----	-------------	-------------

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

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

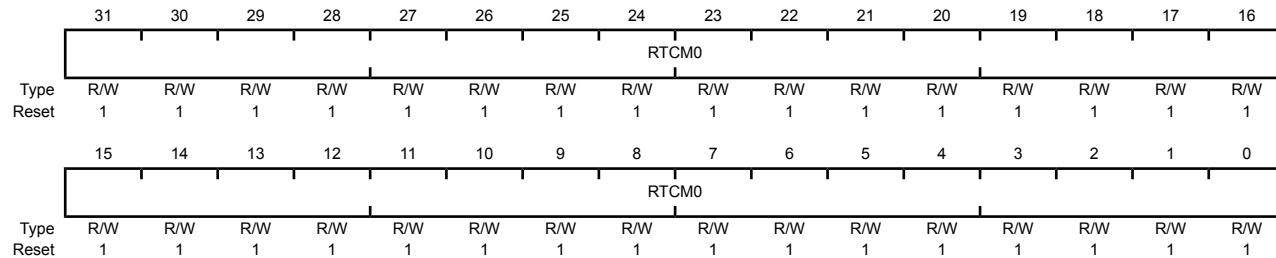
This register is the 32-bit match 0 register for the RTC counter.

### Hibernation RTC Match 0 (HIBRTCM0)

Base 0x400F.C000

Offset 0x004

Type R/W, reset 0xFFFF.FFFF



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

31:0	RTCM0	R/W	0xFFFF.FFFF	RTC Match 0
------	-------	-----	-------------	-------------

A write loads the value into the RTC match register.

A read returns the current match value.

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

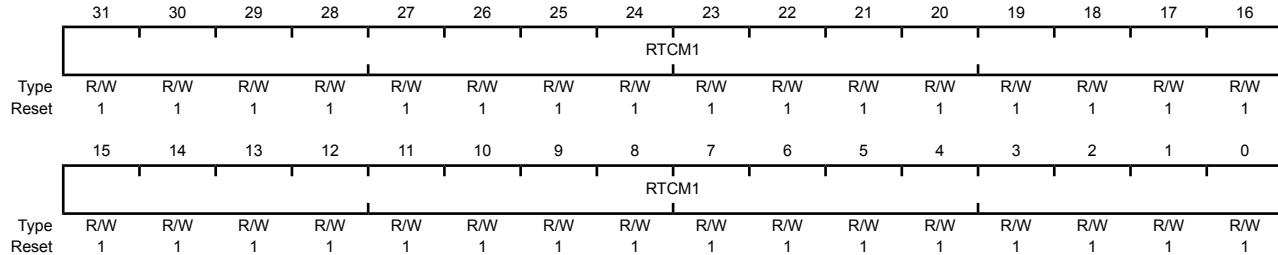
This register is the 32-bit match 1 register for the RTC counter.

Hibernation RTC Match 1 (HIBRTCM1)

Base 0x400F.C000

Offset 0x008

Type R/W, reset 0xFFFF.FFFF



Bit/Field                  Name                  Type                  Reset                  Description

31:0                  RTCM1                  R/W                  0xFFFF.FFFF                  RTC Match 1

A write loads the value into the RTC match register.

A read returns the current match value.

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

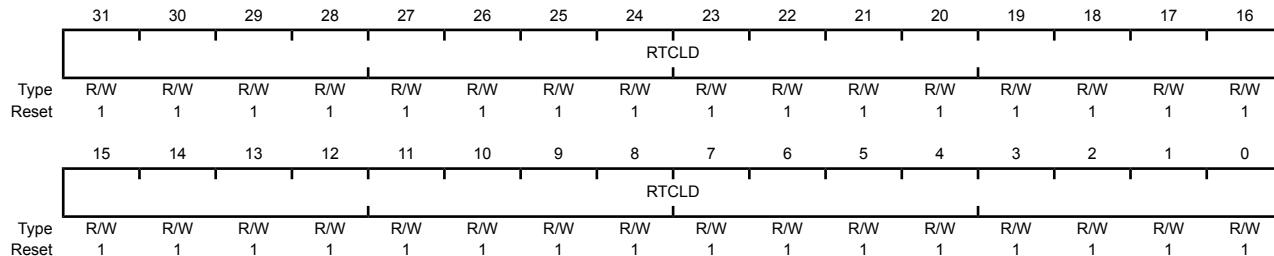
This register is the 32-bit value loaded into the RTC counter.

### Hibernation RTC Load (HIBRTCLD)

Base 0x400F.C000

Offset 0x00C

Type R/W, reset 0xFFFF.FFFF



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

31:0	RTCLD	R/W	0xFFFF.FFFF	RTC Load
------	-------	-----	-------------	----------

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

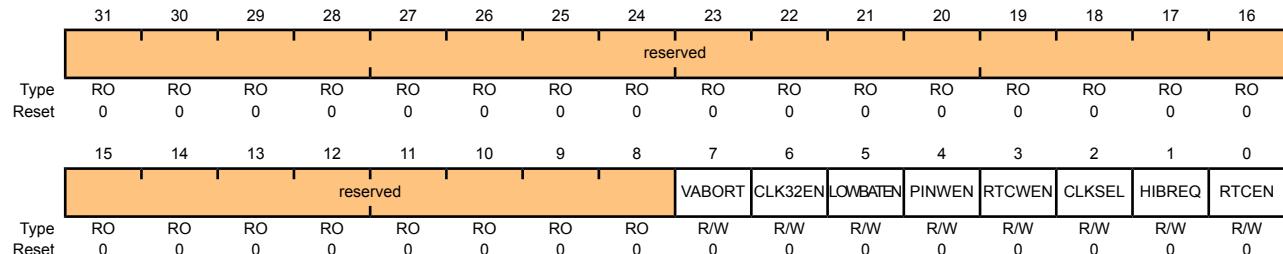
A read returns the 32-bit load value.

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

This register is the control register for the Hibernation module.

### Hibernation Control (HIBCTL)

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



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	VABORT	R/W	0	Power Cut Abort Enable 0: Power cut occurs during a low-battery alert 1: Power cut is aborted
6	CLK32EN	R/W	0	32-kHz Oscillator Enable 0: Disabled 1: Enabled  This bit must be enabled to use the Hibernation module. If a crystal is used, then software should wait 20 ms after setting this bit to allow the crystal to power up and stabilize.
5	LOWBATEN	R/W	0	Low Battery Monitoring Enable 0: Disabled 1: Enabled  When set, low battery voltage detection is enabled.
4	PINWEN	R/W	0	External $\overline{\text{WAKE}}$ Pin Enable 0: Disabled 1: Enabled  When set, an external event on the $\overline{\text{WAKE}}$ pin will re-power the device.
3	RTCWEN	R/W	0	RTC Wake-up Enable 0: Disabled 1: Enabled  When set, an RTC match event (RTCM0 or RTCM1) will re-power the device based on the RTC counter value matching the corresponding match register 0 or 1.
1:0	CLKSEL	R/W	0	
0	HIBREQ	R/W	0	
0	RTCN	R/W	0	

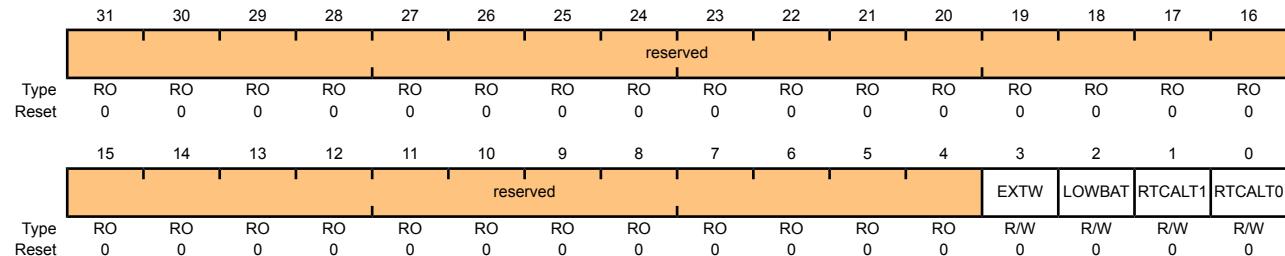
Bit/Field	Name	Type	Reset	Description
2	CLKSEL	R/W	0	Hibernation Module Clock Select 0: Use Divide by 128 output. Use this value for a 4-MHz crystal. 1: Use raw output. Use this value for a 32-kHz oscillator.
1	HIBREQ	R/W	0	Hibernation Request 0: Disabled 1: Hibernation initiated After a wake-up event, this bit is cleared by hardware.
0	RTCEN	R/W	0	RTC Timer Enable 0: Disabled 1: Enabled

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

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

### Hibernation Interrupt Mask (HIBIM)

Base 0x400F.C000  
Offset 0x014  
Type R/W, reset 0x0000.0000



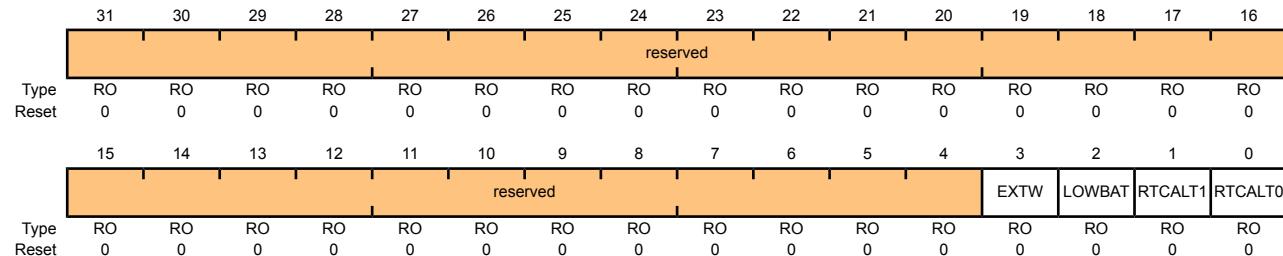
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	R/W	0	External Wake-Up Interrupt Mask 0: Masked 1: Unmasked
2	LOWBAT	R/W	0	Low Battery Voltage Interrupt Mask 0: Masked 1: Unmasked
1	RTCAUT1	R/W	0	RTC Alert1 Interrupt Mask 0: Masked 1: Unmasked
0	RTCAUT0	R/W	0	RTC Alert0 Interrupt Mask 0: Masked 1: Unmasked

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

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

### Hibernation Raw Interrupt Status (HIBRIS)

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



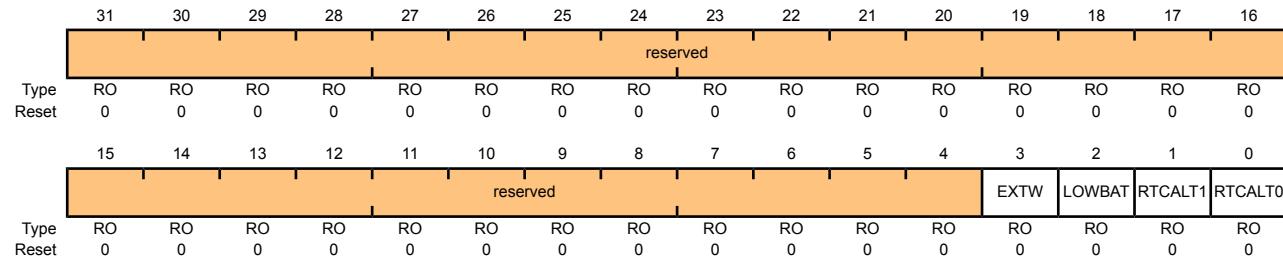
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	RO	0	External Wake-Up Raw Interrupt Status
2	LOWBAT	RO	0	Low Battery Voltage Raw Interrupt Status
1	RTCALT1	RO	0	RTC Alert1 Raw Interrupt Status
0	RTCALT0	RO	0	RTC Alert0 Raw Interrupt Status

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

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

### Hibernation Masked Interrupt Status (HIBMIS)

Base 0x400F.C000  
Offset 0x01C  
Type RO, reset 0x0000.0000



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	EXTW	RO	0	External Wake-Up Masked Interrupt Status
2	LOWBAT	RO	0	Low Battery Voltage Masked Interrupt Status
1	RTCALT1	RO	0	RTC Alert1 Masked Interrupt Status
0	RTCALTO	RO	0	RTC Alert0 Masked Interrupt Status

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

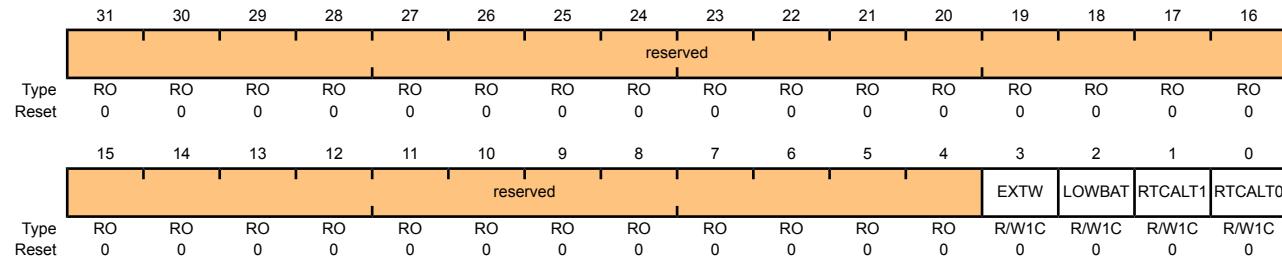
This register is the interrupt write-one-to-clear register for the Hibernation module interrupt sources.

### Hibernation Interrupt Clear (HIBIC)

Base 0x400F.C000

Offset 0x020

Type R/W1C, reset 0x0000.0000



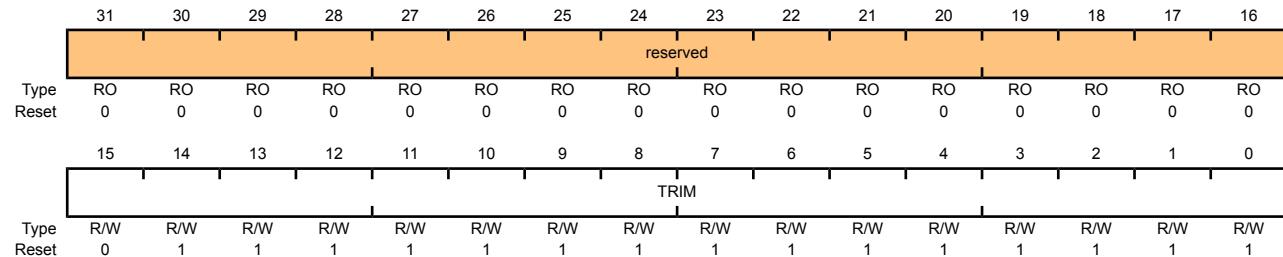
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	R/W1C	0	External Wake-Up Masked Interrupt Clear Reads return an indeterminate value.
2	LOWBAT	R/W1C	0	Low Battery Voltage Masked Interrupt Clear Reads return an indeterminate value.
1	RTCALT1	R/W1C	0	RTC Alert1 Masked Interrupt Clear Reads return an indeterminate value.
0	RTCALT0	R/W1C	0	RTC Alert0 Masked Interrupt Clear Reads return an indeterminate value.

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

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

### Hibernation RTC Trim (HIBRTCT)

Base 0x400F.C000  
Offset 0x024  
Type R/W, reset 0x0000.7FFF



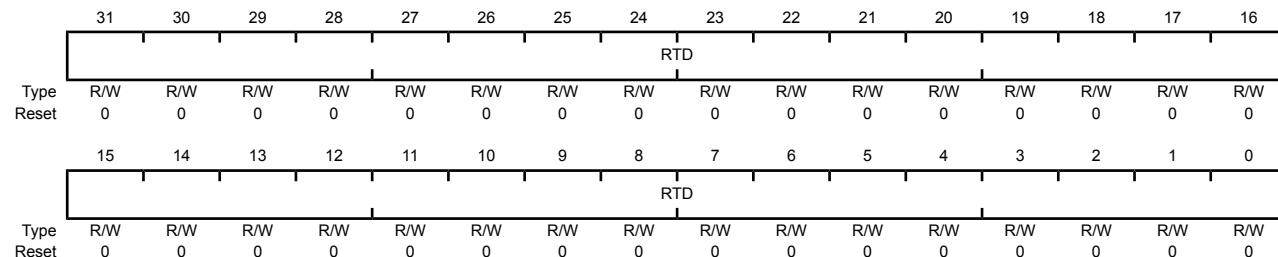
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TRIM	R/W	0x7FFF	RTC Trim Value  This value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

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

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

### Hibernation Data (HIBDATA)

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



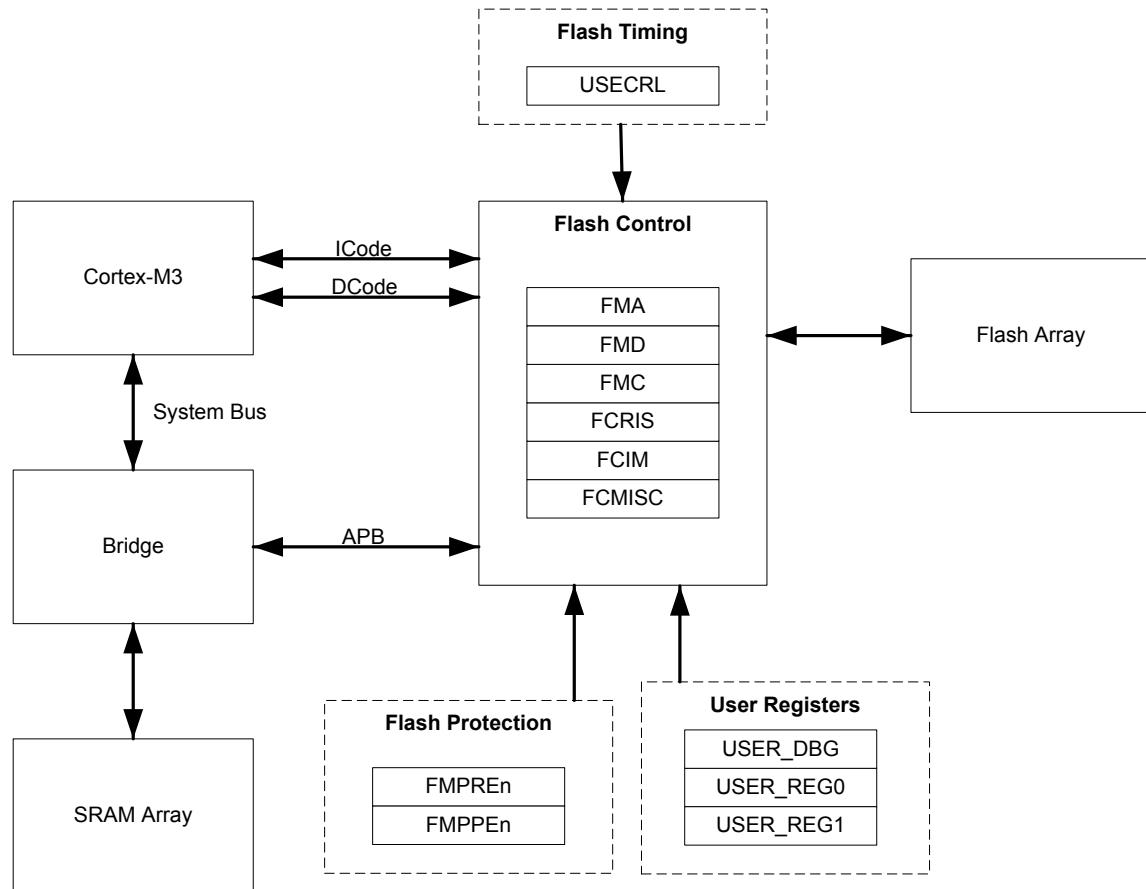
Bit/Field	Name	Type	Reset	Description
31:0	RTD	R/W	0x0000.0000	Hibernation Module NV Registers[63:0]

## 8 Internal Memory

The LM3S8962 microcontroller comes with 64 KB of bit-banded SRAM and 256 KB of flash memory. The flash controller provides a user-friendly interface, making flash programming a simple task. Flash protection can be applied to the flash memory on a 2-KB block basis.

### 8.1 Block Diagram

**Figure 8-1. Flash Block Diagram**



### 8.2 Functional Description

This section describes the functionality of both the flash and SRAM memories.

#### 8.2.1 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:

```
bit-band alias = bit-band base + (byte offset * 32) + (bit number * 4)
```

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

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

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

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

## 8.2.2 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 594 for a preprogrammed flash-resident utility used to download code to the flash memory of a device without the use of a debug interface.

### 8.2.2.1 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.

### 8.2.2.2 Flash Memory Protection

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

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

The policies may be combined as shown in Table 8-1 on page 143.

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

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

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

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

The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This implements a policy of open access and programmability. The register bits may be changed by writing the specific register bit. The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. Details on programming these bits are discussed in “Nonvolatile Register Programming” on page 144.

## 8.3 Flash Memory Initialization and Configuration

### 8.3.1 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**.

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

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

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

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

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

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

### 8.3.2 Nonvolatile Register Programming

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

**Important:** These registers can only have bits changed from 1 to 0 by the user and there is no mechanism for the user to erase them back to a 1 value.

In addition, the **USER\_REG0**, **USER\_REG1**, and **USER\_DBG** use bit 31 (**NW**) of their respective registers to indicate that they are available for user write. These three registers can only be written once whereas the flash protection registers may be written multiple times. Table 8-2 on page 144 provides the FMA address required for commitment of each of the registers and the source of the data to be written when the COMT bit of the **FMC** register is written with a value of 0xA442.0008. After writing the COMT bit, the user may poll the **FMC** register to wait for the commit operation to complete.

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

Register to be Committed	FMA Value	Data Source
FMPRE0	0x0000.0000	FMPRE0
FMPRE1	0x0000.0002	FMPRE1
FMPRE2	0x0000.0004	FMPRE2
FMPRE3	0x0000.0008	FMPRE3
FMPPE0	0x0000.0001	FMPPE0
FMPPE1	0x0000.0003	FMPPE1
FMPPE2	0x0000.0005	FMPPE2
FMPPE3	0x0000.0007	FMPPE3
USER_REG0	0x8000.0000	USER_REG0
USER_REG1	0x8000.0001	USER_REG1
USER_DBG	0x7510.0000	FMD

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

## 8.4 Register Map

Table 8-3 on page 144 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** registers are relative to the Flash control base address of 0x400F.D000. The **FMPREn**, **FMPPEn**, **USECRL**, **USER\_DBG**, and **USER\_REGn** registers are relative to the System Control base address of 0x400F.E000.

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

Offset	Name	Type	Reset	Description	See page
<b>Flash Control Offset</b>					
0x000	FMA	R/W	0x0000.0000	Flash Memory Address	146

Offset	Name	Type	Reset	Description	See page
0x004	FMD	R/W	0x0000.0000	Flash Memory Data	147
0x008	FMC	R/W	0x0000.0000	Flash Memory Control	148
0x00C	FCRIS	RO	0x0000.0000	Flash Controller Raw Interrupt Status	150
0x010	FCIM	R/W	0x0000.0000	Flash Controller Interrupt Mask	151
0x014	FCMISC	R/W1C	0x0000.0000	Flash Controller Masked Interrupt Status and Clear	152
<b>System Control Offset</b>					
0x130	FMPRE0	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 0	154
0x200	FMPRE0	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 0	154
0x134	FMPPE0	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 0	155
0x400	FMPPE0	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 0	155
0x140	USECRL	R/W	0x31	USec Reload	153
0x1D0	USER_DBG	R/W	0xFFFF.FFFE	User Debug	156
0x1E0	USER_REG0	R/W	0xFFFF.FFFF	User Register 0	157
0x1E4	USER_REG1	R/W	0xFFFF.FFFF	User Register 1	158
0x204	FMPRE1	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 1	159
0x208	FMPRE2	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 2	160
0x20C	FMPRE3	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 3	161
0x404	FMPPE1	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 1	162
0x408	FMPPE2	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 2	163
0x40C	FMPPE3	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 3	164

## 8.5 Flash Register Descriptions (Flash Control Offset)

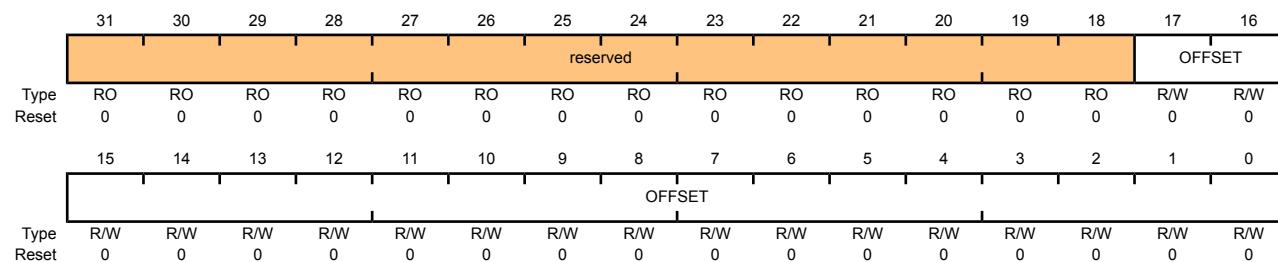
The remainder of this section lists and describes the Flash Memory registers, in numerical order by address offset. Registers in this section are relative to the Flash control base address of 0x400F.D000.

## 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
31:18	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.
17:0	OFFSET	R/W	0x0	Address Offset  Address offset in flash where operation is performed, except for nonvolatile registers (see “Nonvolatile Register Programming” on page 144 for details on values for this field).

## Register 2: Flash Memory Data (FMD), offset 0x004

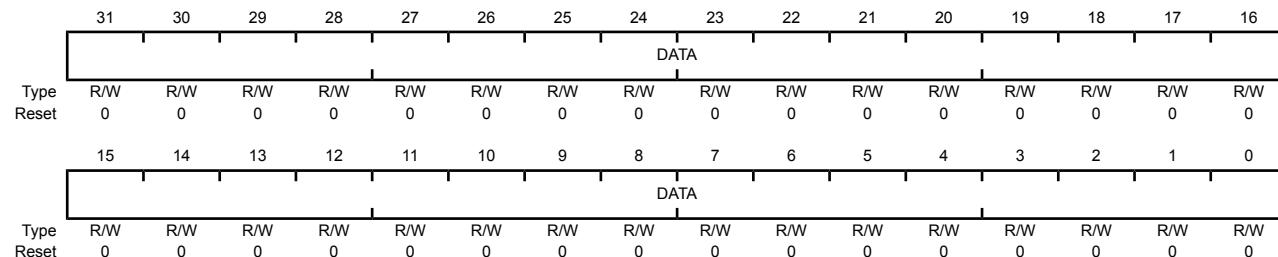
This register contains the data to be written during the programming cycle or read during the read cycle. Note that the contents of this register are undefined for a read access of an execute-only block. This register is not used during the erase cycles.

### 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 146). If the access is a write access, the data contained in the **Flash Memory Data (FMD)** register (see page 147) is written.

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

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

### Flash Memory Control (FMC)

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

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

Bit/Field	Name	Type	Reset	Description
31:16	WRKEY	WO	0x0	Flash Write Key  This field contains a write key, which is used to minimize the incidence of accidental flash writes. The value 0xA442 must be written into this field for a write to occur. Writes to the <b>FMC</b> register without this <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.

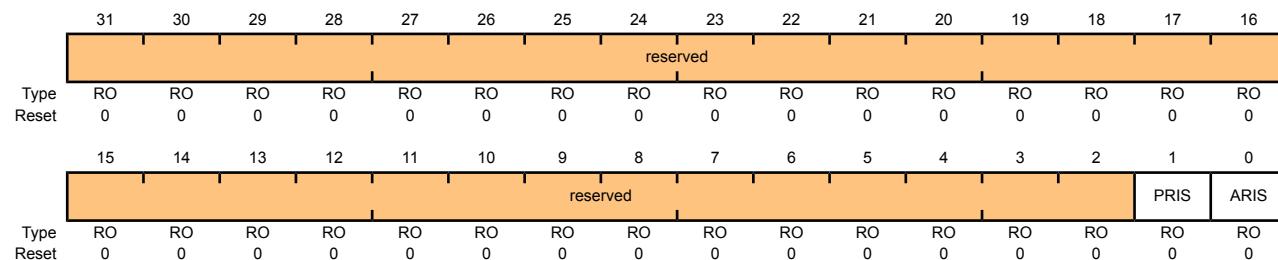
Bit/Field	Name	Type	Reset	Description
1	ERASE	R/W	0	<p>Erase a Page of Flash Memory</p> <p>If this bit is set, the page of flash main memory as specified by the contents of <b>FMA</b> is erased. A write of 0 has no effect on the state of this bit.</p> <p>If read, the state of the previous erase access 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



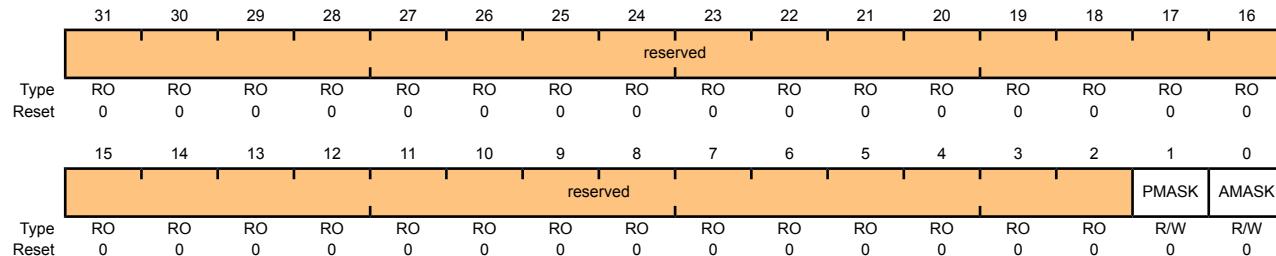
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PRIS	RO	0	Programming Raw Interrupt Status  This bit indicates the current state of the programming cycle. If set, the programming cycle completed; if cleared, the programming cycle has not completed. Programming cycles are either write or erase actions generated through the <b>Flash Memory Control (FMC)</b> register bits (see page 148).
0	ARIS	RO	0	Access Raw Interrupt Status  This bit indicates if the flash was improperly accessed. If set, the program tried to access the flash counter to the policy as set in the <b>Flash Memory Protection Read Enable (FMPREn)</b> and <b>Flash Memory Protection Program Enable (FMPPEn)</b> registers. Otherwise, no access has tried to improperly access the flash.

## 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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PMASK	R/W	0	Programming Interrupt Mask  This bit controls the reporting of the programming raw interrupt status to the controller. If set, a programming-generated interrupt is promoted to the controller. Otherwise, interrupts are recorded but suppressed from the controller.
0	AMASK	R/W	0	Access Interrupt Mask  This bit controls the reporting of the access raw interrupt status to the controller. If set, an access-generated interrupt is promoted to the controller. Otherwise, interrupts are recorded but suppressed from the controller.

## Register 6: Flash Controller Masked Interrupt Status and Clear (FCMISC), offset 0x014

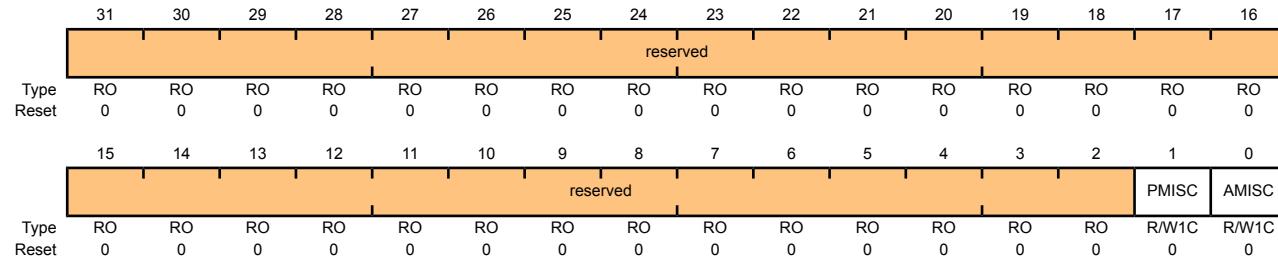
This register provides two functions. First, it reports the cause of an interrupt by indicating which interrupt source or sources are signalling the interrupt. Second, it serves as the method to clear the interrupt reporting.

Flash Controller Masked Interrupt Status and Clear (FCMISC)

Base 0x400F.D000

Offset 0x014

Type R/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	PMISC	R/W1C	0	Programming Masked Interrupt Status and Clear This bit indicates whether an interrupt was signaled because a programming cycle completed and was not masked. This bit is cleared by writing a 1. The PRIS bit in the <b>FCRIS</b> register (see page 150) is also cleared when the PMISC bit is cleared.
0	AMISC	R/W1C	0	Access Masked Interrupt Status and Clear This bit indicates whether an interrupt was signaled because an improper access was attempted and was not masked. This bit is cleared by writing a 1. The ARIS bit in the <b>FCRIS</b> register is also cleared when the AMISC bit is cleared.

## 8.6 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

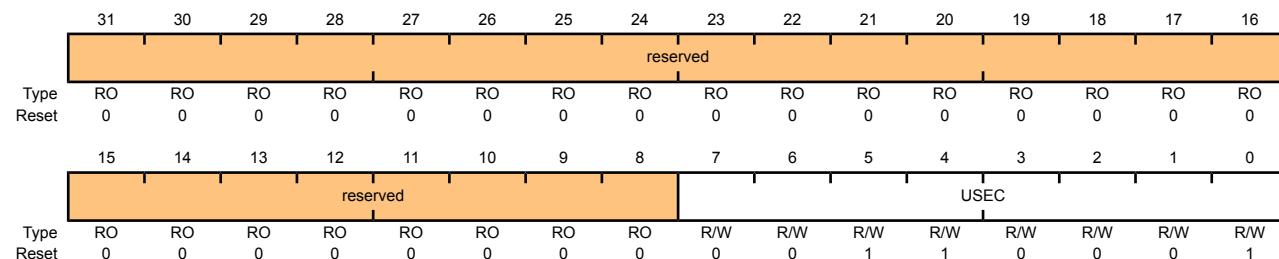
This register is provided as a means of creating a 1- $\mu$ s tick divider reload value for the flash controller. The internal flash has specific minimum and maximum requirements on the length of time the high voltage write pulse can be applied. It is required that this register contain the operating 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



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	USEC	R/W	0x31	Microsecond Reload Value MHz -1 of the controller clock when the flash is being erased or programmed. USEC should be set to 0x31 (50 MHz) whenever the flash is being erased or programmed.

## Register 8: Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x130 and 0x200

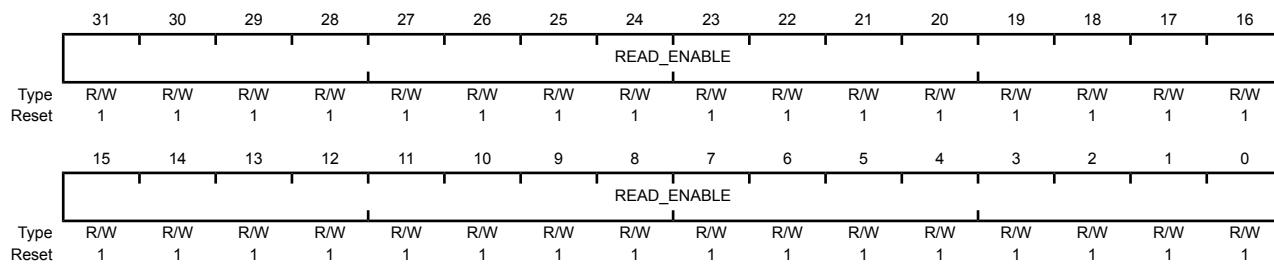
**Note:** This register is aliased for backwards compatibility.

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

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

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

Base 0x400F.D000  
Offset 0x130 and 0x200  
Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0xFFFFFFFF	Flash Read Enable

Enables 2-KB flash blocks to be executed or read. The policies may be combined as shown in the table "Flash Protection Policy Combinations".

Value	Description
0xFFFFFFFF	Enables 256 KB of flash.

## Register 9: Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x134 and 0x400

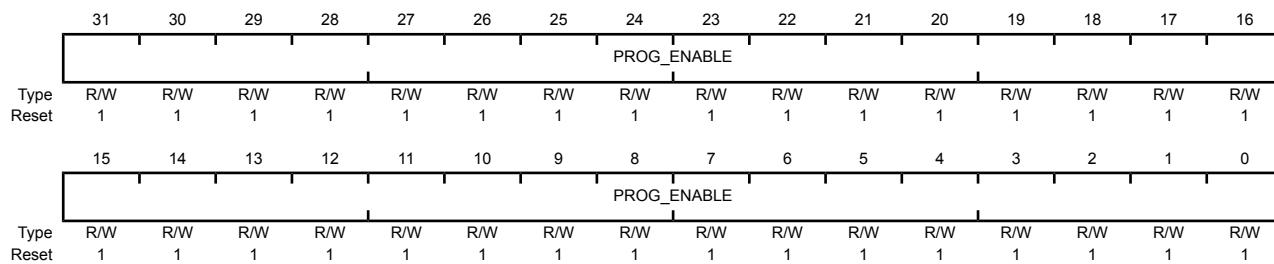
**Note:** This register is aliased for backwards compatibility.

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

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

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

Base 0x400F.D000  
Offset 0x134 and 0x400  
Type R/W, reset 0xFFFF.FFFF



#### Bit/Field      Name      Type      Reset      Description

31:0	PROG_ENABLE	R/W	0xFFFFFFFF	Flash Programming Enable
------	-------------	-----	------------	--------------------------

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

Value	Description
0xFFFFFFFF	Enables 256 KB of flash.

## Register 10: User Debug (USER\_DBG), offset 0x1D0

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

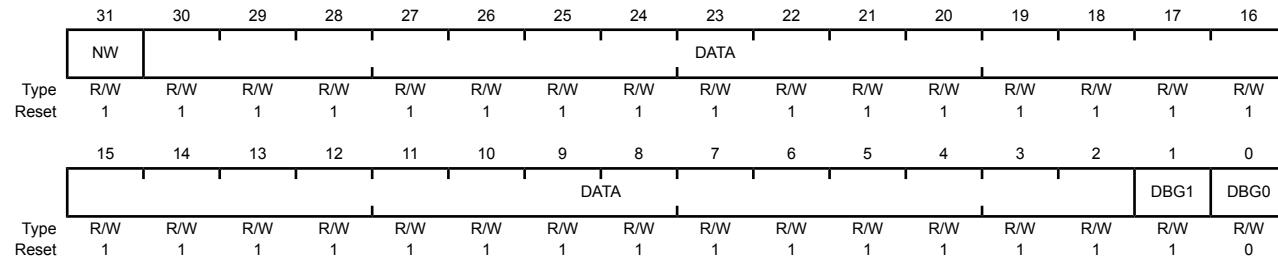
This register provides a write-once mechanism to disable external debugger access to the device in addition to 27 additional bits of user-defined data. The `DBG0` bit (bit 0) is set to 0 from the factory and the `DBG1` bit (bit 1) is set to 1, which enables external debuggers. Changing the `DBG1` bit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

### User Debug (USER\_DBG)

Base 0x400F.E000

Offset 0x1D0

Type R/W, reset 0xFFFF.FFFF



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

## Register 11: User Register 0 (USER\_REG0), offset 0x1E0

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

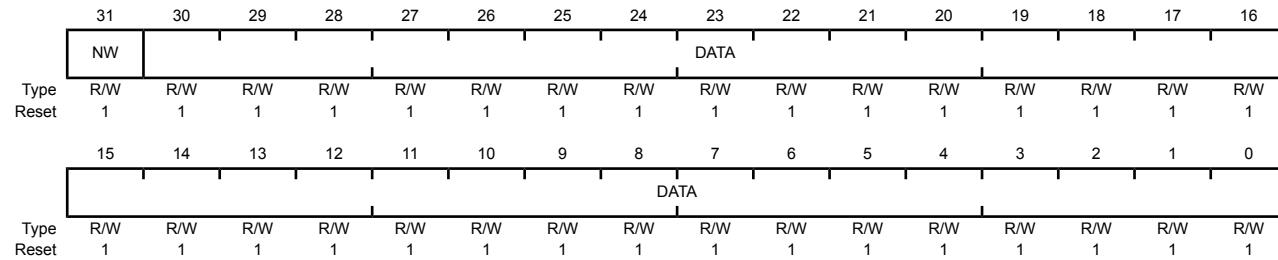
This register provides 31 bits of user-defined data that is non-volatile and can only be written once. Bit 31 indicates that the register is available to be written and is controlled through hardware to ensure that the register is only written once. The write-once characteristics of this register are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

### User Register 0 (USER\_REG0)

Base 0x400F.E000

Offset 0x1E0

Type R/W, reset 0xFFFF.FFFF



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

## Register 12: User Register 1 (USER\_REG1), offset 0x1E4

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

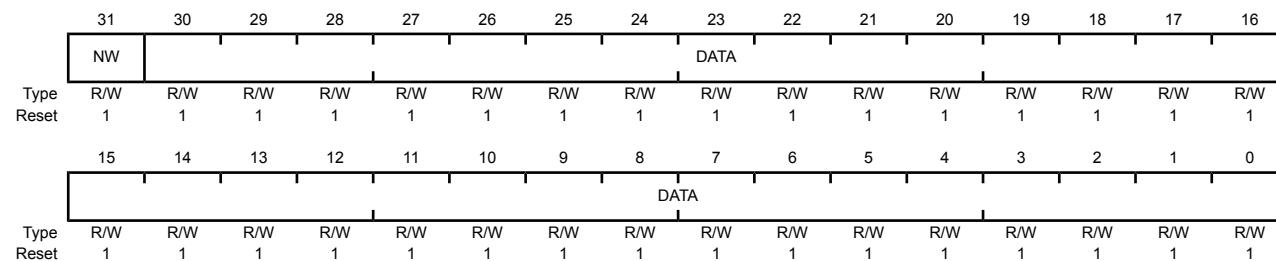
This register provides 31 bits of user-defined data that is non-volatile and can only be written once. Bit 31 indicates that the register is available to be written and is controlled through hardware to ensure that the register is only written once. The write-once characteristics of this register are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

### User Register 1 (USER\_REG1)

Base 0x400F.E000

Offset 0x1E4

Type R/W, reset 0xFFFF.FFFF



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

## Register 13: Flash Memory Protection Read Enable 1 (FMPRE1), offset 0x204

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

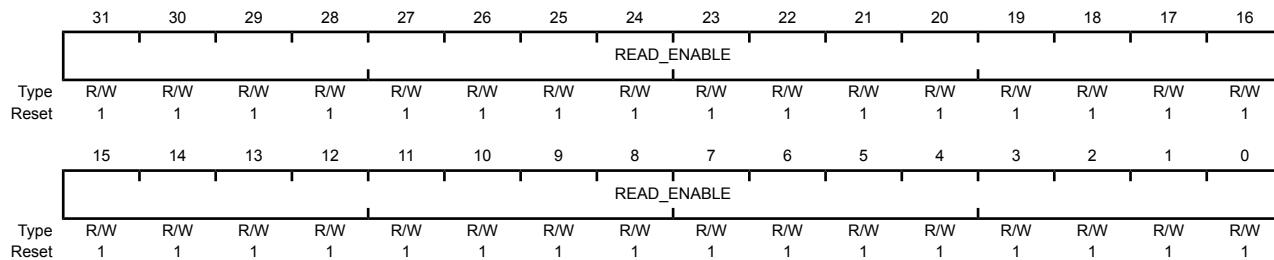
This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves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

### Flash Memory Protection Read Enable 1 (FMPRE1)

Base 0x400F.E000

Offset 0x204

Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0xFFFFFFFF	Flash Read Enable Enables 2-KB flash blocks to be executed or read. The policies may be combined as shown in the table "Flash Protection Policy Combinations".
				Value      Description
				0xFFFFFFFF      Enables 256 KB of flash.

## Register 14: Flash Memory Protection Read Enable 2 (FMPRE2), offset 0x208

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

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

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

Base 0x400F.E000

Offset 0x208

Type R/W, reset 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	1	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	1

Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0xFFFFFFFF	Flash Read Enable
Enables 2-KB flash blocks to be executed or read. The policies may be combined as shown in the table "Flash Protection Policy Combinations".				
Value	Description			
0xFFFFFFFF	Enables 256 KB of flash.			

## Register 15: Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C

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

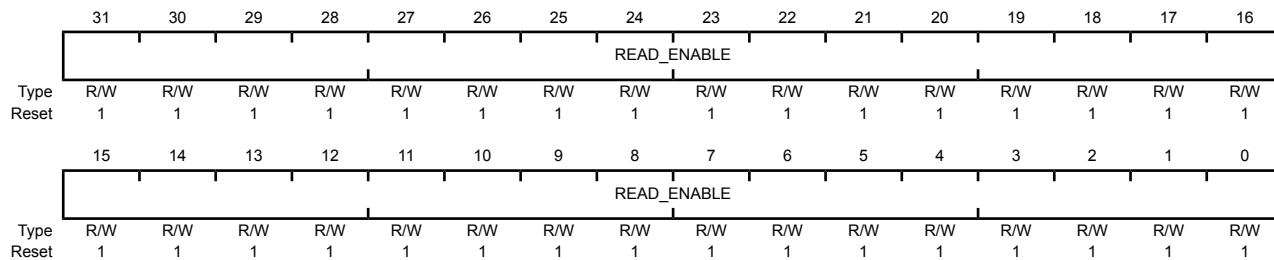
This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves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

### Flash Memory Protection Read Enable 3 (FMPRE3)

Base 0x400F.E000

Offset 0x20C

Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0xFFFFFFFF	Flash Read Enable Enables 2-KB flash blocks to be executed or read. The policies may be combined as shown in the table "Flash Protection Policy Combinations".
				Value      Description
				0xFFFFFFFF      Enables 256 KB of flash.

## Register 16: Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404

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

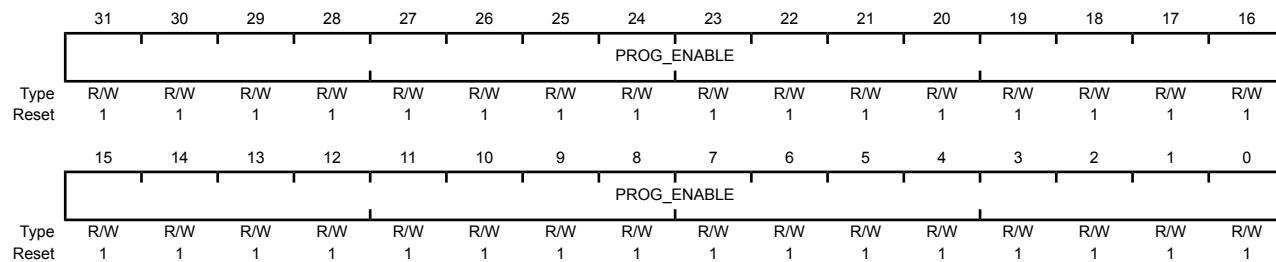
This register stores the execute-only protection bits for each 2-KB flash block (**FMPREn** stores the execute-only bits). This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves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

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

Base 0x400F.E000

Offset 0x404

Type R/W, reset 0xFFFF.FFFF



Bit/Field                  Name                  Type                  Reset                  Description

31:0                  PROG\_ENABLE                  R/W                  0xFFFFFFFF                  Flash Programming Enable

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

Value                  Description

0xFFFFFFFF                  Enables 256 KB of flash.

## Register 17: Flash Memory Protection Program Enable 2 (FMPPE2), offset 0x408

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

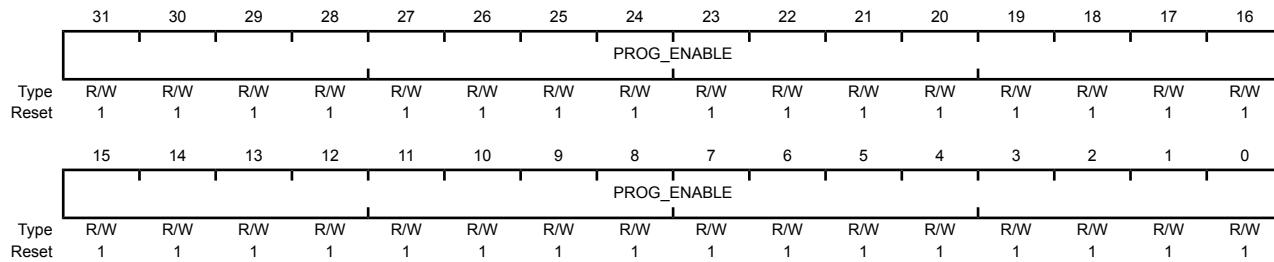
This register stores the execute-only protection bits for each 2-KB flash block (**FMPREn** stores the execute-only bits). This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves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

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

Base 0x400F.E000

Offset 0x408

Type R/W, reset 0xFFFF.FFFF



Bit/Field      Name      Type      Reset      Description

31:0      PROG\_ENABLE      R/W      0xFFFFFFFF      Flash Programming Enable

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

Value      Description

0xFFFFFFFF      Enables 256 KB of flash.

## Register 18: Flash Memory Protection Program Enable 3 (FMPPE3), offset 0x40C

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

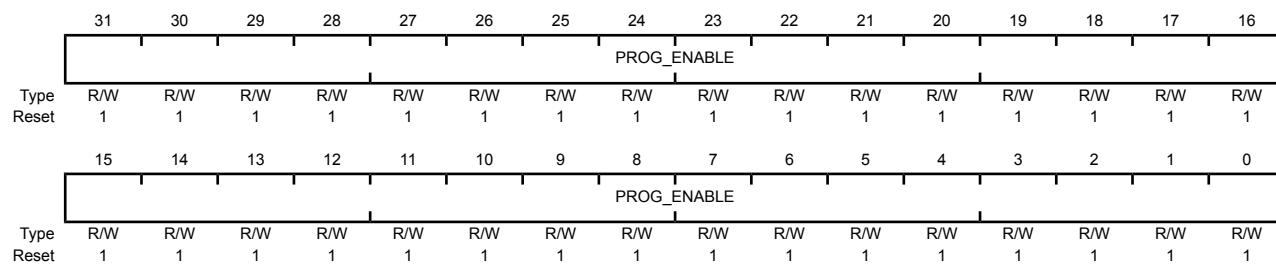
This register stores the execute-only protection bits for each 2-KB flash block (**FMPREn** stores the execute-only bits). This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves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

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

Base 0x400F.E000

Offset 0x40C

Type R/W, reset 0xFFFF.FFFF



Bit/Field      Name      Type      Reset      Description

31:0      PROG\_ENABLE      R/W      0xFFFFFFFF      Flash Programming Enable

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

Value      Description

0xFFFFFFFF      Enables 256 KB of flash.

## 9 General-Purpose Input/Outputs (GPIOs)

The GPIO module is composed of seven physical GPIO blocks, each corresponding to an individual GPIO port (Port A, Port B, Port C, Port D, Port E, Port F, and Port G, ). The GPIO module is FiRM-compliant and supports 5-42 programmable input/output pins, depending on the peripherals being used.

The GPIO module has the following features:

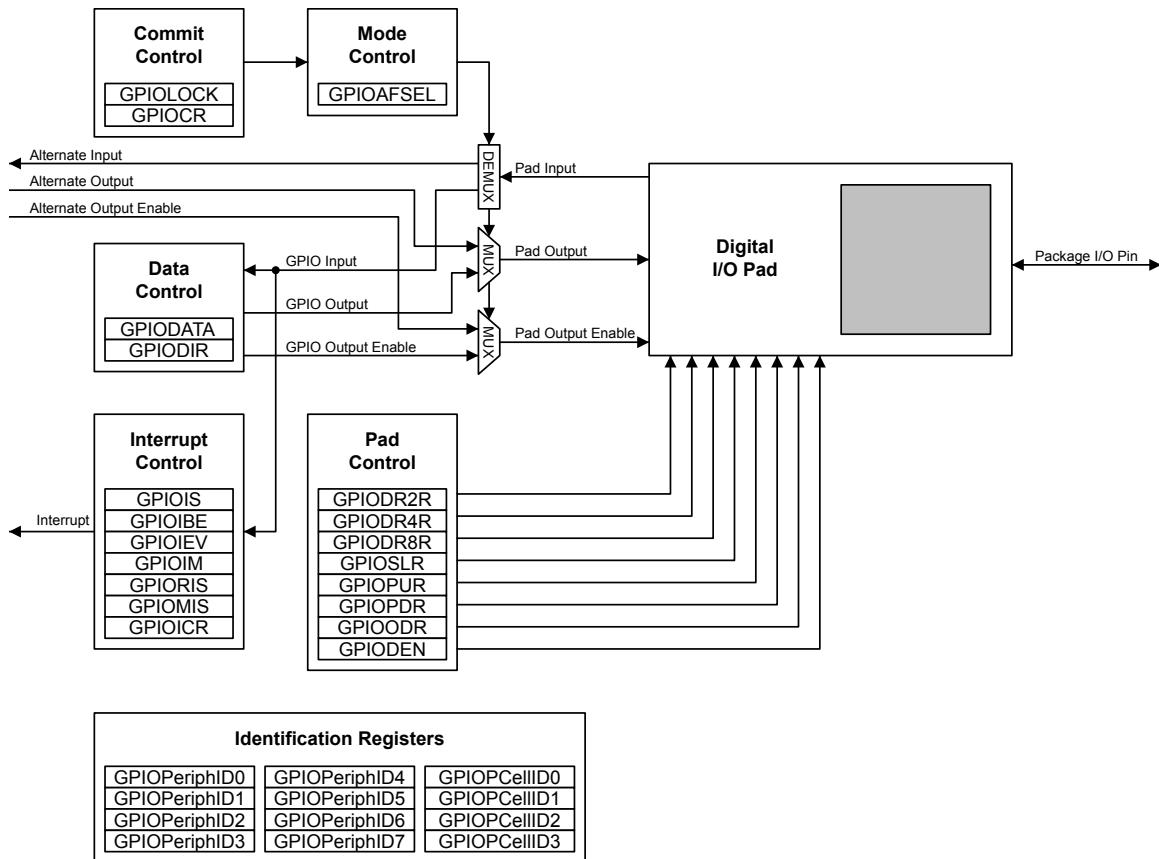
- Programmable control for GPIO interrupts
  - Interrupt generation masking
  - Edge-triggered on rising, falling, or both
  - Level-sensitive on High or Low values
- 5-V-tolerant input/outputs
- Bit masking in both read and write operations through address lines
- Programmable control for GPIO pad configuration
  - Weak pull-up or pull-down resistors
  - 2-mA, 4-mA, and 8-mA pad drive
  - Slew rate control for the 8-mA drive
  - Open drain enables
  - Digital input enables

### 9.1 Functional Description

**Important:** All GPIO pins are tri-stated by default (**GPIOAFSEL=0**, **GPIODEN=0**, **GPIOPDR=0**, and **GPIOPUR=0**), with the exception of the five JTAG/SWD pins (PB7 and PC[3:0]). The JTAG/SWD pins default to their JTAG/SWD functionality (**GPIOAFSEL=1**, **GPIODEN=1** and **GPIOPUR=1**). A Power-On-Reset ( $\overline{\text{POR}}$ ) or asserting  $\overline{\text{RST}}$  puts both groups of pins back to their default state.

Each GPIO port is a separate hardware instantiation of the same physical block (see Figure 9-1 on page 166). The LM3S8962 microcontroller contains seven ports and thus seven of these physical GPIO blocks.

Figure 9-1. GPIO Port Block Diagram



## 9.1.1 Data Control

The data control registers allow software to configure the operational modes of the GPIOs. The data direction register configures the GPIO as an input or an output while the data register either captures incoming data or drives it out to the pads.

### 9.1.1.1 Data Direction Operation

The **GPIO Direction (GPIODIR)** register (see page 173) 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.

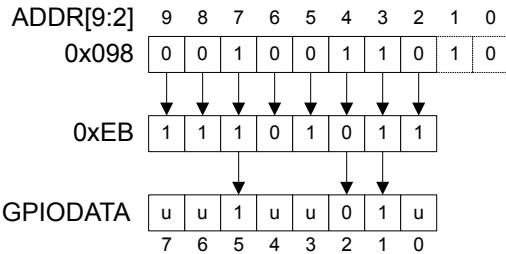
### 9.1.1.2 Data Register Operation

To aid in the efficiency of software, the GPIO ports allow for the modification of individual bits in the **GPIO Data (GPIODATA)** register (see page 172) by using bits [9:2] of the address bus as a mask. This allows software drivers to modify individual GPIO pins in a single instruction, without affecting the state of the other pins. This is in contrast to the "typical" method of doing a read-modify-write operation to set or clear an individual GPIO pin. To accommodate this feature, the **GPIODATA** register covers 256 locations in the memory map.

During a write, if the address bit associated with that data bit is set to 1, the value of the **GPIODATA** register is altered. If it is cleared to 0, it is left unchanged.

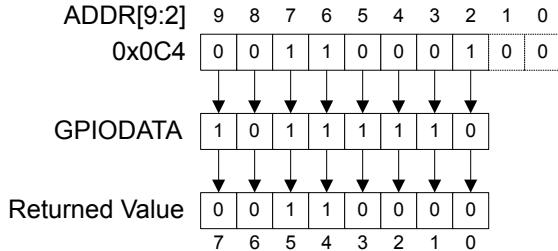
For example, writing a value of 0xEB to the address GPIO DATA + 0x098 would yield as shown in Figure 9-2 on page 167, where *u* is data unchanged by the write.

**Figure 9-2. 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 9-3 on page 167.

**Figure 9-3. GPIO DATA Read Example**



### 9.1.2 Interrupt Control

The interrupt capabilities of each GPIO port are controlled by a set of seven registers. With these registers, it is possible to select the source of the interrupt, its polarity, and the edge properties. When one or more GPIO inputs cause an interrupt, a single interrupt output is sent to the interrupt controller for the entire GPIO port. For edge-triggered interrupts, software must clear the interrupt to enable any further interrupts. For a level-sensitive interrupt, it is assumed that the external source holds the level constant for the interrupt to be recognized by the controller.

Three registers are required to define the edge or sense that causes interrupts:

- **GPIO Interrupt Sense (GPIOIS)** register (see page 174)
- **GPIO Interrupt Both Edges (GPIOIBE)** register (see page 175)
- **GPIO Interrupt Event (GPIOIEV)** register (see page 176)

Interrupts are enabled/disabled via the **GPIO Interrupt Mask (GPIOIM)** register (see page 177).

When an interrupt condition occurs, the state of the interrupt signal can be viewed in two locations: the **GPIO Raw Interrupt Status (GPIOISR)** and **GPIO Masked Interrupt Status (GPIOIMS)** registers (see page 178 and page 179). As the name implies, the **GPIOIMS** 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 (GPIOIM is set to 1), not only is an interrupt for PortB generated, but an external trigger signal is sent to the ADC. If the **ADC Event Multiplexer Select (ADCEMUX)** register is configured to use the external trigger, an ADC conversion is initiated.

If no other PortB pins are being used to generate interrupts, the ARM Integrated Nested Vectored Interrupt Controller (NVIC) Interrupt Set Enable (SETNA) register can disable the PortB interrupts and the ADC interrupt can be used to read back the converted data. Otherwise, the PortB interrupt handler needs to ignore and clear interrupts on B4, and wait for the ADC interrupt or the ADC interrupt needs to be disabled in the SETNA register and the PortB interrupt handler polls the ADC registers until the conversion is completed.

Interrupts are cleared by writing a 1 to the **GPIO Interrupt Clear (GPIOICR)** register (see page 180).

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.

### 9.1.3 Mode Control

The GPIO pins can be controlled by either hardware or software. When hardware control is enabled via the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 181), 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.

### 9.1.4 Commit Control

The commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 181) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 191) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 192) have been set to 1.

### 9.1.5 Pad Control

The pad control registers allow for GPIO pad configuration by software based on the application requirements. The pad control registers include the **GPIODR2R**, **GPIODR4R**, **GPIODR8R**, **GPIOODR**, **GPIOPUR**, **GPIOPDR**, **GPIOSLR**, and **GPIODEN** registers.

### 9.1.6 Identification

The identification registers configured at reset allow software to detect and identify the module as a GPIO block. The identification registers include the **GPIOPeriphID0-GPIOPeriphID7** registers as well as the **GPIOCellID0-GPIOCellID3** registers.

## 9.2 Initialization and Configuration

To use the GPIO, the peripheral clock must be enabled by setting the appropriate GPIO Port bit field (GPIO<sub>n</sub>) in the **RCGC2** register.

On reset, all GPIO pins (except for the five JTAG pins) are configured out of reset to be undriven (tristate): **GPIOAFSEL=0**, **GPIODEN=0**, **GPIOPDR=0**, and **GPIOPUR=0**. Table 9-1 on page 169 shows all possible configurations of the GPIO pads and the control register settings required to achieve them. Table 9-2 on page 169 shows how a rising edge interrupt would be configured for pin 2 of a GPIO port.

**Table 9-1. GPIO Pad Configuration Examples**

Configuration	GPIO Register Bit Value <sup>a</sup>									
	AFSEL	DIR	ODR	DEN	PUR	PDR	DR2R	DR4R	DR8R	SLR
Digital Input (GPIO)	0	0	0	1	?	?	X	X	X	X
Digital Output (GPIO)	0	1	0	1	?	?	?	?	?	?
Open Drain Input (GPIO)	0	0	1	1	X	X	X	X	X	X
Open Drain Output (GPIO)	0	1	1	1	X	X	?	?	?	?
Open Drain Input/Output (I <sup>2</sup> C)	1	X	1	1	X	X	?	?	?	?
Digital Input (Timer CCP)	1	X	0	1	?	?	X	X	X	X
Digital Input (QEI)	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 9-2. GPIO Interrupt Configuration Example**

Register	Desired Interrupt Event Trigger	Pin 2 Bit Value <sup>a</sup>								
		7	6	5	4	3	2	1	0	
GPIOIS	0=edge 1=level	X	X	X	X	X	0	X	X	X
GPIOIBE	0=single edge 1=both edges	X	X	X	X	X	0	X	X	X
GPIOEV	0=Low level, or negative edge 1=High level, or positive edge	X	X	X	X	X	1	X	X	X
GPIOIM	0=masked 1=not masked	0	0	0	0	0	1	0	0	0

a. X=Ignored (don't care bit)

## 9.3 Register Map

Table 9-3 on page 170 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
- GPIO Port F: 0x4002.5000
- GPIO Port G: 0x4002.6000

**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**, **GPIOPUR**, and **GPIODEN** registers are 0x0000.0000 for all GPIO pins, with the exception of the five JTAG/SWD pins (**PB7** and **PC[3:0]**). These five pins default to JTAG/SWD functionality. Because of this, the default reset value of these registers for GPIO Port B is 0x0000.0080 while the default reset value for Port C is 0x0000.000F.

The default register type for the **GPIOCR** register is RO for all GPIO pins, with the exception of the five JTAG/SWD pins (**PB7** and **PC[3:0]**). These five pins are currently the only GPIOs that are protected by the **GPIOCR** register. Because of this, the register type for GPIO Port B7 and GPIO Port C[3:0] is R/W.

The default reset value for the **GPIOCR** register is 0x0000.00FF for all GPIO pins, with the exception of the five JTAG/SWD pins (**PB7** and **PC[3:0]**). To ensure that the JTAG port is not accidentally programmed as a GPIO, these five pins default to non-commitable. Because of this, the default reset value of **GPIOCR** for GPIO Port B is 0x0000.007F while the default reset value of **GPIOCR** for Port C is 0x0000.00F0.

**Table 9-3. GPIO Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	GPIODATA	R/W	0x0000.0000	GPIO Data	172
0x400	GPIODIR	R/W	0x0000.0000	GPIO Direction	173
0x404	GPIOIS	R/W	0x0000.0000	GPIO Interrupt Sense	174
0x408	GPIOIBE	R/W	0x0000.0000	GPIO Interrupt Both Edges	175
0x40C	GPIOEV	R/W	0x0000.0000	GPIO Interrupt Event	176
0x410	GPIOIM	R/W	0x0000.0000	GPIO Interrupt Mask	177

Offset	Name	Type	Reset	Description	See page
0x414	GPIOISR	RO	0x0000.0000	GPIO Raw Interrupt Status	178
0x418	GPIOMIS	RO	0x0000.0000	GPIO Masked Interrupt Status	179
0x41C	GPIOICR	W1C	0x0000.0000	GPIO Interrupt Clear	180
0x420	GPIOAFSEL	R/W	-	GPIO Alternate Function Select	181
0x500	GPIODR2R	R/W	0x0000.00FF	GPIO 2-mA Drive Select	183
0x504	GPIODR4R	R/W	0x0000.0000	GPIO 4-mA Drive Select	184
0x508	GPIODR8R	R/W	0x0000.0000	GPIO 8-mA Drive Select	185
0x50C	GPIOODR	R/W	0x0000.0000	GPIO Open Drain Select	186
0x510	GPIOPUR	R/W	-	GPIO Pull-Up Select	187
0x514	GPIOPDR	R/W	0x0000.0000	GPIO Pull-Down Select	188
0x518	GPIOSLR	R/W	0x0000.0000	GPIO Slew Rate Control Select	189
0x51C	GPIODEN	R/W	-	GPIO Digital Enable	190
0x520	GPIOLOCK	R/W	0x0000.0001	GPIO Lock	191
0x524	GPIOCR	-	-	GPIO Commit	192
0xFD0	GPIOPeriphID4	RO	0x0000.0000	GPIO Peripheral Identification 4	194
0xFD4	GPIOPeriphID5	RO	0x0000.0000	GPIO Peripheral Identification 5	195
0xFD8	GPIOPeriphID6	RO	0x0000.0000	GPIO Peripheral Identification 6	196
0xFDC	GPIOPeriphID7	RO	0x0000.0000	GPIO Peripheral Identification 7	197
0xFE0	GPIOPeriphID0	RO	0x0000.0061	GPIO Peripheral Identification 0	198
0xFE4	GPIOPeriphID1	RO	0x0000.0000	GPIO Peripheral Identification 1	199
0xFE8	GPIOPeriphID2	RO	0x0000.0018	GPIO Peripheral Identification 2	200
0xFEC	GPIOPeriphID3	RO	0x0000.0001	GPIO Peripheral Identification 3	201
0xFF0	GPIOPCellID0	RO	0x0000.000D	GPIO PrimeCell Identification 0	202
0xFF4	GPIOPCellID1	RO	0x0000.00F0	GPIO PrimeCell Identification 1	203
0xFF8	GPIOPCellID2	RO	0x0000.0005	GPIO PrimeCell Identification 2	204
0xFFC	GPIOPCellID3	RO	0x0000.00B1	GPIO PrimeCell Identification 3	205

## 9.4 Register Descriptions

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

## Register 1: GPIO Data (GPIODATA), offset 0x000

The **GPIODATA** register is the data register. In software control mode, values written in the **GPIODATA** register are transferred onto the GPIO port pins if the respective pins have been configured as outputs through the **GPIO Direction (GPIODIR)** register (see page 173).

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

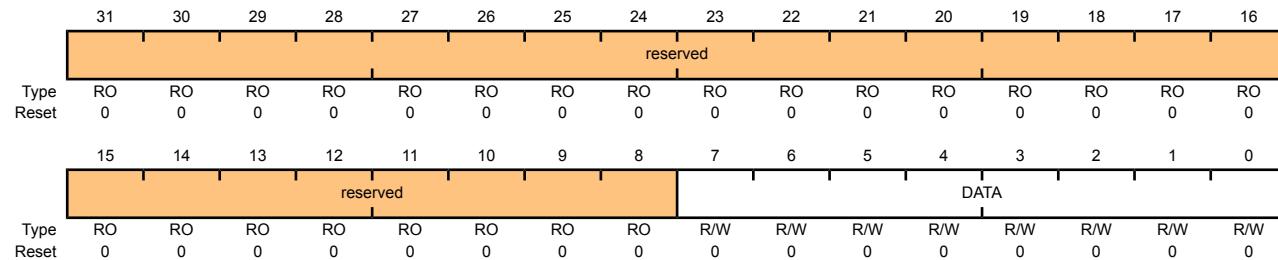
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

Offset 0x000

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x00	<p>GPIO Data</p> <p>This register is virtually mapped to 256 locations in the address space. To facilitate the reading and writing of data to these registers by independent drivers, the data read from and the data written to the registers are masked by the eight address lines <code>ipaddr[9:2]</code>. Reads from this register return its current state. Writes to this register only affect bits that are not masked by <code>ipaddr[9:2]</code> and are configured as outputs. See “Data Register Operation” on page 166 for examples of reads and writes.</p>

## 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

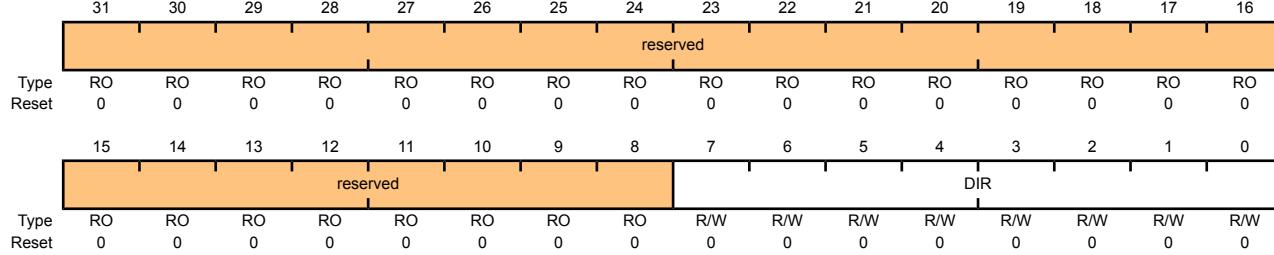
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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

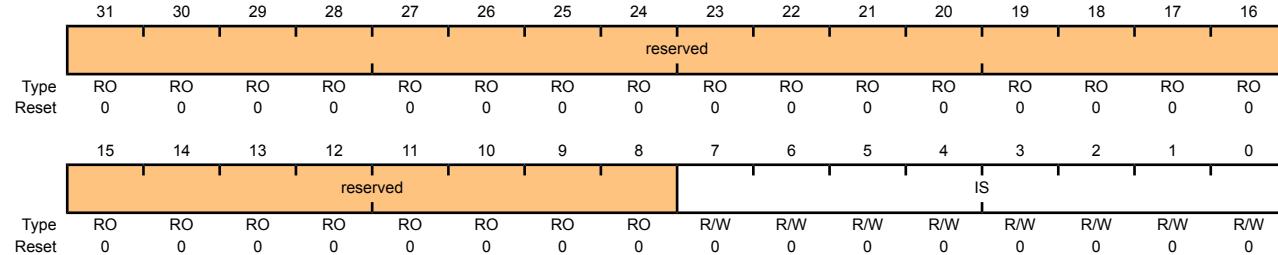
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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 174) 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 176). 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

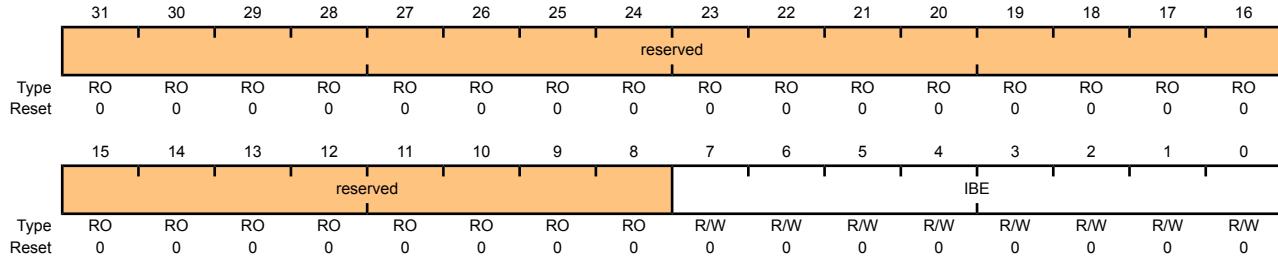
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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 176).
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 174). 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

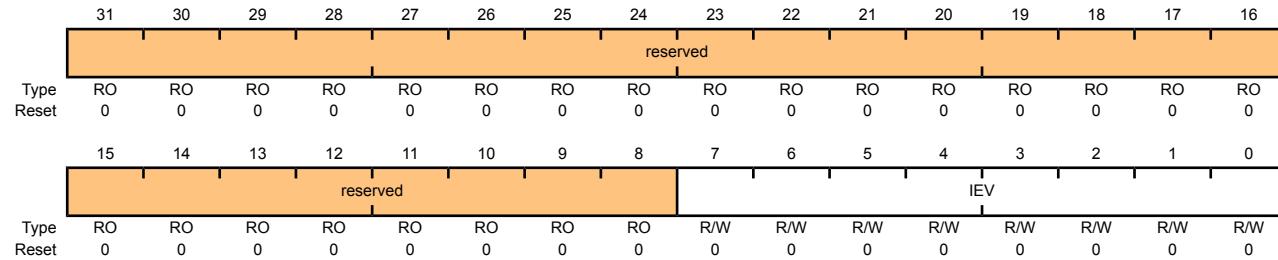
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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

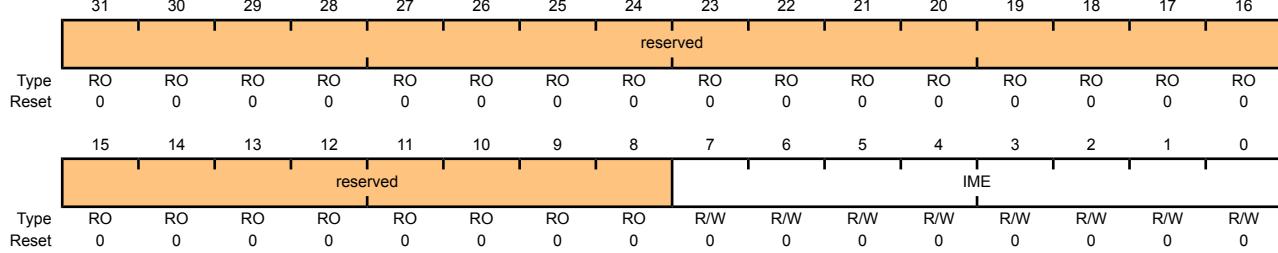
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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 177). 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

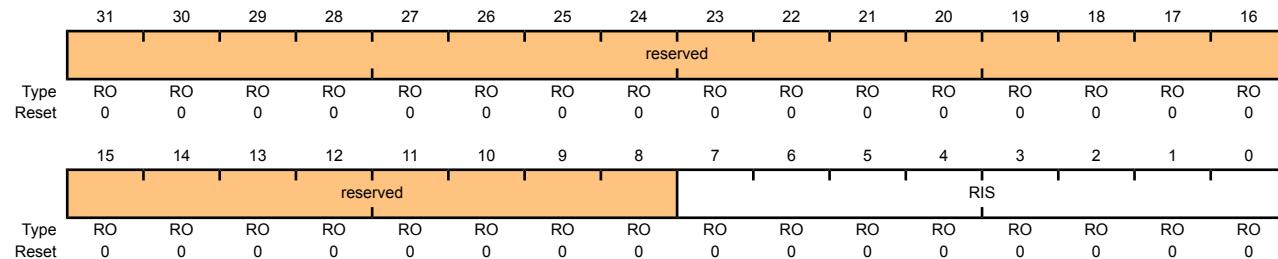
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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	GPIO Interrupt Raw Status Reflects the status of interrupt trigger condition detection on pins (raw, prior to masking).
-----	-----	----	------	--

The RIS values are defined as follows:

Value	Description
0	Corresponding pin interrupt requirements not met.
1	Corresponding pin interrupt has met requirements.

## Register 8: GPIO Masked Interrupt Status (GPIOIMIS), offset 0x418

The **GPIOIMIS** register is the masked interrupt status register. Bits read High in **GPIOIMIS** 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 (GPIOIM is set to 1), not only is an interrupt for PortB generated, but an external trigger signal is sent to the ADC. If the **ADC Event Multiplexer Select (ADCEMUX)** register is configured to use the external trigger, an ADC conversion is initiated.

If no other PortB pins are being used to generate interrupts, the ARM Integrated Nested Vectored Interrupt Controller (NVIC) Interrupt Set Enable (SETNA) register can disable the PortB interrupts and the ADC interrupt can be used to read back the converted data. Otherwise, the PortB interrupt handler needs to ignore and clear interrupts on B4, and wait for the ADC interrupt or the ADC interrupt needs to be disabled in the SETNA register and the PortB interrupt handler polls the ADC registers until the conversion is completed.

**GPIOIMIS** is the state of the interrupt after masking.

### GPIO Masked Interrupt Status (GPIOIMIS)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

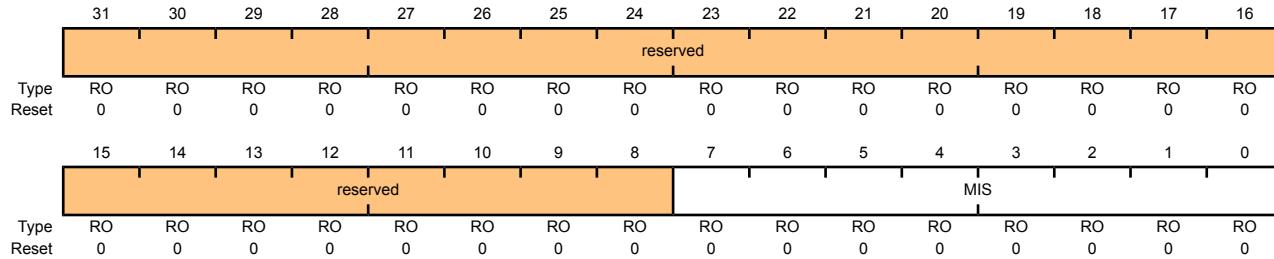
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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

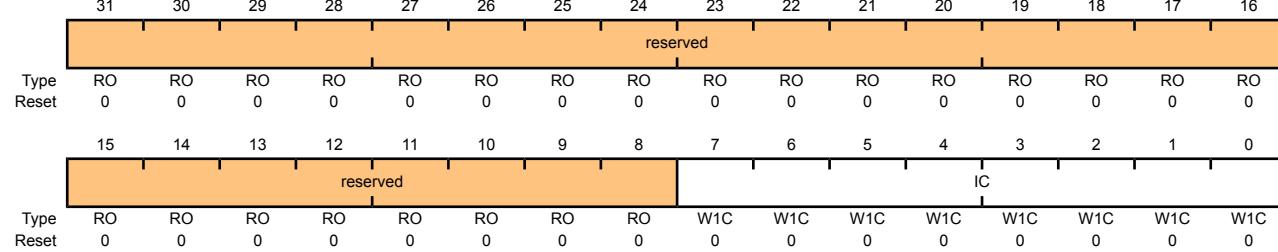
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

Offset 0x41C

Type W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IC	W1C	0x00	GPIO Interrupt Clear

The IC values are defined as follows:

#### Value Description

0 Corresponding interrupt is unaffected.

1 Corresponding interrupt is cleared.

## Register 10: GPIO Alternate Function Select (GPIOAFSEL), offset 0x420

The **GPIOAFSEL** register is the mode control select register. Writing a 1 to any bit in this register selects the hardware control for the corresponding GPIO line. All bits are cleared by a reset, therefore no GPIO line is set to hardware control by default.

The commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 181) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 191) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 192) have been set to 1.

**Important:** All GPIO pins are tri-stated by default (**GPIOAFSEL=0**, **GPIODEN=0**, **GPIOPDR=0**, and **GPIOPUR=0**), with the exception of the five JTAG/SWD pins (**PB7** and **PC[3:0]**). The JTAG/SWD pins default to their JTAG/SWD functionality (**GPIOAFSEL=1**, **GPIODEN=1** and **GPIOPUR=1**). A Power-On-Reset (**POR**) or asserting **RST** puts both groups of pins back to their default state.

**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.**

In addition, 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

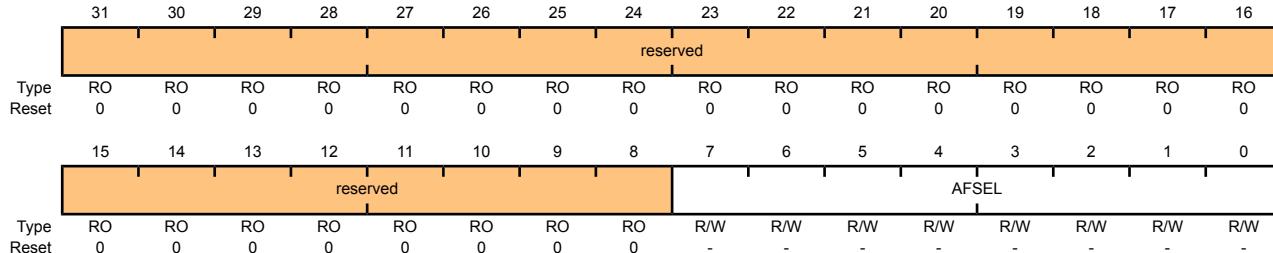
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

Offset 0x420

Type R/W, reset -



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

31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
------	----------	----	------	---

Bit/Field	Name	Type	Reset	Description
7:0	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).				
<p><b>Note:</b> The default reset value for the <b>GPIOAFSEL</b>, <b>GPIOPUR</b>, and <b>GPIODEN</b> registers are 0x0000.0000 for all GPIO pins, with the exception of the five JTAG/SWD pins (PB7 and PC[3:0]). These five pins default to JTAG/SWD functionality. Because of this, the default reset value of these registers for GPIO Port B is 0x0000.0080 while the default reset value for Port C is 0x0000.000F.</p>				

## 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

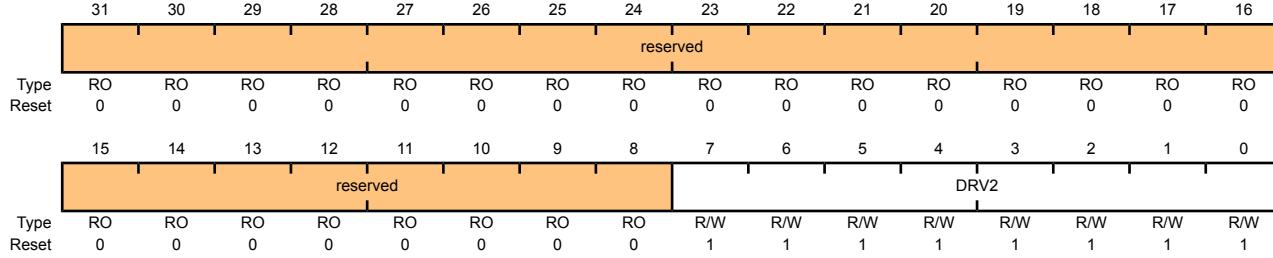
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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

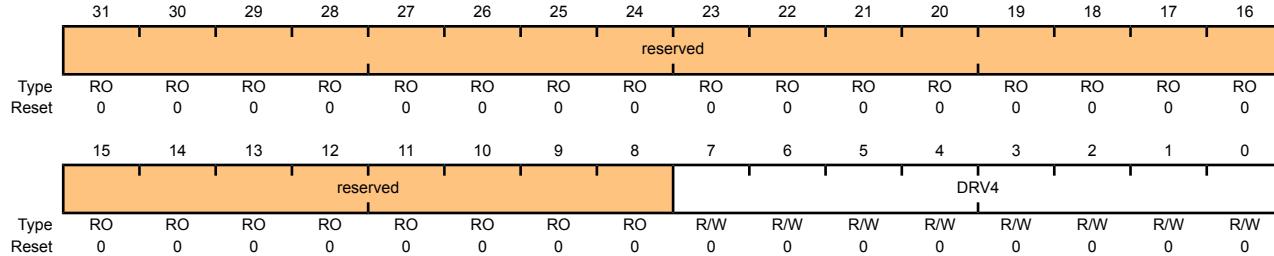
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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	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

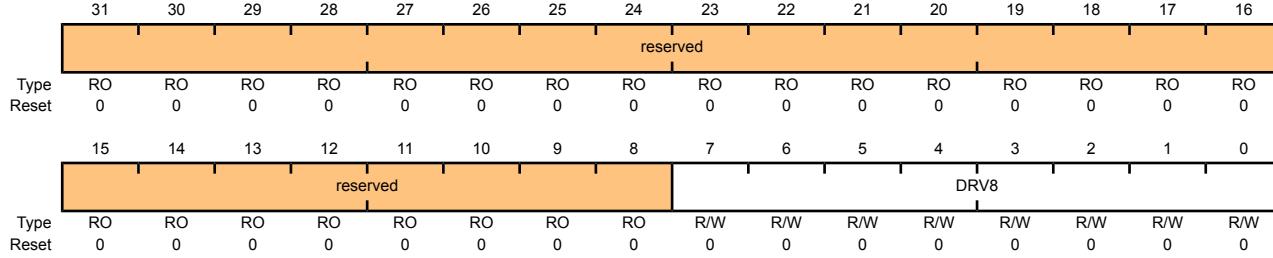
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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 Input Enable (GPIODEN)** register (see page 190). Corresponding bits in the drive strength registers (**GPIODR2R**, **GPIODR4R**, **GPIODR8R**, and **GPIOSLR**) can be set to achieve the desired rise and fall times. The GPIO acts as an open drain input if the corresponding bit in the **GPIODIR** register is set to 0; and as an open drain output when set to 1.

When using the I<sup>2</sup>C module, the **GPIO Alternate Function Select (GPIOAFSEL)** register bit for PB2 and PB3 should be set to 1 (see examples in “Initialization and Configuration” on page 168).

### 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

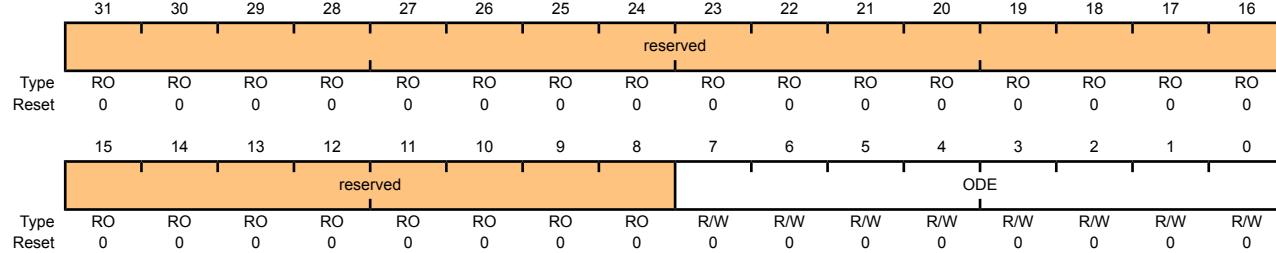
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

Offset 0x50C

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ODE	R/W	0x00	Output Pad Open Drain Enable The ODE values are defined as follows:

#### Value Description

0 Open drain configuration is disabled.

1 Open drain configuration is enabled.

## Register 15: GPIO Pull-Up Select (GPIOPUR), offset 0x510

The **GPIOPUR** register is the pull-up control register. When a bit is set to 1, it enables a weak pull-up resistor on the corresponding GPIO signal. Setting a bit in **GPIOPUR** automatically clears the corresponding bit in the **GPIO Pull-Down Select (GPIOPDR)** register (see page 188).

### 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

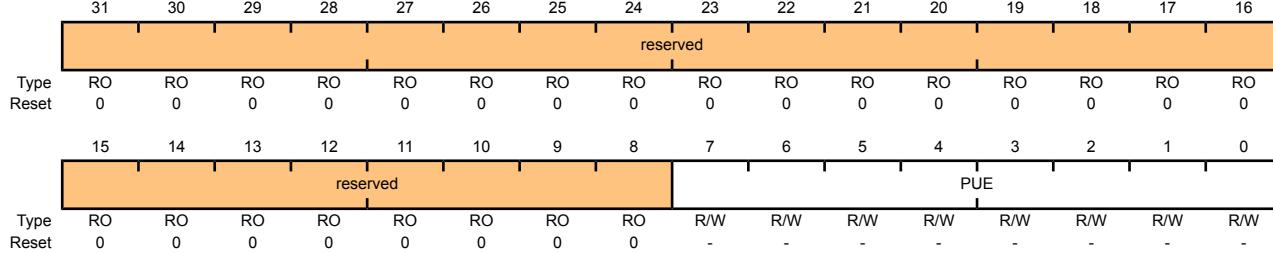
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

Offset 0x510

Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PUE	R/W	-	Pad Weak Pull-Up Enable A write of 1 to <b>GPIOPDR[n]</b> clears the corresponding <b>GPIOPUR[n]</b> enables. The change is effective on the second clock cycle after the write.

**Note:** The default reset value for the **GPIOAFSEL**, **GPIOPUR**, and **GPIODEN** registers are 0x0000.0000 for all GPIO pins, with the exception of the five JTAG/SWD pins ( $PB[7]$  and  $PC[3:0]$ ). These five pins default to JTAG/SWD functionality. Because of this, the default reset value of these registers for GPIO Port B is 0x0000.0080 while the default reset value for Port C is 0x0000.000F.

## Register 16: GPIO Pull-Down Select (GPIO\_PDR), offset 0x514

The **GPIO\_PDR** 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 **GPIO\_PDR** automatically clears the corresponding bit in the **GPIO Pull-Up Select (GPIO\_PUR)** register (see page 187).

### GPIO Pull-Down Select (GPIO\_PDR)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

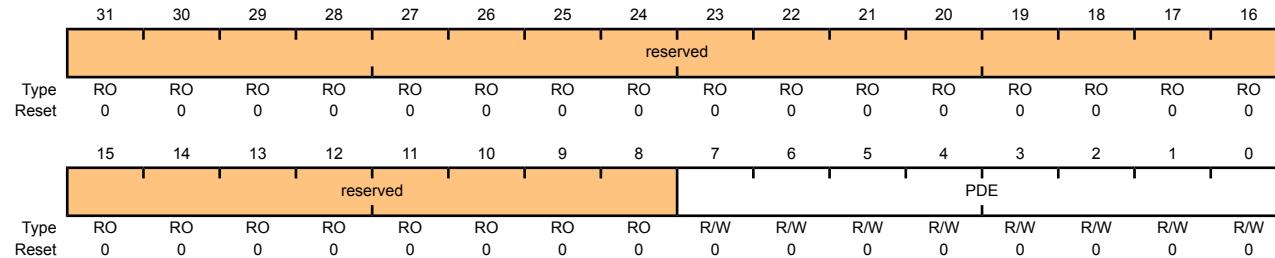
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

Offset 0x514

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PDE	R/W	0x00	Pad Weak Pull-Down Enable A write of 1 to <b>GPIO_PUR[n]</b> clears the corresponding <b>GPIO_PDR[n]</b> enables. The change is effective on the second clock cycle after the write.

## Register 17: GPIO Slew Rate Control Select (GPIOSLR), offset 0x518

The **GPIOSLR** register is the slew rate control register. Slew rate control is only available when using the 8-mA drive strength option via the **GPIO 8-mA Drive Select (GPIODR8R)** register (see page 185).

### 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

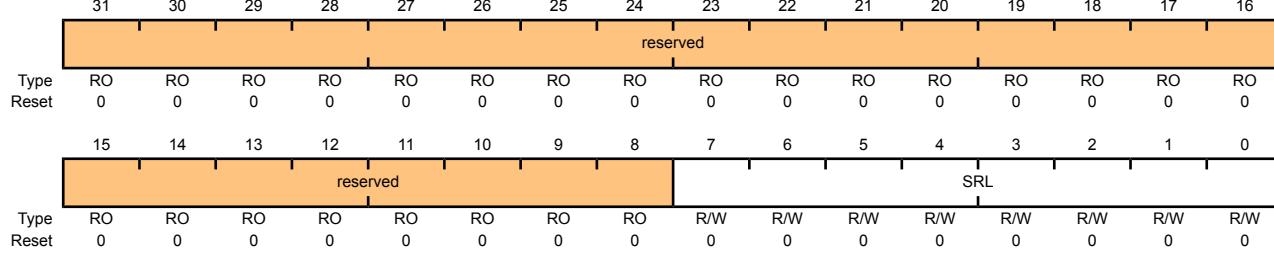
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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

The **GPIODEN** register is the digital enable register. By default, with the exception of the GPIO signals used for JTAG/SWD function, all other GPIO signals are configured out of reset to be undriven (tristate). Their digital function is disabled; they do not drive a logic value on the pin and they do not allow the pin voltage into the GPIO receiver. To use the pin in a digital function (either GPIO or alternate function), the corresponding **GPIODEN** bit must be set.

### 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

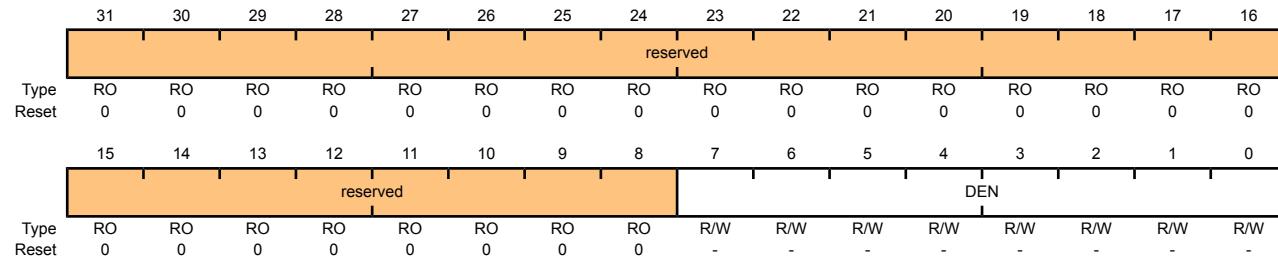
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

Offset 0x51C

Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DEN	R/W	-	Digital Enable

The **DEN** values are defined as follows:

Value	Description
0	Digital functions disabled.
1	Digital functions enabled.

**Note:** The default reset value for the **GPIOAFSEL**, **GPIOPUR**, and **GPIODEN** registers are 0x0000.0000 for all GPIO pins, with the exception of the five JTAG/SWD pins (PB7 and PC[3:0]). These five pins default to JTAG/SWD functionality. Because of this, the default reset value of these registers for GPIO Port B is 0x0000.0080 while the default reset value for Port C is 0x0000.000F.

## Register 19: GPIO Lock (GPIOLOCK), offset 0x520

The **GPIOLOCK** register enables write access to the **GPIOCR** register (see page 192). Writing 0x1ACCE551 to the **GPIOLOCK** register will unlock the **GPIOCR** register. Writing any other value to the **GPIOLOCK** register re-enables the locked state. Reading the **GPIOLOCK** register returns the lock status rather than the 32-bit value that was previously written. Therefore, when write accesses are disabled, or locked, reading the **GPIOLOCK** register returns 0x00000001. When write accesses are enabled, or unlocked, reading the **GPIOLOCK** register returns 0x00000000.

### GPIO Lock (GPIOLOCK)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

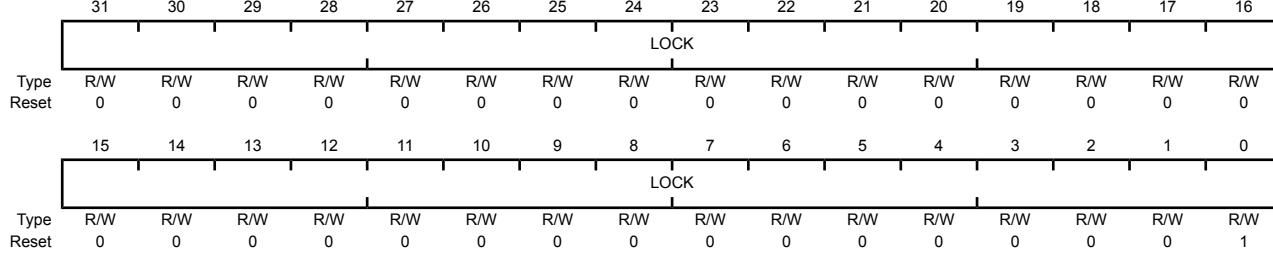
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

Offset 0x520

Type R/W, reset 0x0000.0001



Bit/Field      Name      Type      Reset      Description

31:0      LOCK      R/W      0x0000.0001      GPIO Lock

A write of the value 0x1ACCE551 unlocks the **GPIO Commit (GPIOCR)** register for write access. A write of any other value re-applies the lock, preventing any register updates. A read of this register returns the following values:

Value	Description
0x0000.0001	locked
0x0000.0000	unlocked

## Register 20: GPIO Commit (GPIOCR), offset 0x524

The **GPIOCR** register is the commit register. The value of the **GPIOCR** register determines which bits of the **GPIOAFSEL** register will be committed when a write to the **GPIOAFSEL** register is performed. If a bit in the **GPIOCR** register is a zero, the data being written to the corresponding bit in the **GPIOAFSEL** register will not be committed and will retain its previous value. If a bit in the **GPIOCR** register is a one, the data being written to the corresponding bit of the **GPIOAFSEL** register will be committed to the register and will reflect the new value.

The contents of the **GPIOCR** register can only be modified if the **GPIOLOCK** register is unlocked. Writes to the **GPIOCR** register will be ignored if the **GPIOLOCK** register is locked.

**Important:** This register is designed to prevent accidental programming of the **GPIOAFSEL** registers that control connectivity to the JTAG/SWD debug hardware. By initializing the bits of the **GPIOCR** register to 0 for PB7 and PC[3:0], the JTAG/SWD debug port can only be converted to GPIOs through a deliberate set of writes to the **GPIOLOCK**, **GPIOCR**, and **GPIOAFSEL** registers.

Because this protection is currently only implemented on the JTAG/SWD pins on PB7 and PC[3:0], all of the other bits in the **GPIOCR** registers cannot be written with 0x0. These bits are hardwired to 0x1, ensuring that it is always possible to commit new values to the **GPIOAFSEL** register bits of these other pins.

### GPIO Commit (GPIOCR)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

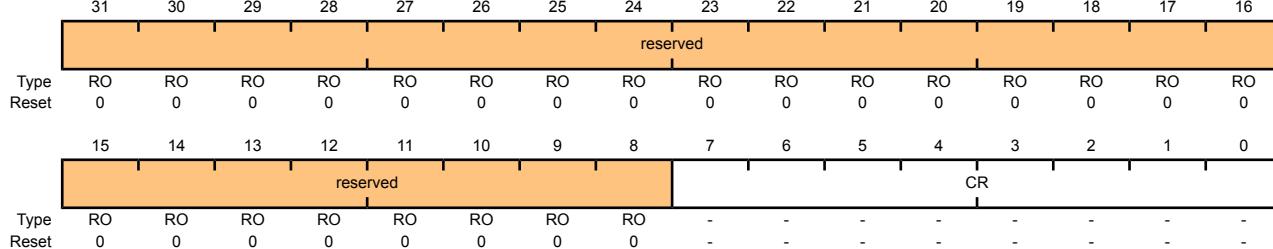
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

Offset 0x524

Type -, reset -



#### Bit/Field      Name      Type      Reset      Description

31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
------	----------	----	------	---

Bit/Field	Name	Type	Reset	Description
7:0	CR	-	-	<p>GPIO Commit</p> <p>On a bit-wise basis, any bit set allows the corresponding <code>GPIOAFSEL</code> bit to be set to its alternate function.</p> <p><b>Note:</b> The default register type for the <b>GPIOCR</b> register is RO for all GPIO pins, with the exception of the five JTAG/SWD pins (<code>PB7</code> and <code>PC[3:0]</code>). These five pins are currently the only GPIOs that are protected by the <b>GPIOCR</b> register. Because of this, the register type for GPIO Port B7 and GPIO Port C[3:0] is R/W.</p> <p>The default reset value for the <b>GPIOCR</b> register is 0x0000.00FF for all GPIO pins, with the exception of the five JTAG/SWD pins (<code>PB7</code> and <code>PC[3:0]</code>). To ensure that the JTAG port is not accidentally programmed as a GPIO, these five pins default to non-commutable. Because of this, the default reset value of <b>GPIOCR</b> for GPIO Port B is 0x0000.007F while the default reset value of <b>GPIOCR</b> for Port C is 0x0000.00F0.</p>

## Register 21: GPIO 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

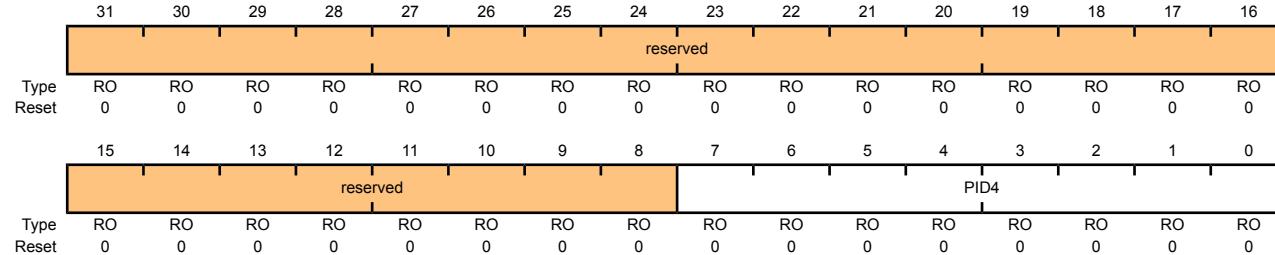
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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 22: 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

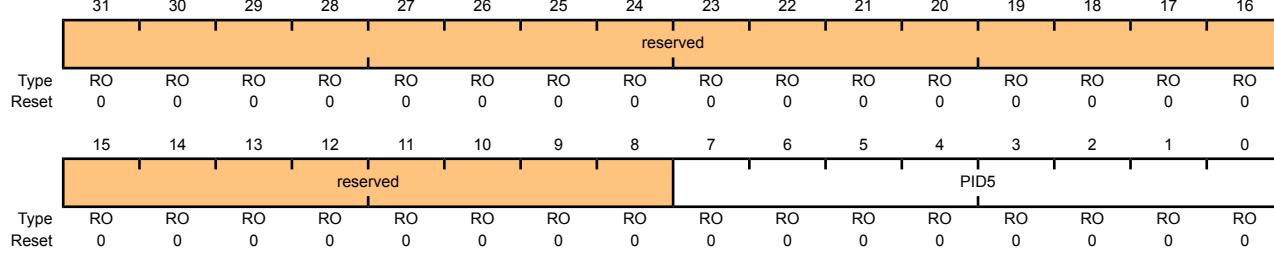
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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 23: 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

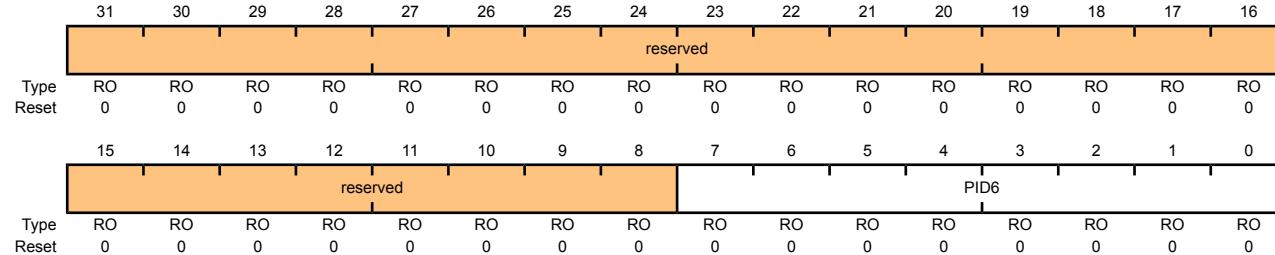
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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 24: 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

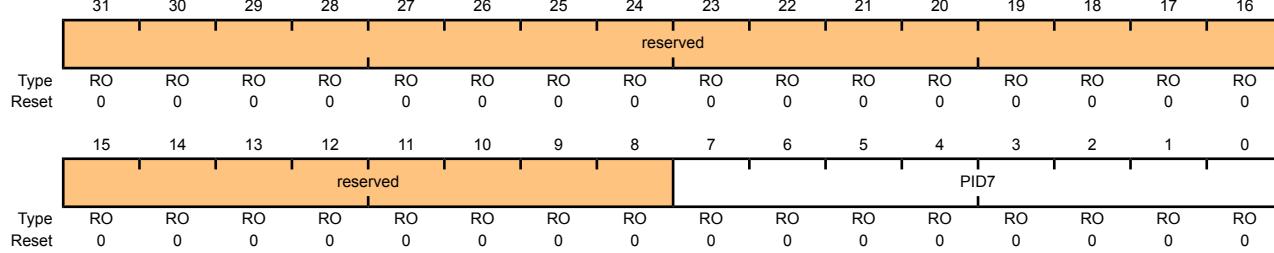
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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 25: 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

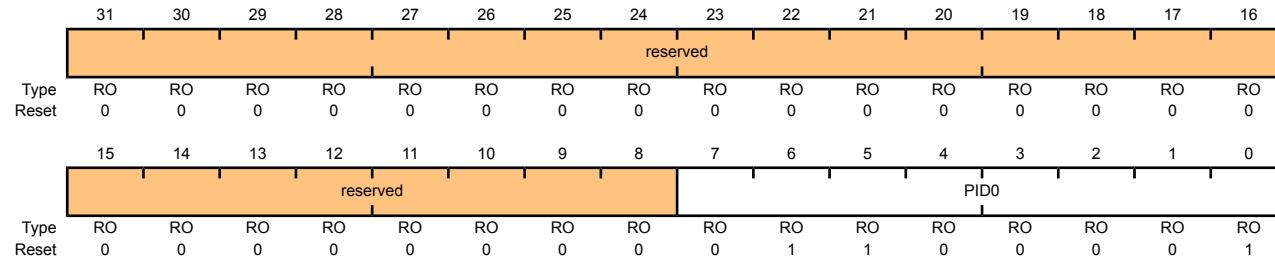
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

Offset 0xFE0

Type RO, reset 0x0000.0061



## Register 26: 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

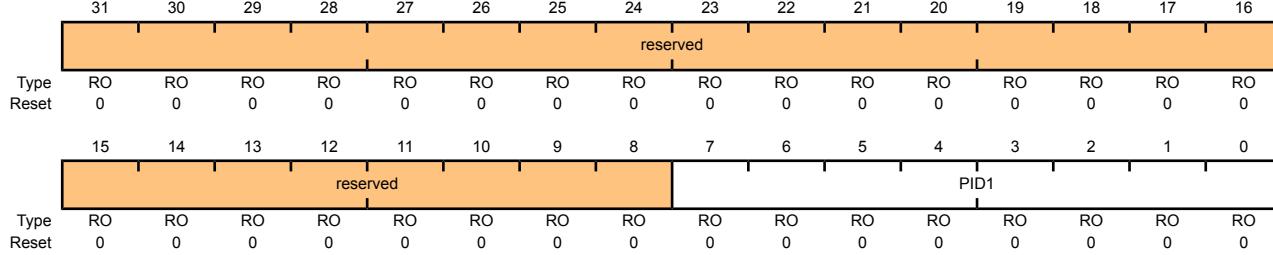
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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 27: 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

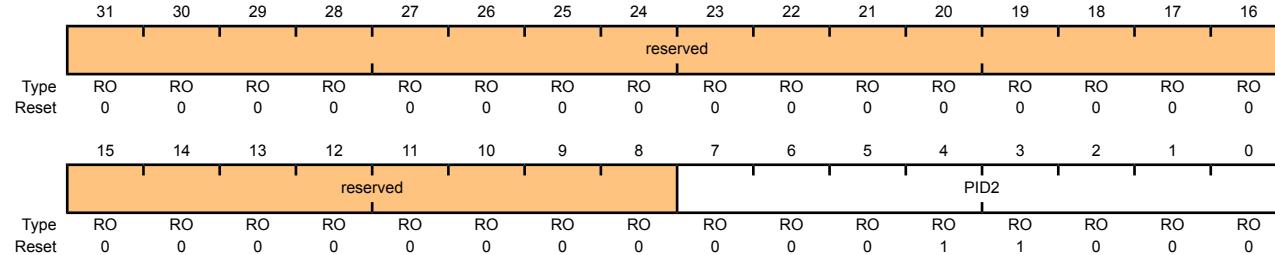
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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 28: 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

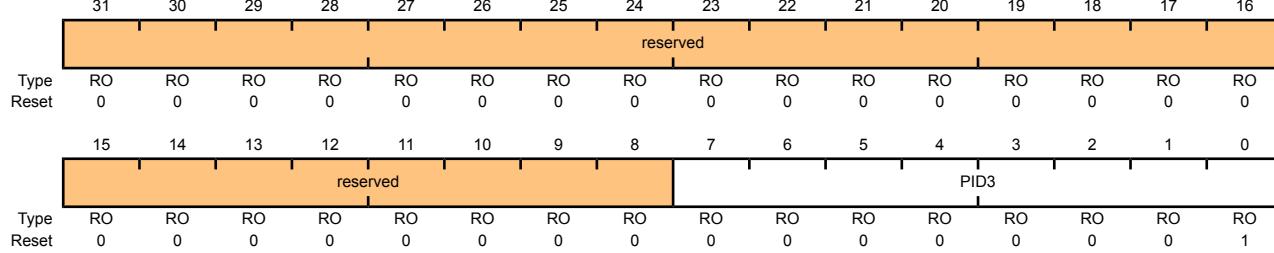
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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 29: 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

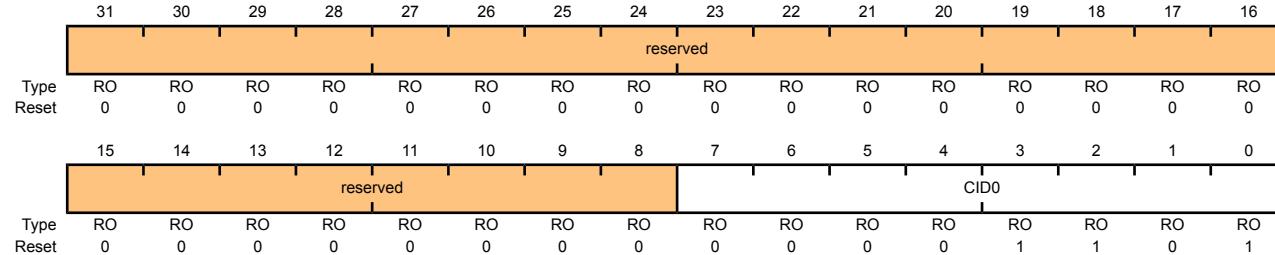
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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 30: 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

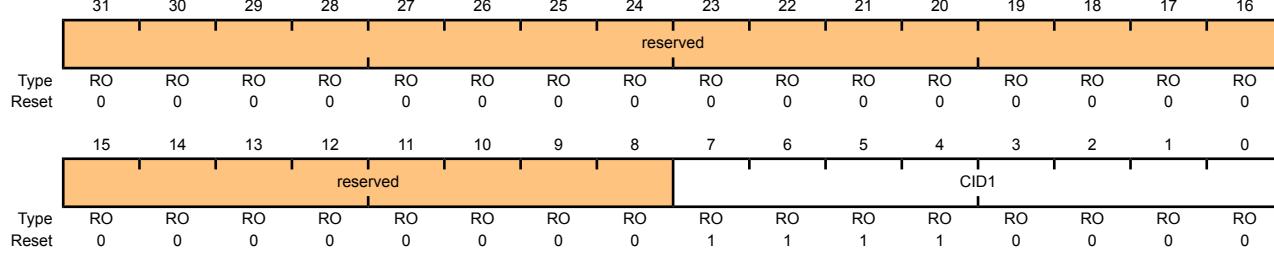
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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 31: 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

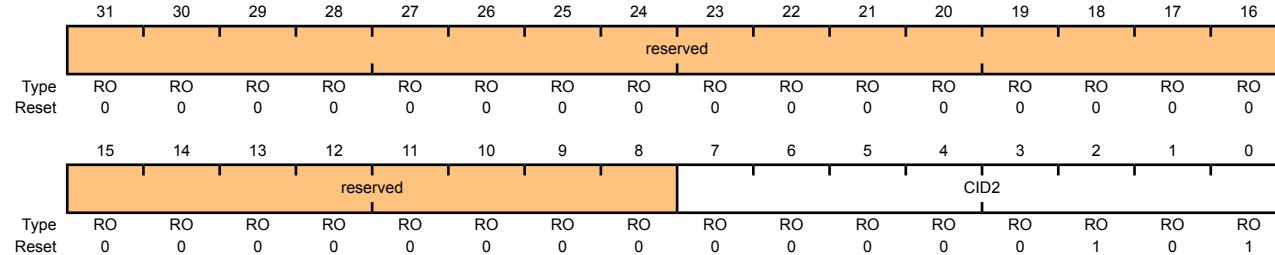
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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 32: 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

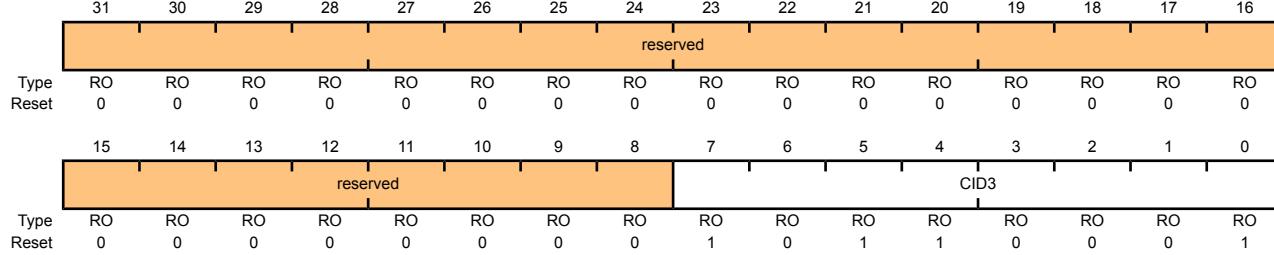
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

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.

# 10 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 four GPTM blocks (Timer0, Timer1, Timer 2, and Timer 3). Each GPTM block provides two 16-bit timers/counters (referred to as TimerA and TimerB) that can be configured to operate independently as timers or event counters, or configured to operate as one 32-bit timer or one 32-bit Real-Time Clock (RTC). Timers can also be used to trigger analog-to-digital (ADC) conversions. The trigger signals from all of the general-purpose timers are ORed together before reaching the ADC module, so only one timer should be used to trigger ADC events.

**Note:** Timer2 is an internal timer and can only be used to generate internal interrupts or trigger ADC events.

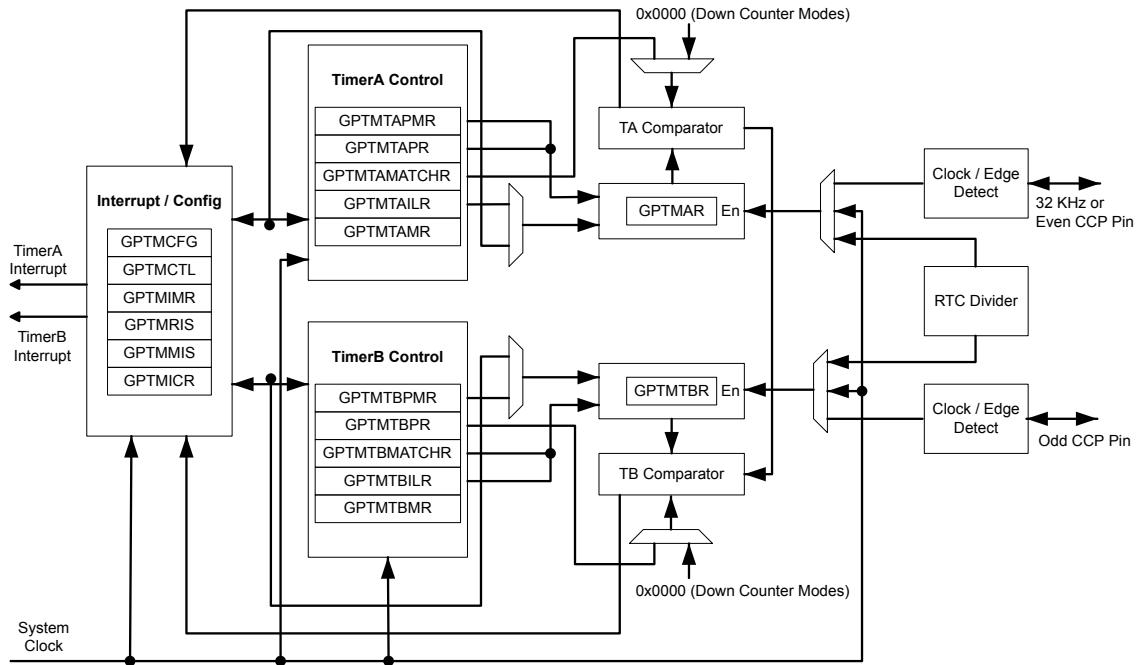
The General-Purpose Timer Module is one timing resource available on the Stellaris® microcontrollers. Other timer resources include the System Timer (SysTick) (see “System Timer (SysTick)” on page 42) and the PWM timer in the PWM module (see “PWM Timer” on page 508).

The following modes are supported:

- 32-bit Timer modes
  - Programmable one-shot timer
  - Programmable periodic timer
  - Real-Time Clock using 32.768-KHz input clock
  - Software-controlled event stalling (excluding RTC mode)
- 16-bit Timer modes
  - General-purpose timer function with an 8-bit prescaler (for one-shot and periodic modes only)
  - Programmable one-shot timer
  - Programmable periodic timer
  - Software-controlled event stalling
- 16-bit Input Capture modes
  - Input edge count capture
  - Input edge time capture
- 16-bit PWM mode
  - Simple PWM mode with software-programmable output inversion of the PWM signal

## 10.1 Block Diagram

**Note:** In Figure 10-1 on page 207, the specific CCP pins available depend on the Stellaris® device. See Table 10-1 on page 207 for the available CCPs.

**Figure 10-1. GPTM Module Block Diagram****Table 10-1. Available CCP Pins**

Timer	16-Bit Up/Down Counter	Even CCP Pin	Odd CCP Pin
Timer 0	TimerA	CCP0	-
	TimerB	-	CCP1
Timer 1	TimerA	-	-
	TimerB	-	-
Timer 2	TimerA	-	-
	TimerB	-	-
Timer 3	TimerA	-	-
	TimerB	-	-

## 10.2 Functional Description

The main components of each GPTM block are two free-running 16-bit up/down counters (referred to as TimerA and TimerB), two 16-bit match registers, 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 218), the **GPTM TimerA Mode (GPTMTAMR)** register (see page 219), and the **GPTM TimerB Mode (GPTMTBMR)** register (see page 221). 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.

### 10.2.1 GPTM Reset Conditions

After reset has been applied to the GPTM module, the module is in an inactive state, and all control registers are cleared and in their default states. Counters TimerA and TimerB are initialized to 0xFFFF, along with their corresponding load registers: the **GPTM TimerA Interval Load (GPTMTAILR)** register (see page 232) and the **GPTM TimerB Interval Load (GPTMTBILR)** register (see page 233). The prescale counters are initialized to 0x00: the **GPTM TimerA Prescale (GPTMTAPR)** register (see page 236) and the **GPTM TimerB Prescale (GPTMTBPR)** register (see page 237).

### 10.2.2 32-Bit Timer Operating Modes

This section describes the three GPTM 32-bit timer modes (One-Shot, Periodic, and RTC) and their configuration.

The GPTM is placed into 32-bit mode by writing a 0 (One-Shot/Periodic 32-bit timer mode) or a 1 (RTC mode) to the **GPTM Configuration (GPTMCFG)** register. In both configurations, certain GPTM registers are concatenated to form pseudo 32-bit registers. These registers include:

- **GPTM TimerA Interval Load (GPTMTAILR)** register [15:0], see page 232
- **GPTM TimerB Interval Load (GPTMTBILR)** register [15:0], see page 233
- **GPTM TimerA (GPTMTAR)** register [15:0], see page 240
- **GPTM TimerB (GPTMTBR)** register [15:0], see page 241

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]`

#### 10.2.2.1 32-Bit One-Shot/Periodic Timer Mode

In 32-bit one-shot and periodic timer modes, the concatenated versions of the TimerA and TimerB registers are configured as a 32-bit down-counter. The selection of one-shot or periodic mode is determined by the value written to the **TAMR** field of the **GPTM TimerA Mode (GPTMTAMR)** register (see page 219), 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 223), 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 output triggers when it reaches the 0x00000000 state. The GPTM sets the **TATORIS** bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register (see page 228), and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register (see page 230). If the time-out interrupt is enabled in the **GPTM Interrupt Mask (GPTIMR)** register (see page 226), the GPTM also sets the **TATOMIS** bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register (see page 229).

The output trigger is a one-clock-cycle pulse that is asserted when the counter hits the 0x0000.0000 state, and deasserted on the following clock cycle. It is enabled by setting the TAOTE bit in **GPTMCTL**, and can trigger SoC-level events such as ADC conversions.

If software reloads the **GPTMTAILR** register while the counter is running, the counter loads the new value on the next clock cycle and continues counting from the new value.

If the TASTALL bit in the **GPTMCTL** register is asserted, the timer freezes counting until the signal is deasserted.

### 10.2.2.2 32-Bit Real-Time Clock Timer Mode

In Real-Time Clock (RTC) mode, the concatenated versions of the TimerA and TimerB registers are configured as a 32-bit up-counter. When RTC mode is selected for the first time, the counter is loaded with a value of 0x0000.0001. All subsequent load values must be written to the **GPTM TimerA Match (GPTMTAMATCHR)** register (see page 234) by the controller.

The input clock on the CCP0, CCP2, or CCP4 pins is required to be 32.768 KHz in RTC mode. The clock signal is then divided down to a 1 Hz rate and is passed along to the input of the 32-bit counter.

When software writes the TAEN bit in the **GPTMCTL** register, the counter starts counting up from its preloaded value of 0x0000.0001. When the current count value matches the preloaded value in the **GPTMTAMATCHR** register, it rolls over to a value of 0x0000.0000 and continues counting until either a hardware reset, or it is disabled by software (clearing the TAEN bit). When a match occurs, the GPTM asserts the RTCRIS bit in **GPTMRIS**. If the RTC interrupt is enabled in **GPTIMR**, the GPTM also sets the RTCMIS bit in **GPTMISR** and generates a controller interrupt. The status flags are cleared by writing the RTCCINT bit in **GPTMICR**.

If the TASTALL and/or TBSTALL bits in the **GPTMCTL** register are set, the timer does not freeze if the RTCEN bit is set in **GPTMCTL**.

### 10.2.3 16-Bit Timer Operating Modes

The GPTM is placed into global 16-bit mode by writing a value of 0x4 to the **GPTM Configuration (GPTMCFG)** register (see page 218). 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.

#### 10.2.3.1 16-Bit One-Shot/Periodic Timer Mode

In 16-bit one-shot and periodic timer modes, the timer is configured as a 16-bit down-counter with an optional 8-bit prescaler that effectively extends the counting range of the timer to 24 bits. The selection of one-shot or periodic mode is determined by the value written to the TnMR field of the **GPTMTnMR** register. The optional prescaler is loaded into the **GPTM Timern Prescale (GPTMTnPR)** register.

When software writes the TnEN bit in the **GPTMCTL** register, the timer begins counting down from its preloaded value. Once the 0x0000 state is reached, the timer reloads its start value from **GPTMTnILR** and **GPTMTnPR** on the next cycle. If configured to be a one-shot timer, the timer stops counting and clears the TnEN bit in the **GPTMCTL** register. If configured as a periodic timer, it continues counting.

In addition to reloading the count value, the timer generates interrupts and output triggers when it reaches the 0x0000 state. The GPTM sets the TnTORIS bit in the **GPTMRIS** register, and holds it until it is cleared by writing the **GPTMICR** register. If the time-out interrupt is enabled in **GPTIMR**, the GPTM also sets the TnTOMIS bit in **GPTMISR** and generates a controller interrupt.

The output trigger is a one-clock-cycle pulse that is asserted when the counter hits the 0x0000 state, and deasserted on the following clock cycle. It is enabled by setting the TnOTE bit in the **GPTMCTL** register, and can trigger SoC-level events such as ADC conversions.

If software reloads the **GPTMTAILR** register while the counter is running, the counter loads the new value on the next clock cycle and continues counting from the new value.

If the TnSTALL bit in the **GPTMCTL** register is enabled, the timer freezes counting until the signal is deasserted.

The following example shows a variety of configurations for a 16-bit free running timer while using the prescaler. All values assume a 50-MHz clock with Tc=20 ns (clock period).

**Table 10-2. 16-Bit Timer With Prescaler Configurations**

Prescale	#Clock (T c) <sup>a</sup>	Max Time	Units
00000000	1	1.3107	μS
00000001	2	2.6214	μS
00000010	3	3.9321	μS
-----	--	--	--
11111100	254	332.9229	μS
11111110	255	334.2336	μS
11111111	256	335.5443	μS

a. Tc is the clock period.

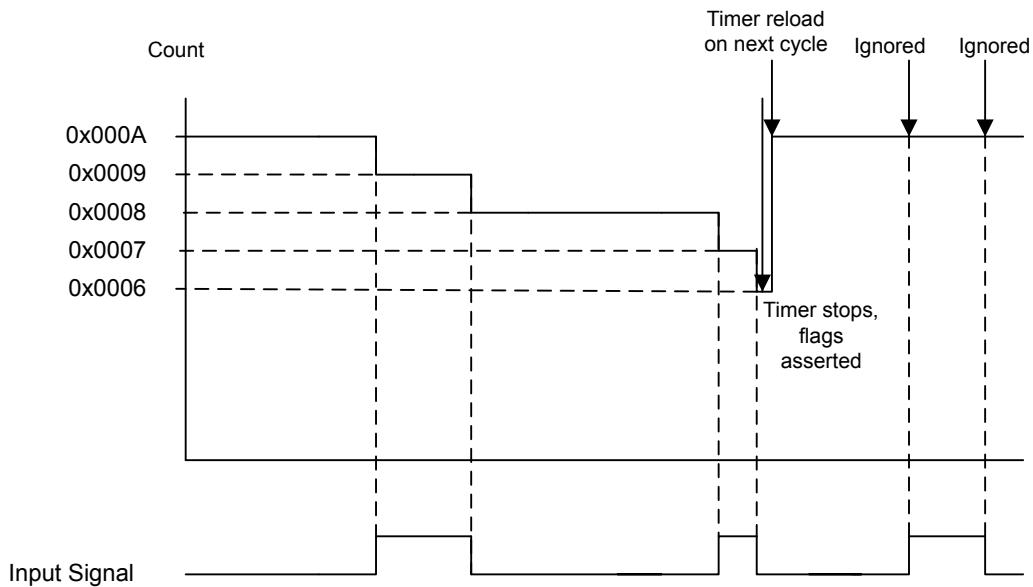
### 10.2.3.2 16-Bit Input Edge Count Mode

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 10-2 on page 211 shows how input edge count mode works. In this case, the timer start value is set to **GPTMnILR** =0x000A and the match value is set to **GPTMnMATCHR** =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 **GPTMnMR** register.

**Figure 10-2. 16-Bit Input Edge Count Mode Example**

### 10.2.3.3 16-Bit Input Edge Time Mode

**Note:** The prescaler is not available in 16-Bit Input Edge Time mode.

In Edge Time mode, the timer is configured as a free-running down-counter initialized to the value loaded in the **GPTMTnILR** register (or 0xFFFF at reset). This mode allows for event capture of both rising and falling edges. The timer is placed into Edge Time mode by setting the **TnCMR** bit in the **GPTMTnMR** register, and the type of event that the timer captures is determined by the **TnEVENT** fields of the **GPTMCnTL** register.

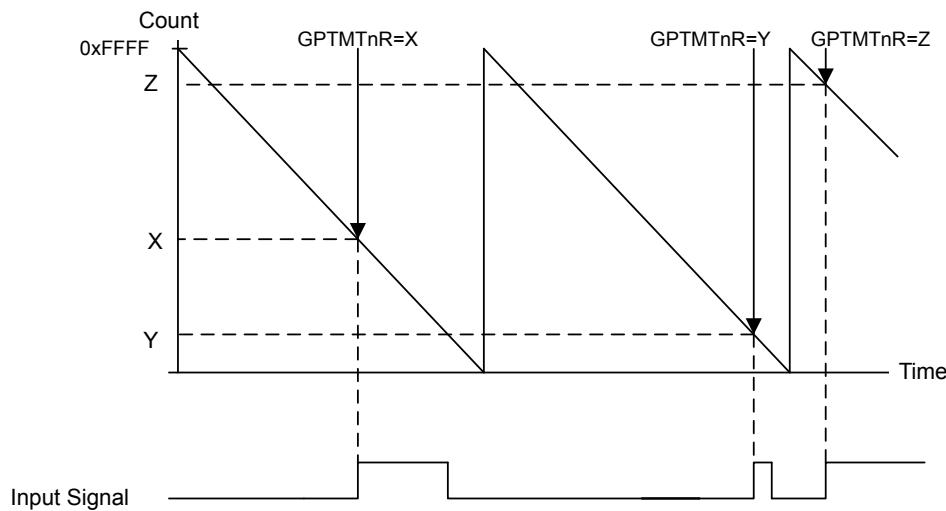
When software writes the **TnEN** bit in the **GPTMCTL** register, the timer is enabled for event capture. When the selected input event is detected, the current **Tn** counter value is captured in the **GPTMTnR** register and is available to be read by the controller. The GPTM then asserts the **CnERIS** bit (and the **CnEMIS** bit, if the interrupt is not masked).

After an event has been captured, the timer does not stop counting. It continues to count until the **TnEN** bit is cleared. When the timer reaches the 0x0000 state, it is reloaded with the value from the **GPTMnILR** register.

Figure 10-3 on page 212 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 10-3. 16-Bit Input Edge Time Mode Example



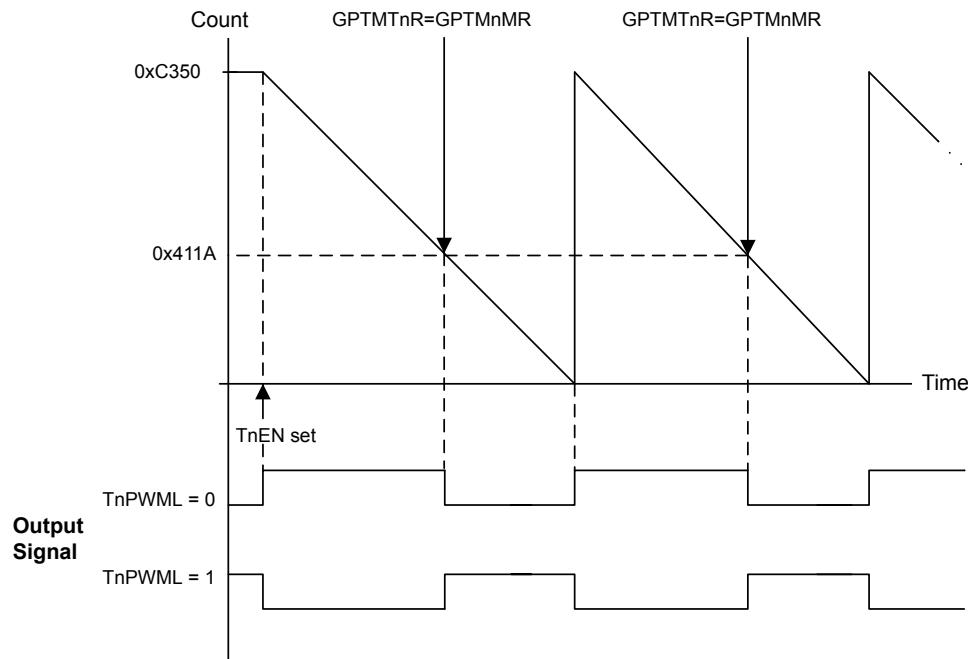
#### 10.2.3.4 16-Bit PWM Mode

The GPTM supports a simple PWM generation mode. In PWM mode, the timer is configured as a down-counter with a start value (and thus period) defined by **GPTMTnILR**. PWM mode is enabled with the **GPTMTnMR** register by setting the **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 **GPTMTnPR** if using a prescaler) 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 Timern Match Register (GPTMnMATCHR)**. Software has the capability of inverting the output PWM signal by setting the **TnPWL** bit in the **GPTMCTL** register.

Figure 10-4 on page 213 shows how to generate an output PWM with a 1-ms period and a 66% duty cycle assuming a 50-MHz input clock and **TnPWL**=0 (duty cycle would be 33% for the **TnPWL**=1 configuration). For this example, the start value is **GPTMnIRL**=0xC350 and the match value is **GPTMnMR**=0x411A.

**Figure 10-4. 16-Bit PWM Mode Example**

## 10.3 Initialization and Configuration

To use the general-purpose timers, the peripheral clock must be enabled by setting the TIMER0, TIMER1, TIMER2, and TIMER3 bits in the **RCGC1** register.

This section shows module initialization and configuration examples for each of the supported timer modes.

### 10.3.1 32-Bit One-Shot/Periodic Timer Mode

The GPTM is configured for 32-bit One-Shot and Periodic modes by the following sequence:

1. Ensure the timer is disabled (the **TAEN** bit in the **GPTMCTL** register is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x0.
3. Set the **TAMR** field in the **GPTM TimerA Mode Register (GPTMTAMR)**:
  - a. Write a value of 0x1 for One-Shot mode.
  - b. Write a value of 0x2 for Periodic mode.
4. Load the start value into the **GPTM TimerA Interval Load Register (GPTMTAILR)**.
5. If interrupts are required, set the **TATOIM** bit in the **GPTM Interrupt Mask Register (GPTMIMR)**.
6. Set the **TAEN** bit in the **GPTMCTL** register to enable the timer and start counting.

7. Poll the TATORIS bit in the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the TATOCINT bit of the **GPTM Interrupt Clear Register (GPTMICR)**.

In One-Shot mode, the timer stops counting after step 7 on page 214. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode does not stop counting after it times out.

### 10.3.2 32-Bit Real-Time Clock (RTC) Mode

To use the RTC mode, the timer must have a 32.768-KHz input signal on its CCP0, CCP2, or CCP4 pins. To enable the RTC feature, follow these steps:

1. Ensure the timer is disabled (the TAEN bit is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x1.
3. Write the desired match value to the **GPTM TimerA Match Register (GPTMTAMATCHR)**.
4. Set/clear the RTCEN bit in the **GPTM Control Register (GPTMCTL)** as desired.
5. If interrupts are required, set the RTCIM bit in the **GPTM Interrupt Mask Register (GPTMIMR)**.
6. Set the TAEN bit in the **GPTMCTL** register to enable the timer and start counting.

When the timer count equals the value in the **GPTMTAMATCHR** register, the counter is re-loaded with 0x0000.0000 and begins counting. If an interrupt is enabled, it does not have to be cleared.

### 10.3.3 16-Bit One-Shot/Periodic Timer Mode

A timer is configured for 16-bit One-Shot and Periodic modes by the following sequence:

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x4.
3. Set the TnMR field in the **GPTM Timer Mode (GPTMTnMR)** register:
  - a. Write a value of 0x1 for One-Shot mode.
  - b. Write a value of 0x2 for Periodic mode.
4. If a prescaler is to be used, write the prescale value to the **GPTM Timern Prescale Register (GPTMTnPR)**.
5. Load the start value into the **GPTM Timer Interval Load Register (GPTMTnILR)**.
6. If interrupts are required, set the TnTOIM bit in the **GPTM Interrupt Mask Register (GPTMIMR)**.
7. Set the TnEN bit in the **GPTM Control Register (GPTMCTL)** to enable the timer and start counting.
8. Poll the TnTORIS bit in the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the TnTOCINT bit of the **GPTM Interrupt Clear Register (GPTMICR)**.

In One-Shot mode, the timer stops counting after step 8 on page 214. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode does not stop counting after it times out.

#### 10.3.4 16-Bit Input Edge Count Mode

A timer is configured to Input Edge Count mode by the following sequence:

1. Ensure the timer is disabled (the `TnEN` bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x4.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the `TnCMR` field to 0x0 and the `TnMR` field to 0x3.
4. Configure the type of event(s) that the timer captures by writing the `TnEVENT` field of the **GPTM Control (GPTMCTL)** register.
5. Load the timer start value into the **GPTM Timern Interval Load (GPTMTnILR)** register.
6. Load the desired event count into the **GPTM Timern Match (GPTMTnMATCHR)** register.
7. If interrupts are required, set the `CnMIM` bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
8. Set the `TnEN` bit in the **GPTMCTL** register to enable the timer and begin waiting for edge events.
9. Poll the `CnMRIS` bit in the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the `CnMCINT` bit of the **GPTM Interrupt Clear (GPTMICR)** register.

In Input Edge Count Mode, the timer stops after the desired number of edge events has been detected. To re-enable the timer, ensure that the `TnEN` bit is cleared and repeat step 4 on page 215 through step 9 on page 215.

#### 10.3.5 16-Bit Input Edge Timing Mode

A timer is configured to Input Edge Timing mode by the following sequence:

1. Ensure the timer is disabled (the `TnEN` bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x4.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the `TnCMR` field to 0x1 and the `TnMR` field to 0x3.
4. Configure the type of event that the timer captures by writing the `TnEVENT` field of the **GPTM Control (GPTMCTL)** register.
5. Load the timer start value into the **GPTM Timern Interval Load (GPTMTnILR)** register.
6. If interrupts are required, set the `CnEIM` bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
7. Set the `TnEN` bit in the **GPTM Control (GPTMCTL)** register to enable the timer and start counting.
8. Poll the `CnERIS` bit in the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the `CnECINT` bit of the **GPTM**

**Interrupt Clear (GPTMICR)** register. The time at which the event happened can be obtained by reading the **GPTM Timern (GPTMTnR)** register.

In Input Edge Timing mode, the timer continues running after an edge event has been detected, but the timer interval can be changed at any time by writing the **GPTMTnILR** register. The change takes effect at the next cycle after the write.

### 10.3.6 16-Bit PWM Mode

A timer is configured to PWM mode using the following sequence:

1. Ensure the timer is disabled (the `TnEN` bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x4.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, set the `TnAMS` bit to 0x1, the `TnCMR` bit to 0x0, and the `TnMR` field to 0x2.
4. Configure the output state of the PWM signal (whether or not it is inverted) in the `TnEVENT` field of the **GPTM Control (GPTMCTL)** register.
5. Load the timer start value into the **GPTM Timern Interval Load (GPTMTnILR)** register.
6. Load the **GPTM Timern Match (GPTMTnMATCHR)** register with the desired value.
7. If a prescaler is going to be used, configure the **GPTM Timern Prescale (GPTMTnPR)** register and the **GPTM Timern Prescale Match (GPTMTnPMR)** register.
8. 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.

## 10.4 Register Map

Table 10-3 on page 216 lists the GPTM registers. The offset listed is a hexadecimal increment to the register's address, relative to that timer's base address:

- Timer0: 0x4003.0000
- Timer1: 0x4003.1000
- Timer2: 0x4003.2000
- Timer3: 0x4003.3000

**Table 10-3. Timers Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	GPTMCFG	R/W	0x0000.0000	GPTM Configuration	218
0x004	GPTMTAMR	R/W	0x0000.0000	GPTM TimerA Mode	219

Offset	Name	Type	Reset	Description	See page
0x008	GPTMTBMR	R/W	0x0000.0000	GPTM TimerB Mode	221
0x00C	GPTMCTL	R/W	0x0000.0000	GPTM Control	223
0x018	GPTMIMR	R/W	0x0000.0000	GPTM Interrupt Mask	226
0x01C	GPTMRIS	RO	0x0000.0000	GPTM Raw Interrupt Status	228
0x020	GPTMMIS	RO	0x0000.0000	GPTM Masked Interrupt Status	229
0x024	GPTMICR	W1C	0x0000.0000	GPTM Interrupt Clear	230
0x028	GPTMTAILR	R/W	0x0000.FFFF (16-bit mode) 0xFFFF.FFFF (32-bit mode)	GPTM TimerA Interval Load	232
0x02C	GPTMTBILR	R/W	0x0000.FFFF	GPTM TimerB Interval Load	233
0x030	GPTMTAMATCHR	R/W	0x0000.FFFF (16-bit mode) 0xFFFF.FFFF (32-bit mode)	GPTM TimerA Match	234
0x034	GPTMTBMATCHR	R/W	0x0000.FFFF	GPTM TimerB Match	235
0x038	GPTMTAPR	R/W	0x0000.0000	GPTM TimerA Prescale	236
0x03C	GPTMTBPR	R/W	0x0000.0000	GPTM TimerB Prescale	237
0x040	GPTMTAPMR	R/W	0x0000.0000	GPTM TimerA Prescale Match	238
0x044	GPTMTBPMR	R/W	0x0000.0000	GPTM TimerB Prescale Match	239
0x048	GPTMTAR	RO	0x0000.FFFF (16-bit mode) 0xFFFF.FFFF (32-bit mode)	GPTM TimerA	240
0x04C	GPTMTBR	RO	0x0000.FFFF	GPTM TimerB	241

## 10.5 Register Descriptions

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

## Register 1: GPTM Configuration (GPTMCFG), offset 0x000

This register configures the global operation of the GPTM module. The value written to this register determines whether the GPTM is in 32- or 16-bit mode.

### GPTM Configuration (GPTMCFG)

Timer0 base: 0x4003.0000

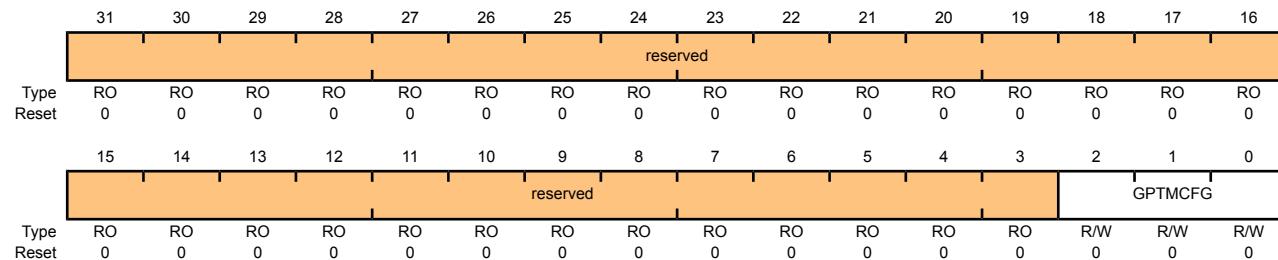
Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Timer3 base: 0x4003.3000

Offset 0x000

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	GPTMCFG	R/W	0x0	GPTM Configuration

The GPTMCFG values are defined as follows:

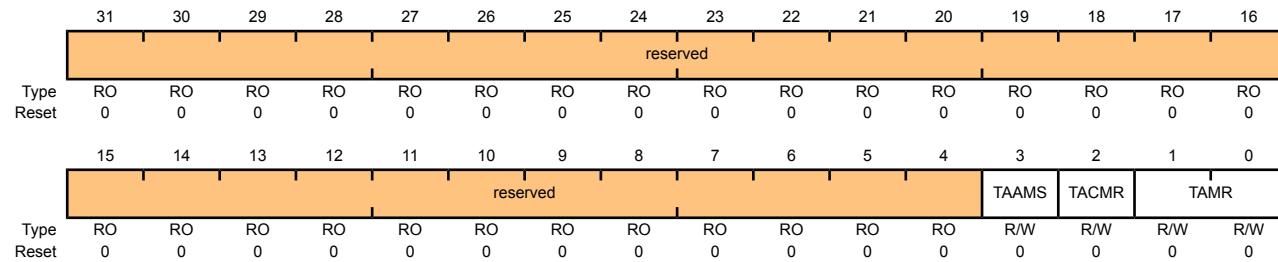
Value	Description
0x0	32-bit timer configuration.
0x1	32-bit real-time clock (RTC) counter configuration.
0x2	Reserved.
0x3	Reserved.
0x4-0x7	16-bit timer configuration; function is controlled by bits 1:0 of <b>GPTMTAMR</b> and <b>GPTMTBMR</b> .

## Register 2: GPTM TimerA Mode (GPTMTAMR), offset 0x004

This register configures the GPTM based on the configuration selected in the **GPTMCFG** register. When in 16-bit PWM mode, set the **TAAMS** bit to 0x1, the **TACMR** bit to 0x0, and the **TAMR** field to 0x2.

### GPTM TimerA Mode (GPTMTAMR)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x004  
 Type R/W, reset 0x0000.0000



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.  <b>Note:</b> 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.

---

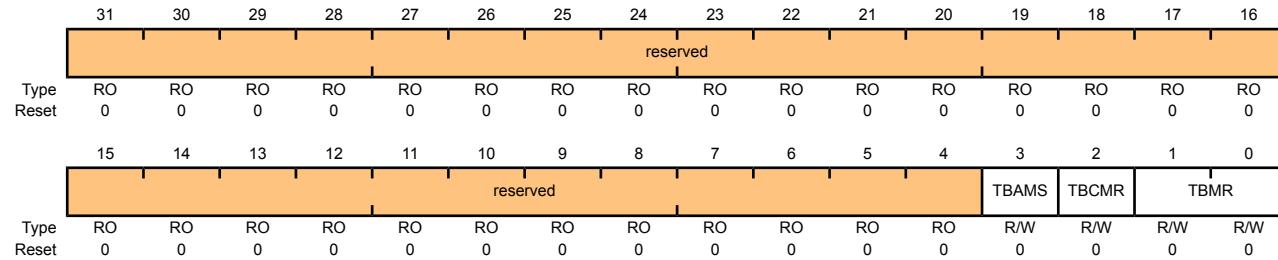
Bit/Field	Name	Type	Reset	Description										
1:0	TAMR	R/W	0x0	GPTM TimerA Mode The TAMR values are defined as follows: <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0x0</td><td>Reserved.</td></tr><tr><td>0x1</td><td>One-Shot Timer mode.</td></tr><tr><td>0x2</td><td>Periodic 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  
 Timer3 base: 0x4003.3000  
 Offset 0x008  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TBAMS	R/W	0	GPTM TimerB Alternate Mode Select  The <b>TBAMS</b> values are defined as follows:  Value Description 0 Capture mode is enabled. 1 PWM mode is enabled.  <b>Note:</b> To enable PWM mode, you must also clear the <b>TBCMR</b> bit and set the <b>TBMR</b> field to 0x2.
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

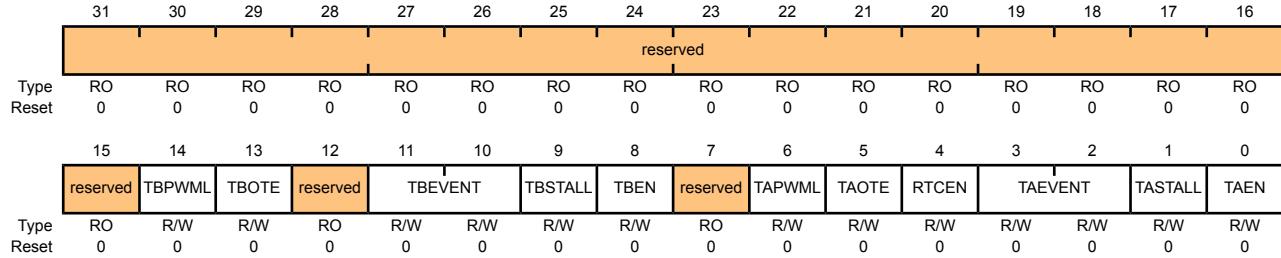
Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Timer3 base: 0x4003.3000

Offset 0x00C

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:15	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	TBPWML	R/W	0	GPTM TimerB PWM Output Level  The <b>TBPWML</b> values are defined as follows:  Value Description 0 Output is unaffected. 1 Output is inverted.
13	TBOTE	R/W	0	GPTM TimerB Output Trigger Enable  The <b>TBOTE</b> values are defined as follows:  Value Description 0 The output TimerB trigger is disabled. 1 The output TimerB trigger is enabled.
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
11:10	TBEVENT	R/W	0x0	GPTM TimerB Event Mode The TBEVENT values are defined as follows:  Value Description 0x0 Positive edge. 0x1 Negative edge. 0x2 Reserved 0x3 Both edges.
9	TBSTALL	R/W	0	GPTM TimerB Stall Enable The TBSTALL values are defined as follows:  Value Description 0 TimerB stalling is disabled. 1 TimerB stalling is enabled.
8	TBEN	R/W	0	GPTM TimerB Enable The TBEN values are defined as follows:  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	GPTM TimerA PWM Output Level The TAPWML values are defined as follows:  Value Description 0 Output is unaffected. 1 Output is inverted.
5	TAOTE	R/W	0	GPTM TimerA Output Trigger Enable The TAOTE values are defined as follows:  Value Description 0 The output TimerA trigger is disabled. 1 The output TimerA trigger is enabled.

Bit/Field	Name	Type	Reset	Description
4	RTCEN	R/W	0	GPTM RTC Enable The RTCEN values are defined as follows:  Value Description 0 RTC counting is disabled. 1 RTC counting is enabled.
3:2	TAEVENT	R/W	0x0	GPTM TimerA Event Mode The TAEVENT values are defined as follows:  Value Description 0x0 Positive edge. 0x1 Negative edge. 0x2 Reserved 0x3 Both edges.
1	TASTALL	R/W	0	GPTM TimerA Stall Enable The TASTALL values are defined as follows:  Value Description 0 TimerA stalling is disabled. 1 TimerA stalling is enabled.
0	TAEN	R/W	0	GPTM TimerA Enable The TAEN values are defined as follows:  Value Description 0 TimerA is disabled. 1 TimerA is enabled and begins counting or the capture logic is enabled based on the GPTMCFG register.

## Register 5: GPTM Interrupt Mask (GPTMIMR), offset 0x018

This register allows software to enable/disable GPTM controller-level interrupts. Writing a 1 enables the interrupt, while writing a 0 disables it.

### GPTM Interrupt Mask (GPTMIMR)

Timer0 base: 0x4003.0000

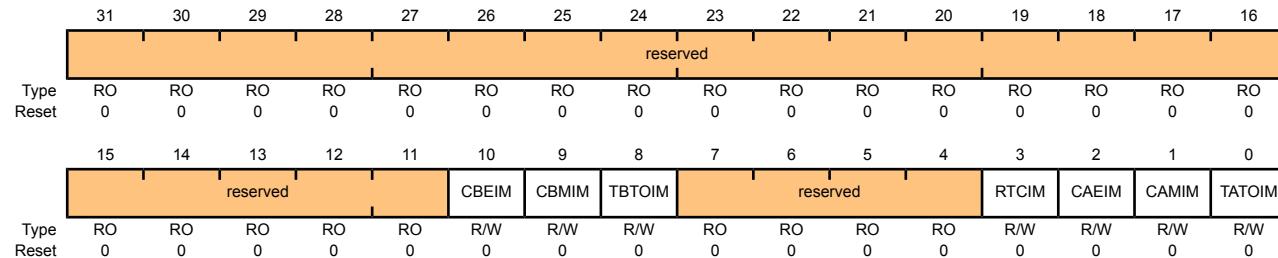
Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Timer3 base: 0x4003.3000

Offset 0x018

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	CBEIM	R/W	0	GPTM CaptureB Event Interrupt Mask The CBEIM values are defined as follows:  Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
9	CBMIM	R/W	0	GPTM CaptureB Match Interrupt Mask The CBMIM values are defined as follows:  Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
8	TBTOIM	R/W	0	GPTM TimerB Time-Out Interrupt Mask The TBTOIM values are defined as follows:  Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
7:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
3	RTCIM	R/W	0	GPTM RTC Interrupt Mask The <b>RTCIM</b> 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 <b>CAEIM</b> 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 <b>CAMIM</b> 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 <b>TATOIM</b> 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

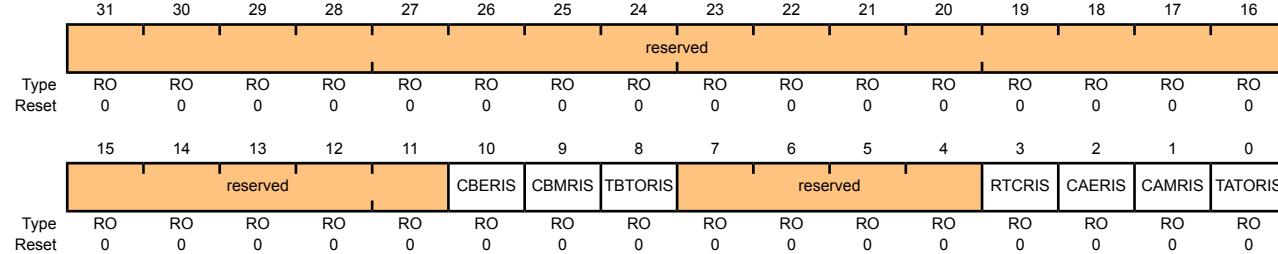
Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Timer3 base: 0x4003.3000

Offset 0x01C

Type RO, reset 0x0000.0000



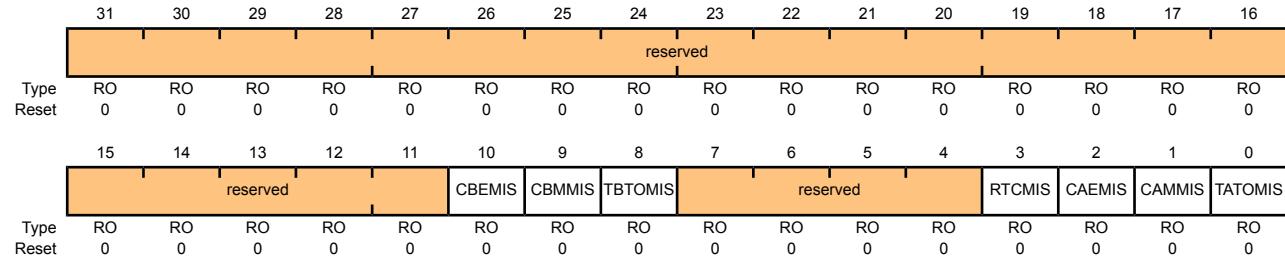
Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	CBERIS	RO	0	GPTM CaptureB Event Raw Interrupt This is the CaptureB Event interrupt status prior to masking.
9	CBMRIS	RO	0	GPTM CaptureB Match Raw Interrupt This is the CaptureB Match interrupt status prior to masking.
8	TBTORIS	RO	0	GPTM TimerB Time-Out Raw Interrupt This is the TimerB time-out interrupt status prior to masking.
7:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	RTCRIS	RO	0	GPTM RTC Raw Interrupt This is the RTC Event interrupt status prior to masking.
2	CAERIS	RO	0	GPTM CaptureA Event Raw Interrupt This is the CaptureA Event interrupt status prior to masking.
1	CAMRIS	RO	0	GPTM CaptureA Match Raw Interrupt This is the CaptureA Match interrupt status prior to masking.
0	TATORIS	RO	0	GPTM TimerA Time-Out Raw Interrupt This 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  
 Timer3 base: 0x4003.3000  
 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

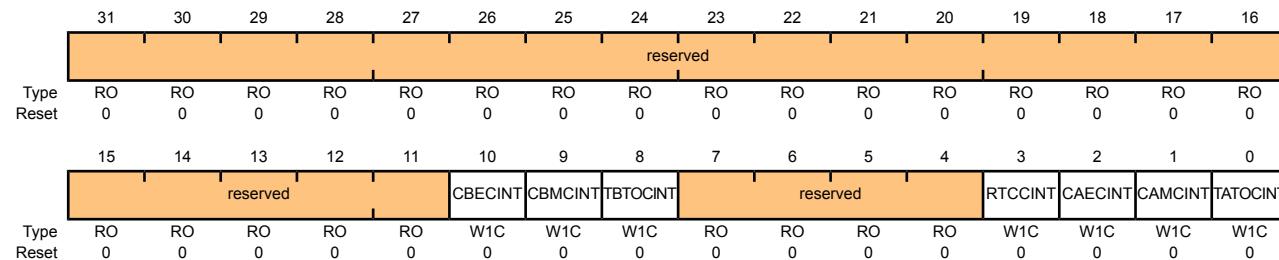
Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Timer3 base: 0x4003.3000

Offset 0x024

Type W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	CBECINT	W1C	0	GPTM CaptureB Event Interrupt Clear  The 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	TBTOCINT	W1C	0	GPTM TimerB Time-Out Interrupt Clear  The TBTOCINT 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 Raw Interrupt This is the CaptureA match interrupt status after masking.
0	TATOCINT	W1C	0	GPTM TimerA Time-Out Raw Interrupt 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

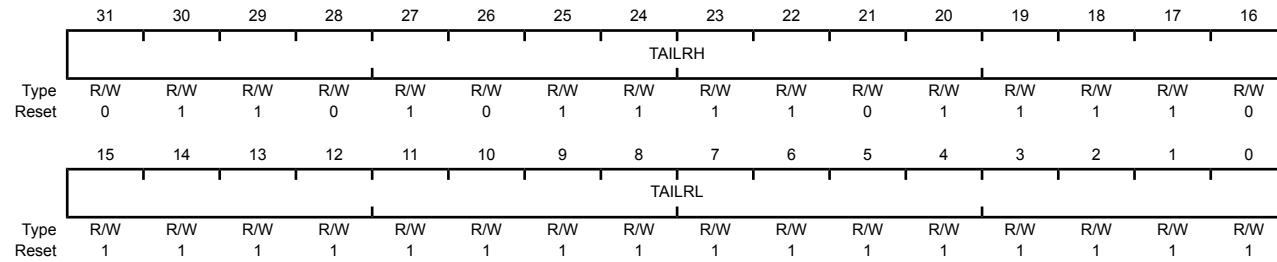
Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Timer3 base: 0x4003.3000

Offset 0x028

Type R/W, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)



Bit/Field	Name	Type	Reset	Description
31:16	TAILRH	R/W	0xFFFF (32-bit mode) 0x0000 (16-bit mode)	GPTM TimerA Interval Load Register High  When configured for 32-bit mode via the <b>GPTMCFG</b> register, the <b>GPTM TimerB Interval Load (GPTMTBILR)</b> register loads this value on a write. A read returns the current value of <b>GPTMTBILR</b> .  In 16-bit mode, this field reads as 0 and does not have an effect on the state of <b>GPTMTBILR</b> .
15:0	TAILRL	R/W	0xFFFF	GPTM TimerA Interval Load Register Low  For both 16- and 32-bit modes, writing this field loads the counter for TimerA. A read returns the current value of <b>GPTMTAILR</b> .

## Register 10: GPTM TimerB Interval Load (GPTMTBILR), offset 0x02C

This register is used to load the starting count value into TimerB. When the GPTM is configured to a 32-bit mode, **GPTMTBILR** returns the current value of TimerB and ignores writes.

### GPTM TimerB Interval Load (GPTMTBILR)

Timer0 base: 0x4003.0000

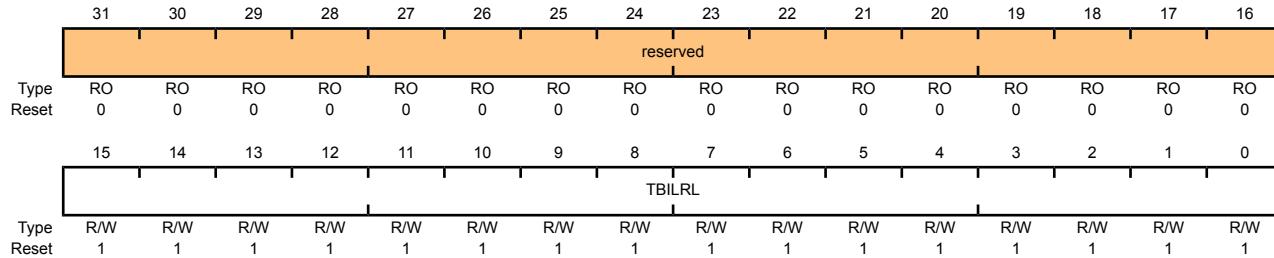
Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Timer3 base: 0x4003.3000

Offset 0x02C

Type R/W, reset 0x0000.FFFF



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

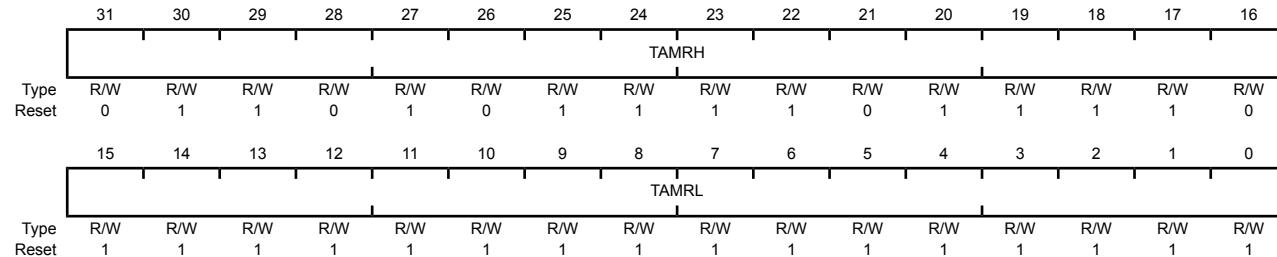
Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Timer3 base: 0x4003.3000

Offset 0x030

Type R/W, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)



Bit/Field	Name	Type	Reset	Description
31:16	TAMRH	R/W	0xFFFF (32-bit mode) 0x0000 (16-bit mode)	GPTM TimerA Match Register High  When configured for 32-bit Real-Time Clock (RTC) mode via the <b>GPTMCFG</b> register, this value is compared to the upper half of <b>GPTMTAR</b> , to determine match events.  In 16-bit mode, this field reads as 0 and does not have an effect on the state of <b>GPTMTBMATCHR</b> .
15:0	TAMRL	R/W	0xFFFF	GPTM TimerA Match Register Low  When configured for 32-bit Real-Time Clock (RTC) mode via the <b>GPTMCFG</b> register, this value is compared to the lower half of <b>GPTMTAR</b> , to determine match events.  When configured for PWM mode, this value along with <b>GPTMTAILR</b> , determines the duty cycle of the output PWM signal.  When configured for Edge Count mode, this value along with <b>GPTMTAILR</b> , determines how many edge events are counted. The total number of edge events counted is equal to the value in <b>GPTMTAILR</b> minus this value.

## Register 12: GPTM TimerB Match (GPTMTBMATCHR), offset 0x034

This register is used in 32-bit Real-Time Clock mode and 16-bit PWM and Input Edge Count modes.

### GPTM TimerB Match (GPTMTBMATCHR)

Timer0 base: 0x4003.0000

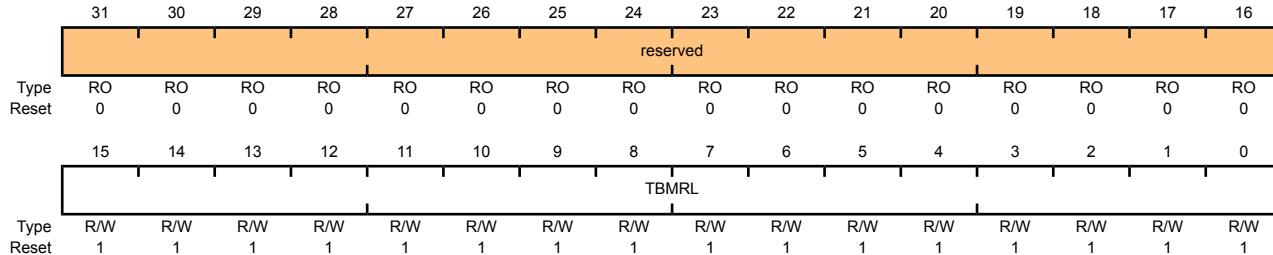
Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Timer3 base: 0x4003.3000

Offset 0x034

Type R/W, reset 0x0000.FFFF



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TBMRL	R/W	0xFFFF	<p>GPTM TimerB Match Register Low</p> <p>When configured for PWM mode, this value along with <b>GPTMTBILR</b>, determines the duty cycle of the output PWM signal.</p> <p>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.</p>

## Register 13: GPTM TimerA Prescale (GPTMTAPR), offset 0x038

This register allows software to extend the range of the 16-bit timers when operating in one-shot or periodic mode.

### GPTM TimerA Prescale (GPTMTAPR)

Timer0 base: 0x4003.0000

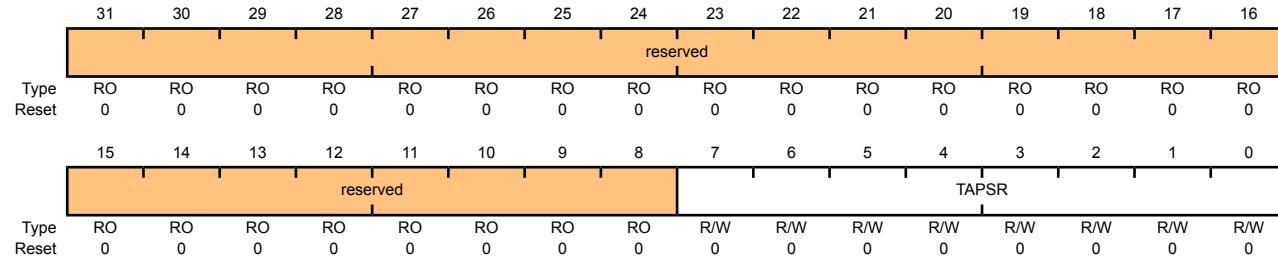
Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Timer3 base: 0x4003.3000

Offset 0x038

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TAPSR	R/W	0x00	<p>GPTM TimerA Prescale</p> <p>The register loads this value on a write. A read returns the current value of the register.</p>

Refer to Table 10-2 on page 210 for more details and an example.

## Register 14: GPTM TimerB Prescale (GPTMTBPR), offset 0x03C

This register allows software to extend the range of the 16-bit timers when operating in one-shot or periodic mode.

### GPTM TimerB Prescale (GPTMTBPR)

Timer0 base: 0x4003.0000

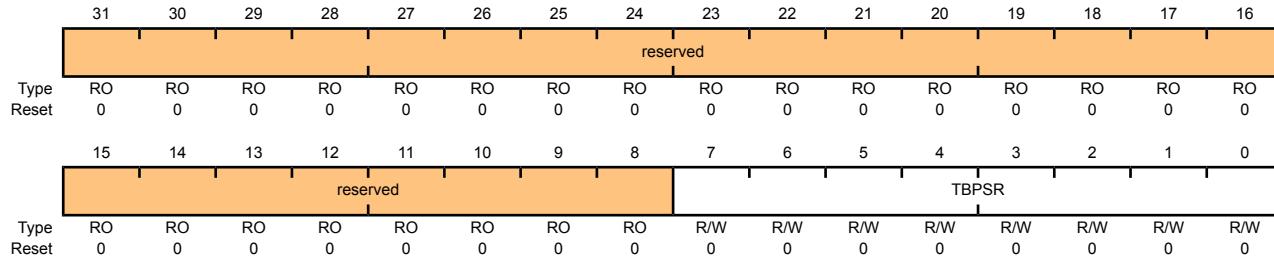
Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Timer3 base: 0x4003.3000

Offset 0x03C

Type R/W, reset 0x0000.0000



Refer to Table 10-2 on page 210 for more details and an example.

## 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

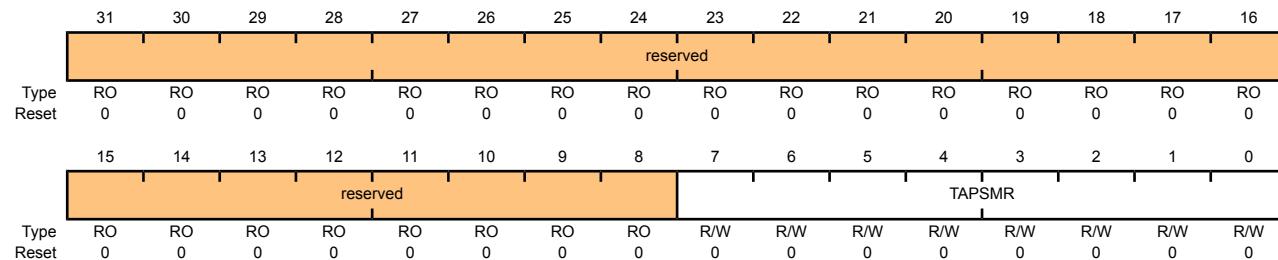
Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Timer3 base: 0x4003.3000

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

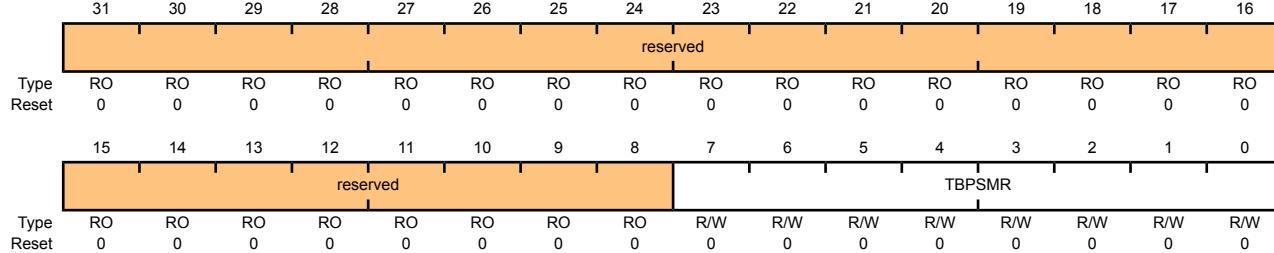
Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Timer3 base: 0x4003.3000

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 time at which the last edge event took place.

### GPTM TimerA (GPTMTAR)

Timer0 base: 0x4003.0000

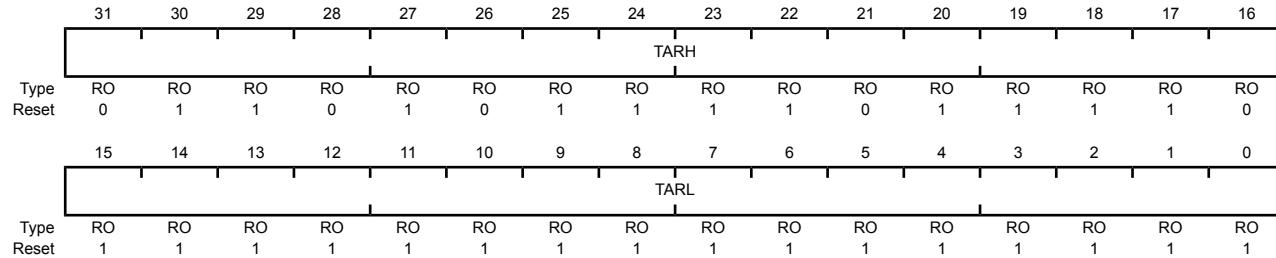
Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Timer3 base: 0x4003.3000

Offset 0x048

Type RO, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)



Bit/Field	Name	Type	Reset	Description
31:16	TARH	RO	0xFFFF (32-bit mode) 0x0000 (16-bit mode)	GPTM TimerA Register High  If the <b>GPTMCFG</b> is in a 32-bit mode, TimerB value is read. If the <b>GPTMCFG</b> is in a 16-bit mode, this is read as zero.
15:0	TARL	RO	0xFFFF	GPTM TimerA Register Low  A read returns the current value of the <b>GPTM TimerA Count Register</b> , except in Input Edge Count mode, when it returns the timestamp from the last edge event.

## Register 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 time at which the last edge event took place.

### GPTM TimerB (GPTMTBR)

Timer0 base: 0x4003.0000

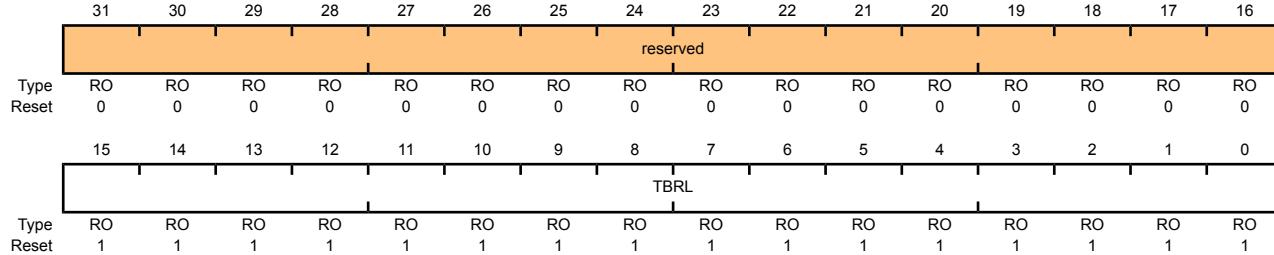
Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Timer3 base: 0x4003.3000

Offset 0x04C

Type RO, reset 0x0000.FFFF



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 timestamp from the last edge event.

# 11 Watchdog Timer

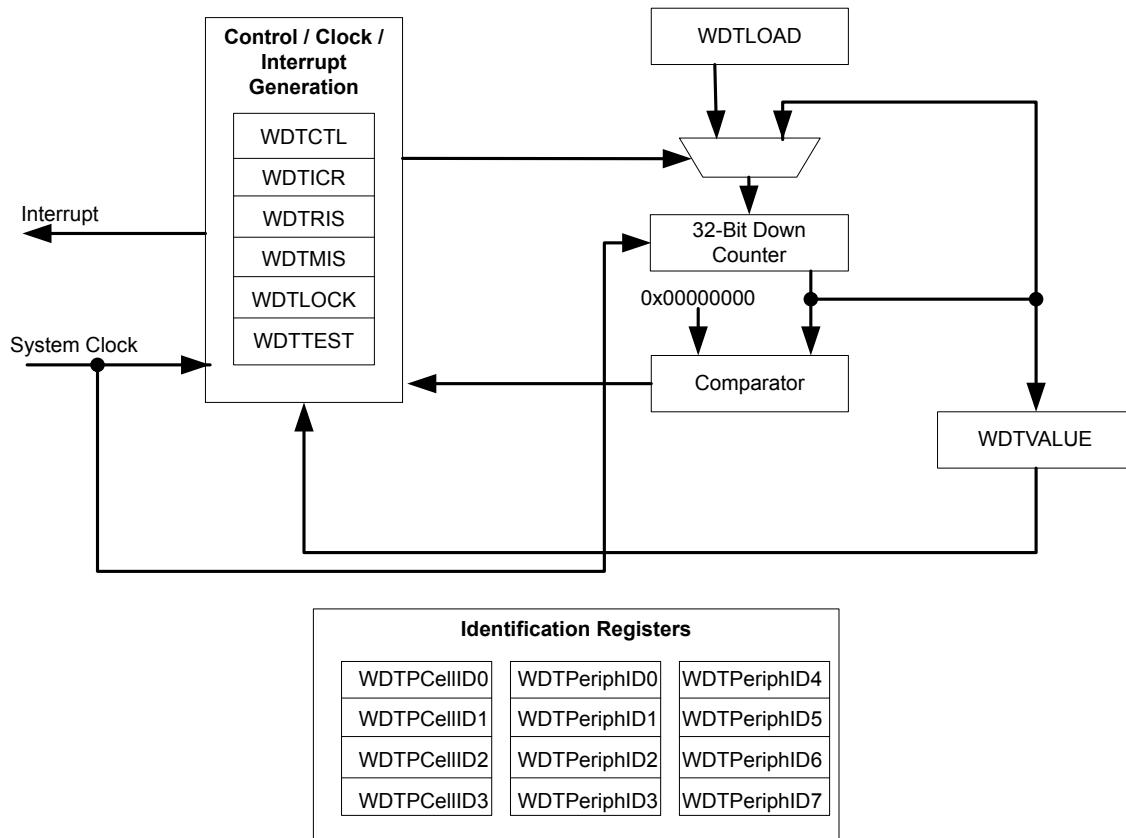
A watchdog timer can generate nonmaskable interrupts (NMIs) or a reset when a time-out value is reached. The watchdog timer is used to regain control when a system has failed due to a software error or due to the failure of an external device to respond in the expected way.

The Stellaris® Watchdog Timer module consists of a 32-bit down counter, a programmable load register, interrupt generation logic, a locking register, and user-enabled stalling.

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

## 11.1 Block Diagram

Figure 11-1. WDT Module Block Diagram



## 11.2 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.

## 11.3 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.

## 11.4 Register Map

Table 11-1 on page 243 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 11-1. Watchdog Timer Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	WDTLOAD	R/W	0xFFFF.FFFF	Watchdog Load	245
0x004	WDTVALUE	RO	0xFFFF.FFFF	Watchdog Value	246
0x008	WDTCTL	R/W	0x0000.0000	Watchdog Control	247
0x00C	WDTICR	WO	-	Watchdog Interrupt Clear	248
0x010	WDTRIS	RO	0x0000.0000	Watchdog Raw Interrupt Status	249
0x014	WDTMIS	RO	0x0000.0000	Watchdog Masked Interrupt Status	250
0x418	WDTTEST	R/W	0x0000.0000	Watchdog Test	251
0xC00	WDTLOCK	R/W	0x0000.0000	Watchdog Lock	252

Offset	Name	Type	Reset	Description	See page
0xFD0	WDTPeriphID4	RO	0x0000.0000	Watchdog Peripheral Identification 4	253
0xFD4	WDTPeriphID5	RO	0x0000.0000	Watchdog Peripheral Identification 5	254
0xFD8	WDTPeriphID6	RO	0x0000.0000	Watchdog Peripheral Identification 6	255
0xFDC	WDTPeriphID7	RO	0x0000.0000	Watchdog Peripheral Identification 7	256
0xFE0	WDTPeriphID0	RO	0x0000.0005	Watchdog Peripheral Identification 0	257
0xFE4	WDTPeriphID1	RO	0x0000.0018	Watchdog Peripheral Identification 1	258
0xFE8	WDTPeriphID2	RO	0x0000.0018	Watchdog Peripheral Identification 2	259
0xFEC	WDTPeriphID3	RO	0x0000.0001	Watchdog Peripheral Identification 3	260
0xFF0	WDTPCellID0	RO	0x0000.000D	Watchdog PrimeCell Identification 0	261
0xFF4	WDTPCellID1	RO	0x0000.00F0	Watchdog PrimeCell Identification 1	262
0xFF8	WDTPCellID2	RO	0x0000.0005	Watchdog PrimeCell Identification 2	263
0xFFC	WDTPCellID3	RO	0x0000.00B1	Watchdog PrimeCell Identification 3	264

## 11.5 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

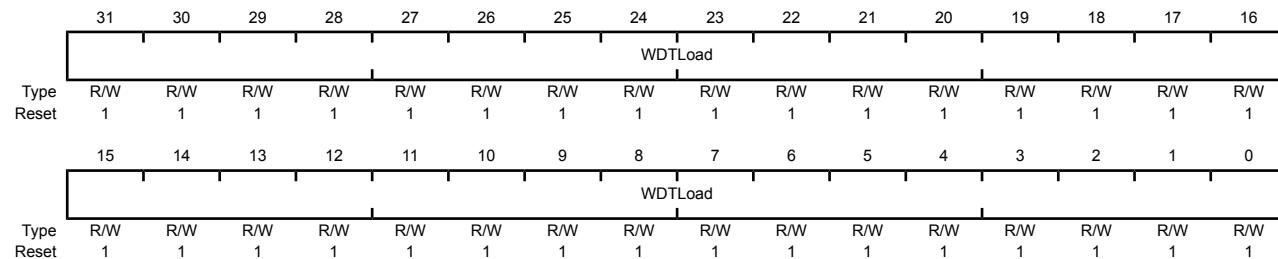
This register is the 32-bit interval value used by the 32-bit counter. When this register is written, the value is immediately loaded and the counter restarts counting down from the new value. If the **WDTLOAD** register is loaded with 0x0000.0000, an interrupt is immediately generated.

### 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

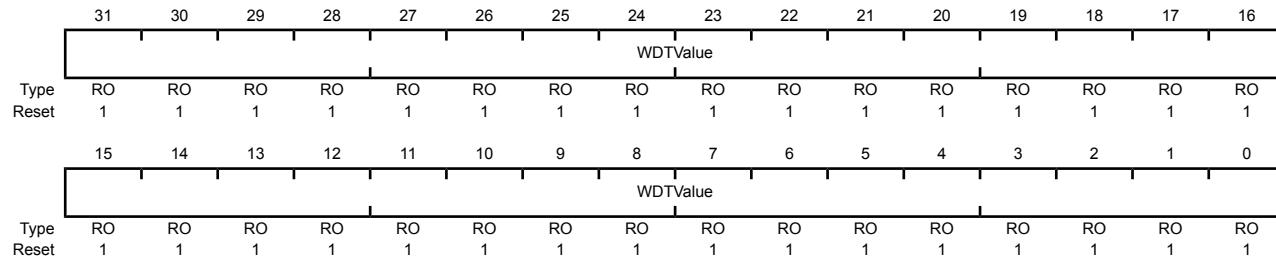
This register contains the current count value of the timer.

### 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.

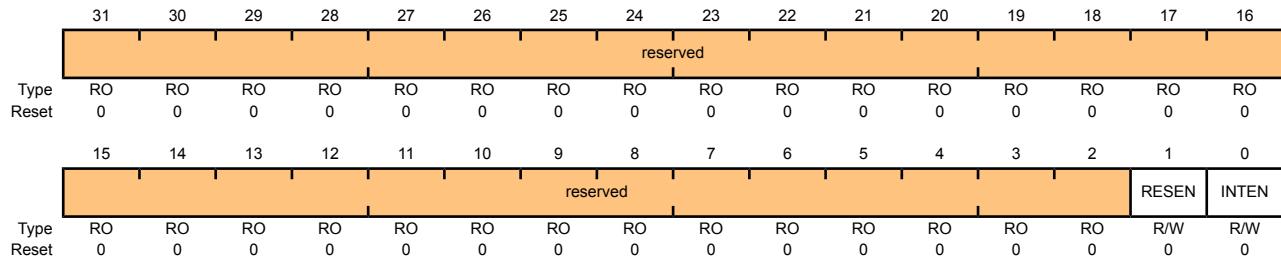
## Register 3: Watchdog Control (WDTCTL), offset 0x008

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

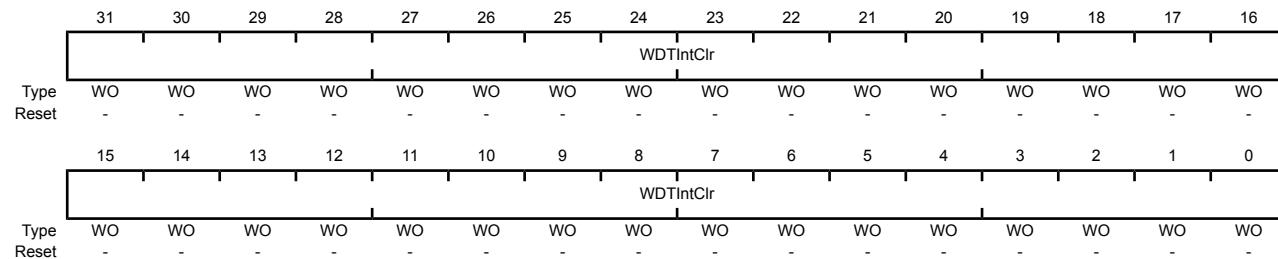
This register is the interrupt clear register. A write of any value to this register clears the Watchdog interrupt and reloads the 32-bit counter from the **WDTLOAD** register. Value for a read or reset is indeterminate.

### Watchdog Interrupt Clear (WDTICR)

Base 0x4000.0000

Offset 0x00C

Type WO, reset -



#### Bit/Field

#### Name

#### Type

#### Reset

#### Description

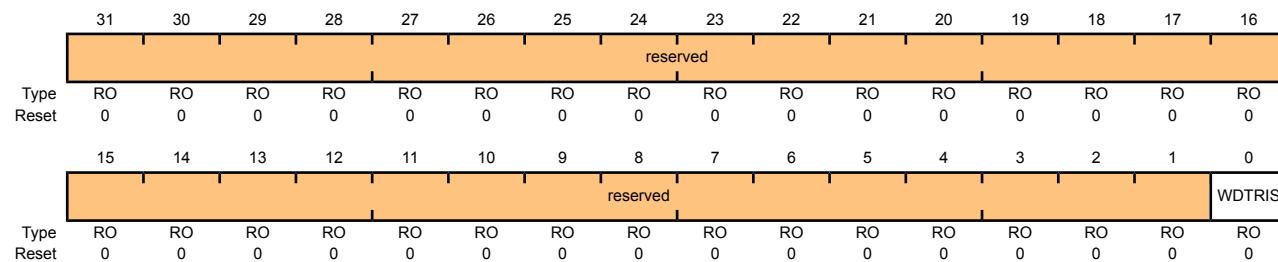
31:0	WDTIntClr	WO	-	Watchdog Interrupt Clear
------	-----------	----	---	--------------------------

## 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



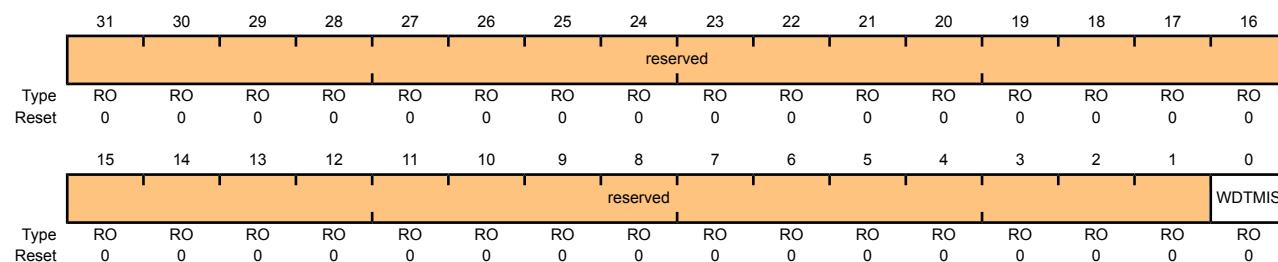
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	WDTRIS	RO	0	Watchdog Raw Interrupt Status  Gives 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



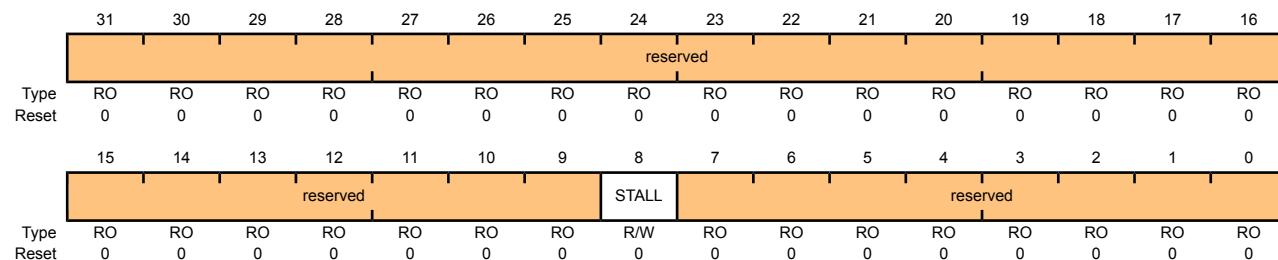
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	WDTMIS	RO	0	Watchdog Masked Interrupt Status  Gives 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

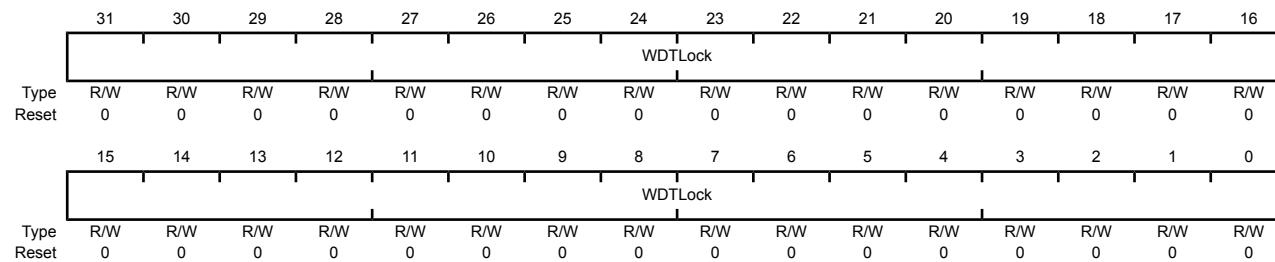
Writing 0x1ACC.E551 to the **WDTLOCK** register enables write access to all other registers. Writing any other value to the **WDTLOCK** register re-enables the locked state for register writes to all the other registers. Reading the **WDTLOCK** register returns the lock status rather than the 32-bit 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 re-applies the lock, preventing any register updates.

A read of this register returns the following values:

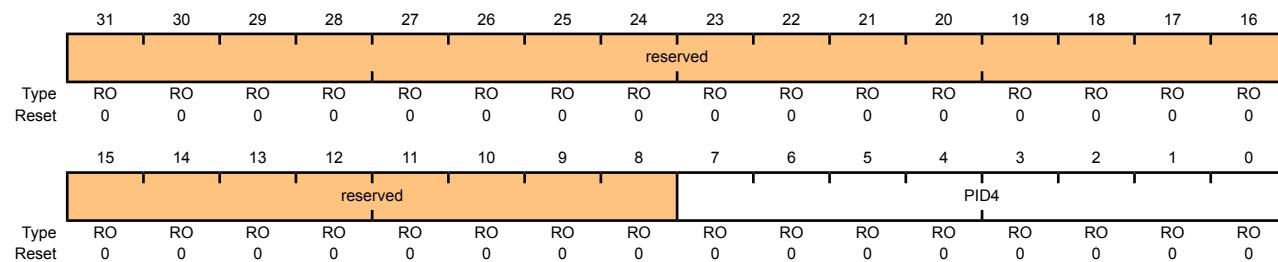
Value	Description
0x0000.0001	Locked
0x0000.0000	Unlocked

## 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

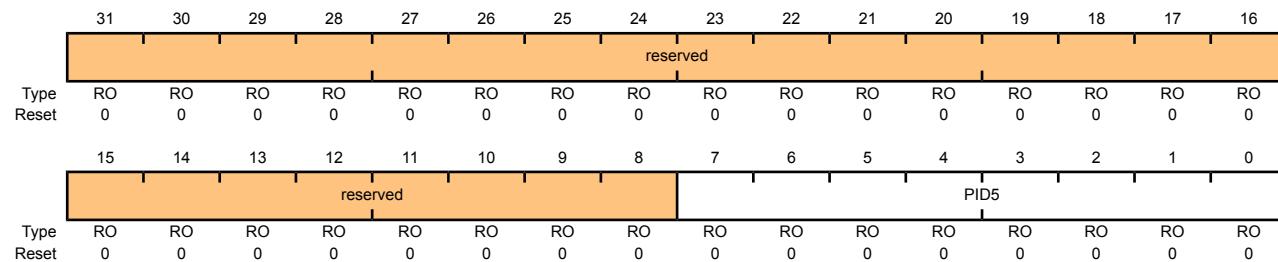
The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### 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

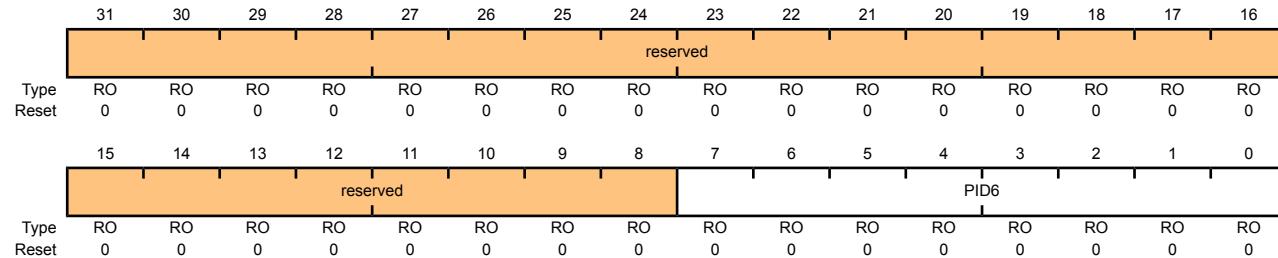
The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### Watchdog Peripheral Identification 6 (WDTPeriphID6)

Base 0x4000.0000

Offset 0xFD8

Type RO, reset 0x0000.0000



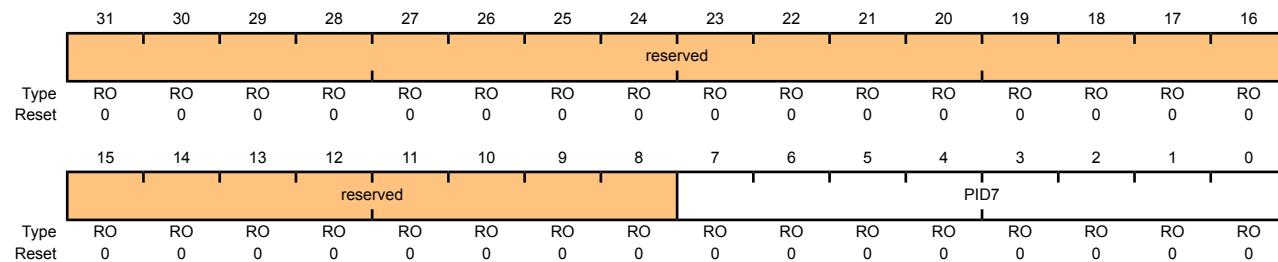
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	WDT Peripheral ID Register[23:16]

## 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

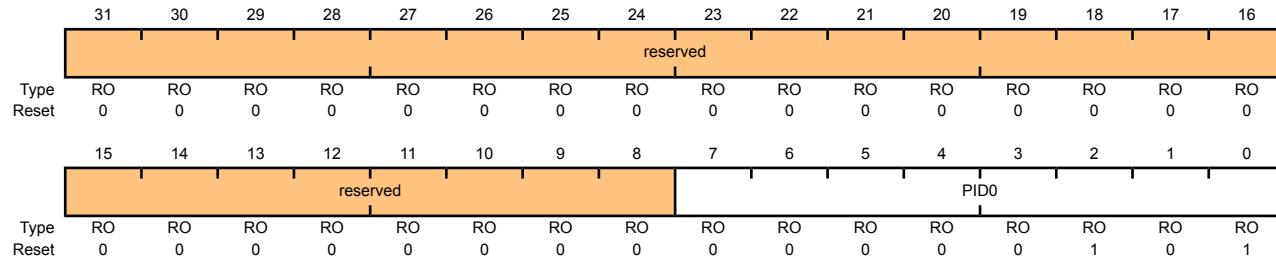
The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### Watchdog Peripheral Identification 0 (WDTPeriphID0)

Base 0x4000.0000

Offset 0xFE0

Type RO, reset 0x0000.0005



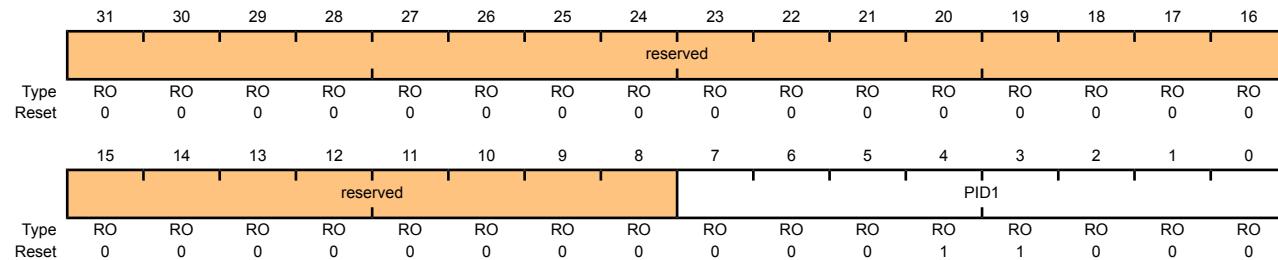
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x05	Watchdog Peripheral ID Register[7:0]

## 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

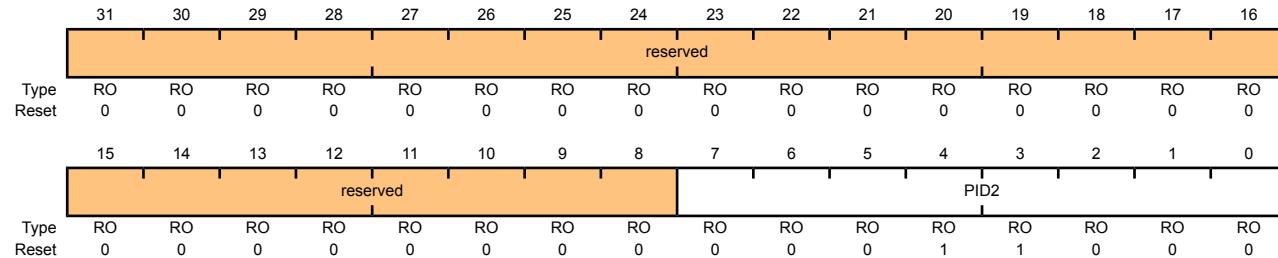
The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### 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

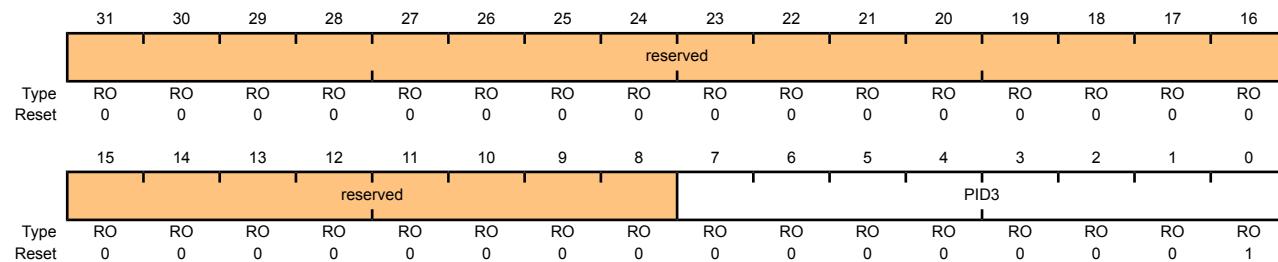
The **WDTPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### Watchdog Peripheral Identification 3 (WDTPeriphID3)

Base 0x4000.0000

Offset 0xFEC

Type RO, reset 0x0000.0001



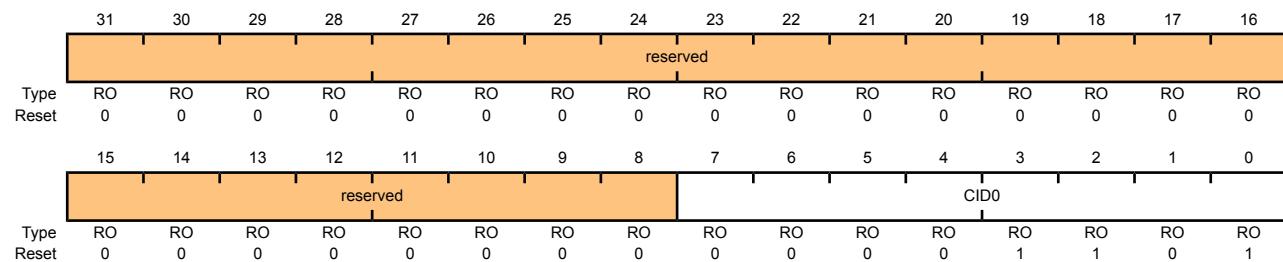
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	Watchdog Peripheral ID Register[31:24]

## 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 (WDTPCellID0)

Base 0x4000.0000  
Offset 0xFF0  
Type RO, reset 0x0000.000D



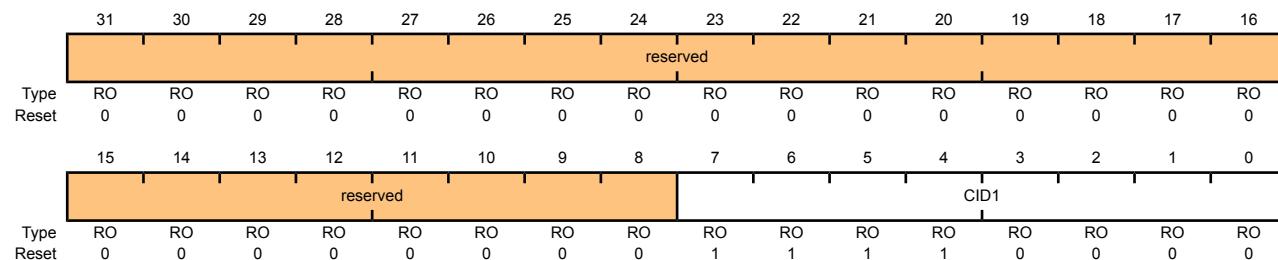
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	Watchdog PrimeCell ID Register[7:0]

## 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



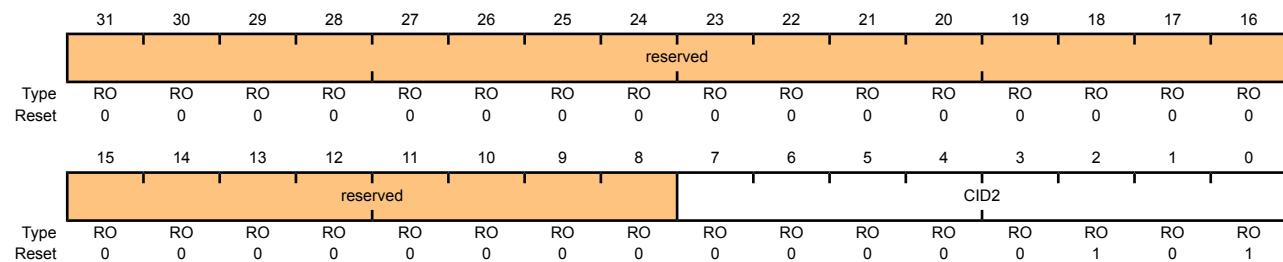
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	Watchdog PrimeCell ID Register[15:8]

## 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



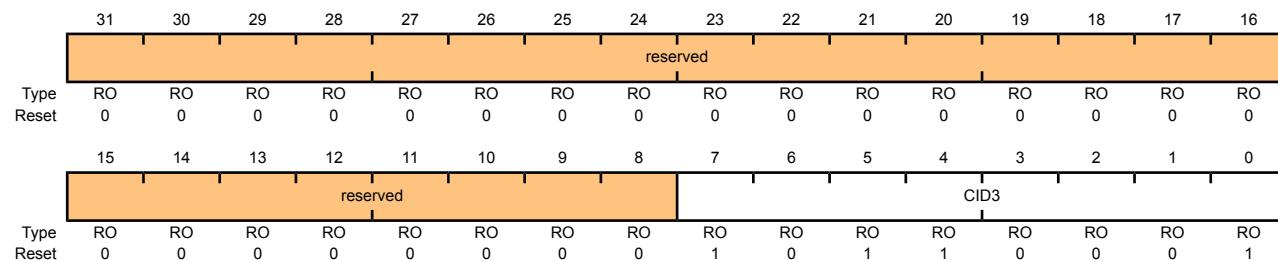
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	Watchdog PrimeCell ID Register[23:16]

## 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]

## 12 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.

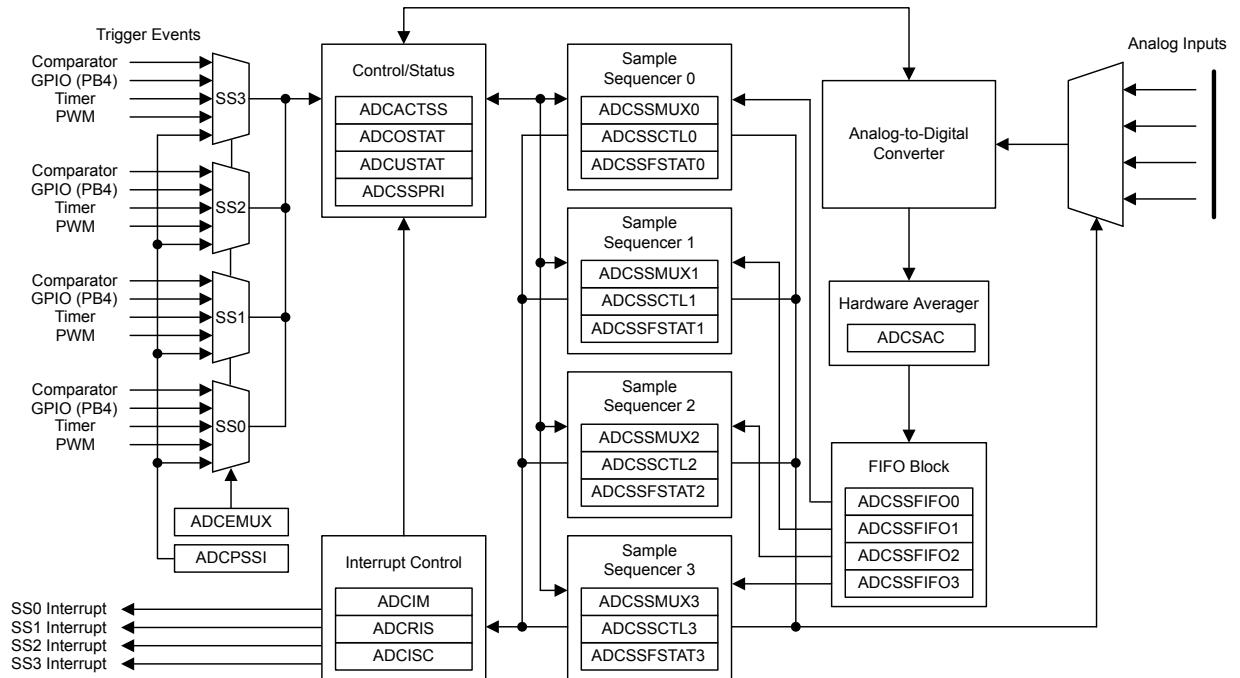
The Stellaris® ADC module features 10-bit conversion resolution and supports four input channels, plus an internal temperature sensor. The ADC module contains a programmable sequencer which allows for the sampling of multiple analog input sources without controller intervention. Each sample sequence provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequence priority.

The Stellaris® ADC provides the following features:

- Four analog input channels
- Single-ended and differential-input configurations
- Internal temperature sensor
- Sample rate of 500 thousand samples/second
- Four programmable sample conversion sequences from one to eight entries long, with corresponding conversion result FIFOs
- Flexible trigger control
  - Controller (software)
  - Timers
  - Analog Comparators
  - PWM
  - GPIO
- Hardware averaging of up to 64 samples for improved accuracy

## 12.1 Block Diagram

Figure 12-1. ADC Module Block Diagram



## 12.2 Functional Description

The Stellaris® ADC collects sample data by using a programmable sequence-based approach instead of the traditional single or double-sampling approach found on many ADC modules. Each *sample sequence* is a fully programmed series of consecutive (back-to-back) samples, allowing the ADC to collect data from multiple input sources without having to be re-configured or serviced by the controller. The programming of each sample in the sample sequence includes parameters such as the input source and mode (differential versus single-ended input), interrupt generation on sample completion, and the indicator for the last sample in the sequence.

### 12.2.1 Sample Sequencers

The sampling control and data capture is handled by the Sample Sequencers. All of the sequencers are identical in implementation except for the number of samples that can be captured and the depth of the FIFO. Table 12-1 on page 266 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 12-1. Samples and FIFO Depth of Sequencers

Sequencer	Number of Samples	Depth of FIFO
SS3	1	1
SS2	4	4
SS1	4	4
SS0	8	8

For a given sample sequence, each sample is defined by two 4-bit nibbles in the **ADC Sample Sequence Input Multiplexer Select (ADCSSMUXn)** and **ADC Sample Sequence Control (ADCSSCTLn)** registers, where "n" corresponds to the sequence number. The **ADCSSMUXn** nibbles select the input pin, while the **ADCSSCTLn** nibbles contain the sample control bits corresponding to parameters such as temperature sensor selection, interrupt enable, end of sequence, and differential input mode. Sample Sequencers are enabled by setting the respective **ASEN<sub>n</sub>** bit in the **ADC Active Sample Sequencer (ADCACTSS)** register, but can be configured before being enabled.

When configuring a sample sequence, multiple uses of the same input pin within the same sequence is allowed. In the **ADCSSCTLn** register, the **Interrupt Enable (IE)** bits can be set for any combination of samples, allowing interrupts to be generated after every sample in the sequence if necessary. Also, the **END** bit can be set at any point within a sample sequence. For example, if Sequencer 0 is used, the **END** bit can be set in the nibble associated with the fifth sample, allowing Sequencer 0 to complete execution of the sample sequence after the fifth sample.

After a sample sequence completes execution, the result data can be retrieved from the **ADC Sample Sequence Result FIFO (ADCSSFIFO<sub>n</sub>)** 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.

## 12.2.2 Module Control

Outside of the Sample Sequencers, the remainder of the control logic is responsible for tasks such as interrupt generation, sequence prioritization, and trigger configuration.

Most of the ADC control logic runs at the ADC clock rate of 14-18 MHz. The internal ADC divider is configured automatically by hardware when the system **XTAL** is selected. The automatic clock divider configuration targets 16.667 MHz operation for all Stellaris® devices.

### 12.2.2.1 Interrupts

The Sample Sequencers dictate the events that cause interrupts, but they don't have control over whether the interrupt is actually sent to the interrupt controller. The ADC module's interrupt signal is controlled by the state of the **MASK** bits in the **ADC Interrupt Mask (ADCIM)** register. Interrupt status can be viewed at two locations: the **ADC Raw Interrupt Status (ADCRIS)** register, which shows the raw status of a Sample Sequencer's interrupt signal, and the **ADC Interrupt Status and Clear (ADCISC)** register, which shows the logical AND of the **ADCRIS** register's **INR** bit and the **ADCIM** register's **MASK** bits. Interrupts are cleared by writing a 1 to the corresponding **IN** bit in **ADCISC**.

### 12.2.2.2 Prioritization

When sampling events (triggers) happen concurrently, they are prioritized for processing by the values in the **ADC Sample Sequencer Priority (ADCSSPRI)** register. Valid priority values are in the range of 0-3, with 0 being the highest priority and 3 being the lowest. Multiple active Sample Sequencer units with the same priority do not provide consistent results, so software must ensure that all active Sample Sequencer units have a unique priority value.

### 12.2.2.3 Sampling Events

Sample triggering for each Sample Sequencer is defined in the **ADC Event Multiplexer Select (ADCEMUX)** register. The external peripheral triggering sources vary by Stellaris® family member,

but all devices share the "Controller" and "Always" triggers. Software can initiate sampling by setting the **CH** bits in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register.

When using the "Always" trigger, care must be taken. If a sequence's priority is too high, it is possible to starve other lower priority sequences.

### 12.2.3 Hardware Sample Averaging Circuit

Higher precision results can be generated using the hardware averaging circuit, however, the improved results are at the cost of throughput. Up to 64 samples can be accumulated and averaged to form a single data entry in the sequencer FIFO. Throughput is decreased proportionally to the number of samples in the averaging calculation. For example, if the averaging circuit is configured to average 16 samples, the throughput is decreased by a factor of 16.

By default the averaging circuit is off and all data from the converter passes through to the sequencer FIFO. The averaging hardware is controlled by the **ADC Sample Averaging Control (ADCSAC)** register (see page 283). There is a single averaging circuit and all input channels receive the same amount of averaging whether they are single-ended or differential.

### 12.2.4 Analog-to-Digital Converter

The converter itself generates a 10-bit output value for selected analog input. Special analog pads are used to minimize the distortion on the input.

### 12.2.5 Test Modes

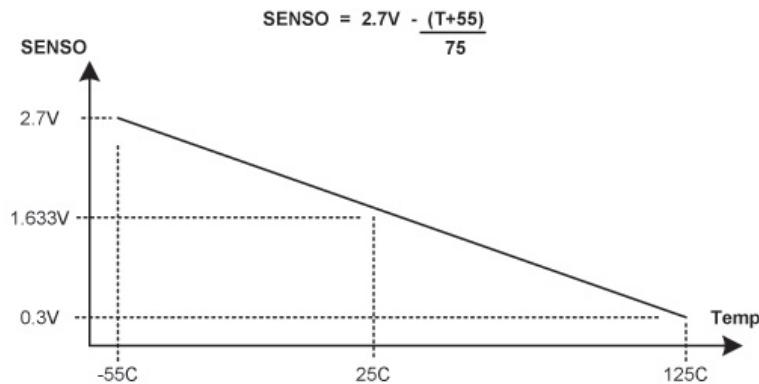
There is a user-available test mode that allows for loopback operation within the digital portion of the ADC module. This can be useful for debugging software without having to provide actual analog stimulus. This mode is available through the **ADC Test Mode Loopback (ADCTMLB)** register (see page 296).

### 12.2.6 Internal Temperature Sensor

The internal temperature sensor provides an analog temperature reading as well as a reference voltage. The voltage at the output terminal SENSO is given by the following equation:

$$SENDO = 2.7 - ((T + 55) / 75)$$

This relation is shown in Figure 12-2 on page 269.

**Figure 12-2. Internal Temperature Sensor Characteristic**

## 12.3 Initialization and Configuration

In order for the ADC module to be used, the PLL must be enabled and using a supported crystal frequency (see the **RCC** register). Using unsupported frequencies can cause faulty operation in the ADC module.

### 12.3.1 Module Initialization

Initialization of the ADC module is a simple process with very few steps. The main steps include enabling the clock to the ADC and reconfiguring the Sample Sequencer priorities (if needed).

The initialization sequence for the ADC is as follows:

1. Enable the ADC clock by writing a value of 0x0001.0000 to the **RCGC1** register (see page 102).
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.

### 12.3.2 Sample Sequencer Configuration

Configuration of the Sample Sequencers is slightly more complex than the module initialization since each sample sequence is completely programmable.

The configuration for each Sample Sequencer should be as follows:

1. Ensure that the Sample Sequencer is disabled by writing a 0 to the corresponding **ASEN** bit in the **ADCACTSS** register. Programming of the Sample Sequencers is allowed without having them enabled. Disabling the Sequencer during programming prevents erroneous execution if a trigger event were to occur during the configuration process.
2. Configure the trigger event for the Sample Sequencer in the **ADCEMUX** register.
3. For each sample in the sample sequence, configure the corresponding input source in the **ADCSSMUXn** register.

4. For each sample in the sample sequence, configure the sample control bits in the corresponding nibble in the **ADCSSCTLn** register. When programming the last nibble, ensure that the `END` bit is set. Failure to set the `END` bit causes unpredictable behavior.
5. If interrupts are to be used, write a 1 to the corresponding `MASK` bit in the **ADCIM** register.
6. Enable the Sample Sequencer logic by writing a 1 to the corresponding `ASEN` bit in the **ADCACTSS** register.

## 12.4 Register Map

Table 12-2 on page 270 lists the ADC registers. The offset listed is a hexadecimal increment to the register's address, relative to the ADC base address of 0x4003.8000.

**Table 12-2. ADC Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	ADCACTSS	R/W	0x0000.0000	ADC Active Sample Sequencer	272
0x004	ADCRIS	RO	0x0000.0000	ADC Raw Interrupt Status	273
0x008	ADCIM	R/W	0x0000.0000	ADC Interrupt Mask	274
0x00C	ADCISC	R/W1C	0x0000.0000	ADC Interrupt Status and Clear	275
0x010	ADCOSTAT	R/W1C	0x0000.0000	ADC Overflow Status	276
0x014	ADCEMUX	R/W	0x0000.0000	ADC Event Multiplexer Select	277
0x018	ADCUSTAT	R/W1C	0x0000.0000	ADC Underflow Status	280
0x020	ADCSPRI	R/W	0x0000.3210	ADC Sample Sequencer Priority	281
0x028	ADCPSSI	WO	-	ADC Processor Sample Sequence Initiate	282
0x030	ADCSAC	R/W	0x0000.0000	ADC Sample Averaging Control	283
0x040	ADCSSMUX0	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 0	284
0x044	ADCSSCTL0	R/W	0x0000.0000	ADC Sample Sequence Control 0	286
0x048	ADCSSFIFO0	RO	0x0000.0000	ADC Sample Sequence Result FIFO 0	289
0x04C	ADCSSFSTAT0	RO	0x0000.0100	ADC Sample Sequence FIFO 0 Status	290
0x060	ADCSSMUX1	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 1	291
0x064	ADCSSCTL1	R/W	0x0000.0000	ADC Sample Sequence Control 1	292
0x068	ADCSSFIFO1	RO	0x0000.0000	ADC Sample Sequence Result FIFO 1	289
0x06C	ADCSSFSTAT1	RO	0x0000.0100	ADC Sample Sequence FIFO 1 Status	290
0x080	ADCSSMUX2	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 2	291
0x084	ADCSSCTL2	R/W	0x0000.0000	ADC Sample Sequence Control 2	292
0x088	ADCSSFIFO2	RO	0x0000.0000	ADC Sample Sequence Result FIFO 2	289
0x08C	ADCSSFSTAT2	RO	0x0000.0100	ADC Sample Sequence FIFO 2 Status	290
0x0A0	ADCSSMUX3	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 3	294

Offset	Name	Type	Reset	Description	See page
0x0A4	ADCSSCTL3	R/W	0x0000.0002	ADC Sample Sequence Control 3	295
0x0A8	ADCSSFIFO3	RO	0x0000.0000	ADC Sample Sequence Result FIFO 3	289
0x0AC	ADCSSFSTAT3	RO	0x0000.0100	ADC Sample Sequence FIFO 3 Status	290
0x100	ADCTMLB	R/W	0x0000.0000	ADC Test Mode Loopback	296

## 12.5 Register Descriptions

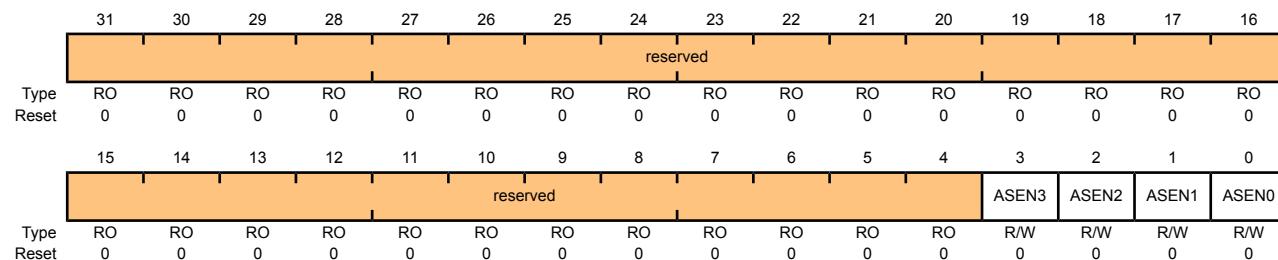
The remainder of this section lists and describes the ADC registers, in numerical order by address offset.

## Register 1: ADC Active Sample Sequencer (ADCACTSS), offset 0x000

This register controls the activation of the Sample Sequencers. Each Sample Sequencer can be enabled/disabled independently.

ADC Active Sample Sequencer (ADCACTSS)

Base 0x4003.8000  
Offset 0x000  
Type R/W, reset 0x0000.0000



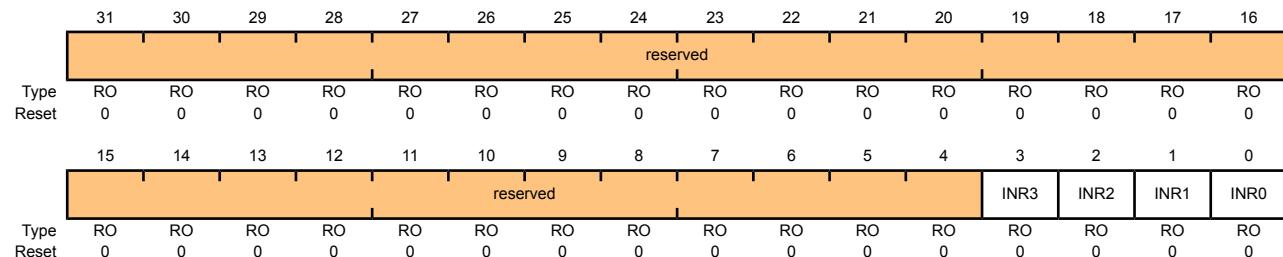
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	ASEN3	R/W	0	ADC SS3 Enable  Specifies whether Sample 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



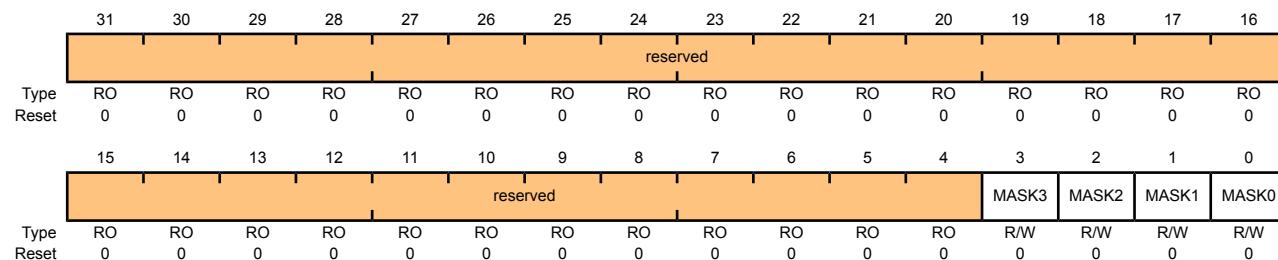
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	INR3	RO	0	SS3 Raw Interrupt Status Set by hardware when a sample with its respective <b>ADCSSCTL3 IE</b> bit has completed conversion. This bit is cleared by writing a 1 to the <b>ADCISC IN3</b> bit.
2	INR2	RO	0	SS2 Raw Interrupt Status Set by hardware when a sample with its respective <b>ADCSSCTL2 IE</b> bit has completed conversion. This bit is cleared by writing a 1 to the <b>ADCISC IN2</b> bit.
1	INR1	RO	0	SS1 Raw Interrupt Status Set by hardware when a sample with its respective <b>ADCSSCTL1 IE</b> bit has completed conversion. This bit is cleared by writing a 1 to the <b>ADCISC IN1</b> bit.
0	INR0	RO	0	SS0 Raw Interrupt Status Set by hardware when a sample with its respective <b>ADCSSCTL0 IE</b> bit has completed conversion. This bit is cleared by writing a 1 to the <b>ADCISC IN0</b> bit.

## Register 3: ADC Interrupt Mask (ADCIM), offset 0x008

This register controls whether the Sample Sequencer raw interrupt signals are promoted to controller interrupts. The raw interrupt signal for each Sample Sequencer can be masked independently.

### ADC Interrupt Mask (ADCIM)

Base 0x4003.8000  
Offset 0x008  
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	MASK3	R/W	0	SS3 Interrupt Mask  Specifies whether the raw interrupt signal from Sample Sequencer 3 ( <b>ADCRIS</b> register INR3 bit) is promoted to a controller interrupt. If set, the raw interrupt signal is promoted to a controller interrupt. Otherwise, it is not.
2	MASK2	R/W	0	SS2 Interrupt Mask  Specifies whether the raw interrupt signal from Sample Sequencer 2 ( <b>ADCRIS</b> register INR2 bit) is promoted to a controller interrupt. If set, the raw interrupt signal is promoted to a controller interrupt. Otherwise, it is not.
1	MASK1	R/W	0	SS1 Interrupt Mask  Specifies whether the raw interrupt signal from Sample Sequencer 1 ( <b>ADCRIS</b> register INR1 bit) is promoted to a controller interrupt. If set, the raw interrupt signal is promoted to a controller interrupt. Otherwise, it is not.
0	MASK0	R/W	0	SS0 Interrupt Mask  Specifies whether the raw interrupt signal from Sample Sequencer 0 ( <b>ADCRIS</b> register INR0 bit) is promoted to a controller interrupt. If set, the raw interrupt signal is promoted to a controller interrupt. Otherwise, it is not.

## Register 4: ADC Interrupt Status and Clear (ADCISC), offset 0x00C

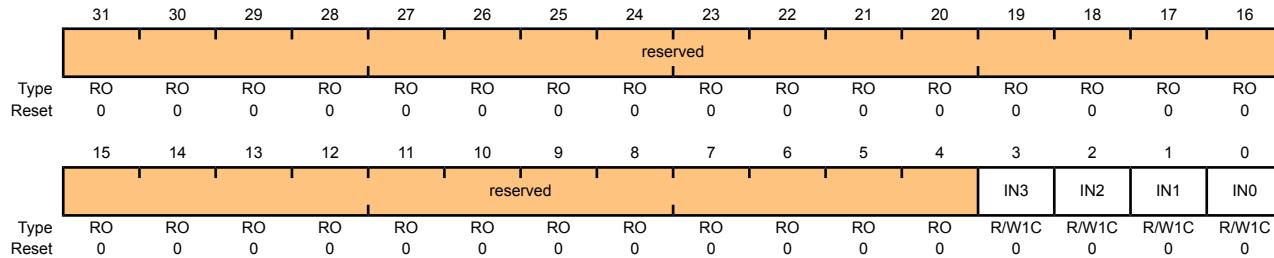
This register provides the mechanism for clearing interrupt conditions, and shows the status of controller interrupts generated by the Sample Sequencers. When read, each bit field is the logical AND of the respective `INR` and `MASK` bits. Interrupts are cleared by writing a 1 to the corresponding bit position. If software is polling the `ADCRIS` instead of generating interrupts, the `INR` bits are still cleared via the `ADCISC` register, even if the `IN` bit is not set.

### ADC Interrupt Status and Clear (ADCISC)

Base 0x4003.8000

Offset 0x00C

Type R/W1C, reset 0x0000.0000



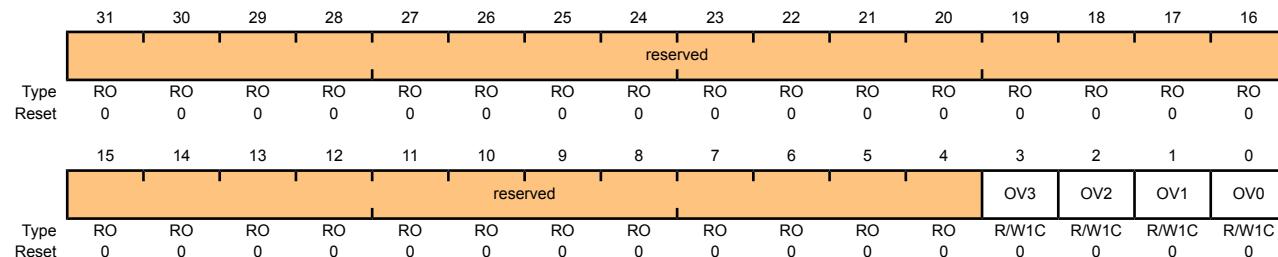
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	IN3	R/W1C	0	SS3 Interrupt Status and Clear  This bit is set by hardware when the <code>MASK3</code> and <code>INR3</code> bits are both 1, providing a level-based interrupt to the controller. It is cleared by writing a 1, and also clears the <code>INR3</code> bit.
2	IN2	R/W1C	0	SS2 Interrupt Status and Clear  This bit is set by hardware when the <code>MASK2</code> and <code>INR2</code> bits are both 1, providing a level based interrupt to the controller. It is cleared by writing a 1, and also clears the <code>INR2</code> bit.
1	IN1	R/W1C	0	SS1 Interrupt Status and Clear  This bit is set by hardware when the <code>MASK1</code> and <code>INR1</code> bits are both 1, providing a level based interrupt to the controller. It is cleared by writing a 1, and also clears the <code>INR1</code> bit.
0	IN0	R/W1C	0	SS0 Interrupt Status and Clear  This bit is set by hardware when the <code>MASK0</code> and <code>INR0</code> bits are both 1, providing a level based interrupt to the controller. It is cleared by writing a 1, and also clears the <code>INR0</code> bit.

## Register 5: ADC Overflow Status (ADCOSTAT), offset 0x010

This register indicates overflow conditions in the Sample Sequencer FIFOs. Once the overflow condition has been handled by software, the condition can be cleared by writing a 1 to the corresponding bit position.

### ADC Overflow Status (ADCOSTAT)

Base 0x4003.8000  
Offset 0x010  
Type R/W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	OV3	R/W1C	0	SS3 FIFO Overflow  This bit specifies that the FIFO for Sample Sequencer 3 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped and this bit is set by hardware to indicate the occurrence of dropped data. This bit is cleared by writing a 1.
2	OV2	R/W1C	0	SS2 FIFO Overflow  This bit specifies that the FIFO for Sample Sequencer 2 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped and this bit is set by hardware to indicate the occurrence of dropped data. This bit is cleared by writing a 1.
1	OV1	R/W1C	0	SS1 FIFO Overflow  This bit specifies that the FIFO for Sample Sequencer 1 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped and this bit is set by hardware to indicate the occurrence of dropped data. This bit is cleared by writing a 1.
0	OV0	R/W1C	0	SS0 FIFO Overflow  This bit specifies that the FIFO for Sample Sequencer 0 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped and this bit is set by hardware to indicate the occurrence of dropped data. This bit is cleared by writing a 1.

## 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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:12	EM3	R/W	0x00	SS3 Trigger Select  This field selects the trigger source for Sample Sequencer 3.  The valid configurations for this field are:

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
11:8	EM2	R/W	0x00	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 0x6 PWM0 0x7 PWM1 0x8 PWM2 0x9-0xE reserved 0xF Always (continuously sample)
7:4	EM1	R/W	0x00	SS1 Trigger Select This field selects the trigger source for Sample Sequencer 1. 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 0x6 PWM0 0x7 PWM1 0x8 PWM2 0x9-0xE reserved 0xF Always (continuously sample)

---

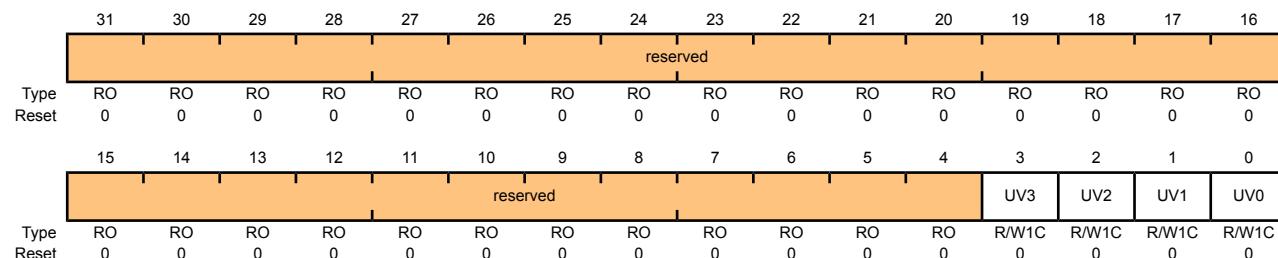
Bit/Field	Name	Type	Reset	Description
3:0	EM0	R/W	0x00	SS0 Trigger Select
This field selects the trigger source for Sample Sequencer 0.				
The valid configurations for this field are:				
		Value		Event
		0x0		Controller (default)
		0x1		Analog Comparator 0
		0x2		Reserved
		0x3		Reserved
		0x4		External (GPIO PB4)
		0x5		Timer
		0x6		PWM0
		0x7		PWM1
		0x8		PWM2
		0x9-0xE		reserved
		0xF		Always (continuously sample)

## Register 7: ADC Underflow Status (ADCUSTAT), offset 0x018

This register indicates underflow conditions in the Sample Sequencer FIFOs. The corresponding underflow condition can be cleared by writing a 1 to the relevant bit position.

### ADC Underflow Status (ADCUSTAT)

Base 0x4003.8000  
Offset 0x018  
Type R/W1C, reset 0x0000.0000



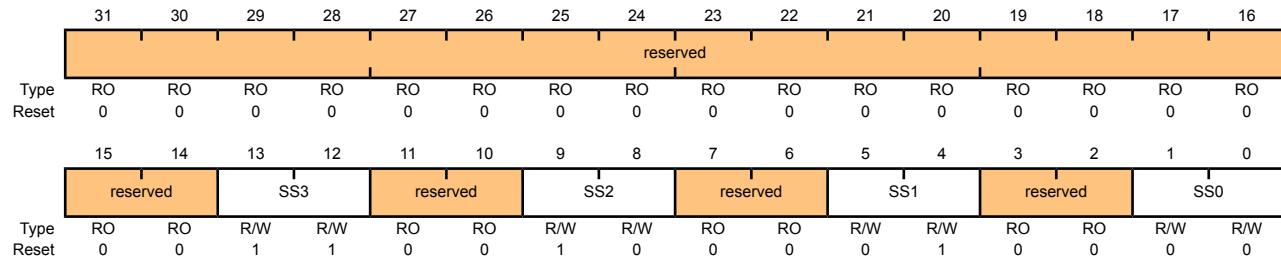
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	UV3	R/W1C	0	SS3 FIFO Underflow  This bit specifies that the FIFO for Sample Sequencer 3 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. This bit is cleared by writing a 1.
2	UV2	R/W1C	0	SS2 FIFO Underflow  This bit specifies that the FIFO for Sample Sequencer 2 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. This bit is cleared by writing a 1.
1	UV1	R/W1C	0	SS1 FIFO Underflow  This bit specifies that the FIFO for Sample Sequencer 1 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. This bit is cleared by writing a 1.
0	UV0	R/W1C	0	SS0 FIFO Underflow  This bit specifies that the FIFO for Sample Sequencer 0 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. This bit is cleared by writing a 1.

## Register 8: ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020

This register sets the priority for each of the Sample Sequencers. Out of reset, Sequencer 0 has the highest priority, and sample sequence 3 has the lowest priority. When reconfiguring sequence priorities, each sequence must have a unique priority or the ADC behavior is inconsistent.

ADC Sample Sequencer Priority (ADCSSPRI)

Base 0x4003.8000  
Offset 0x020  
Type R/W, reset 0x0000.3210



Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13:12	SS3	R/W	0x3	SS3 Priority  The SS3 field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 3. A priority encoding of 0 is highest and 3 is lowest. The priorities assigned to the Sequencers must be uniquely mapped. ADC behavior is not consistent if two or more fields are equal.
11:10	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:8	SS2	R/W	0x2	SS2 Priority  The SS2 field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 2.
7:6	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:4	SS1	R/W	0x1	SS1 Priority  The SS1 field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 1.
3:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1:0	SS0	R/W	0x0	SS0 Priority  The SS0 field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 0.

## Register 9: ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028

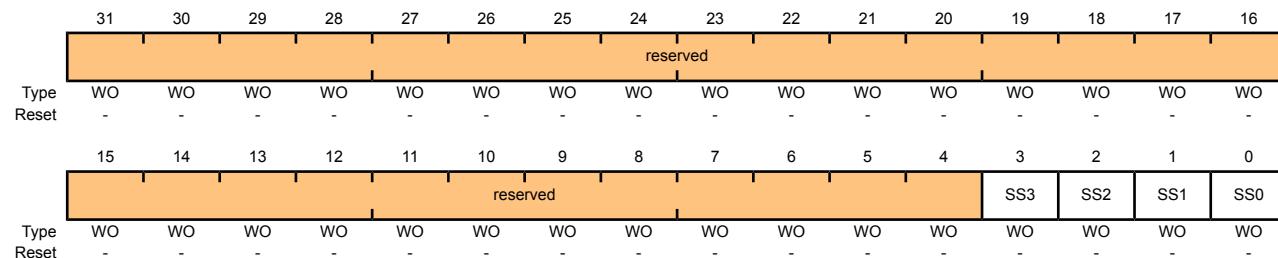
This register provides a mechanism for application software to initiate sampling in the Sample Sequencers. Sample sequences can be initiated individually or in any combination. When multiple sequences are triggered simultaneously, the priority encodings in **ADCSSPRI** dictate execution order.

### ADC Processor Sample Sequence Initiate (ADCPSSI)

Base 0x4003.8000

Offset 0x028

Type WO, reset -



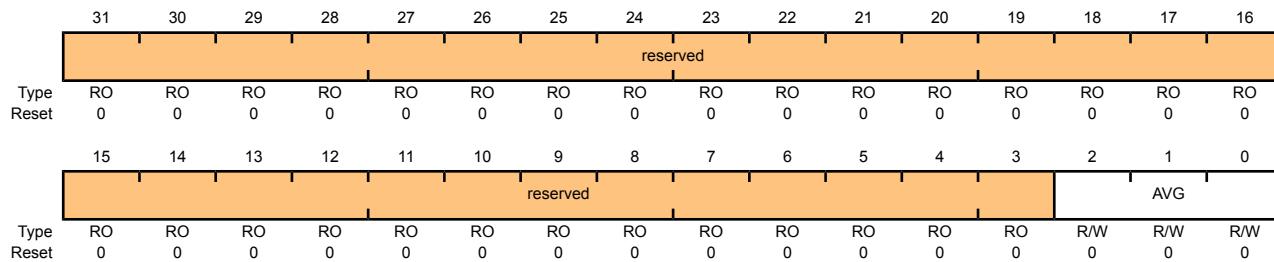
Bit/Field	Name	Type	Reset	Description
31:4	reserved	WO	-	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	SS3	WO	-	SS3 Initiate Only a write by software is valid; a read of the register returns no meaningful data. When set by software, sampling is triggered on Sample Sequencer 3, assuming the Sequencer is enabled in the <b>ADCACTSS</b> register.
2	SS2	WO	-	SS2 Initiate Only a write by software is valid; a read of the register returns no meaningful data. When set by software, sampling is triggered on Sample Sequencer 2, assuming the Sequencer is enabled in the <b>ADCACTSS</b> register.
1	SS1	WO	-	SS1 Initiate Only a write by software is valid; a read of the register returns no meaningful data. When set by software, sampling is triggered on Sample Sequencer 1, assuming the Sequencer is enabled in the <b>ADCACTSS</b> register.
0	SS0	WO	-	SS0 Initiate Only a write by software is valid; a read of the register returns no meaningful data. When set by software, sampling is triggered on Sample Sequencer 0, assuming the Sequencer is enabled in the <b>ADCACTSS</b> register.

## Register 10: ADC Sample Averaging Control (ADCSAC), offset 0x030

This register controls the amount of hardware averaging applied to conversion results. The final conversion result stored in the FIFO is averaged from  $2^{\text{AVG}}$  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	0x00	Software should not rely on the value of 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	0	8th Sample Input Select  The MUX7 field is used during the eighth sample of a sequence executed with the Sample Sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion. The value set here indicates the corresponding pin, for example, a value of 1 indicates the input is ADC1.
27:26	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	0	7th Sample Input Select  The MUX6 field is used during the seventh sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
23:22	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	0	6th Sample Input Select  The MUX5 field is used during the sixth sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
19:18	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	0	5th Sample Input Select The MUX4 field is used during the fifth sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
15:14	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	0	4th Sample Input Select The MUX3 field is used during the fourth sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
11:10	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	0	3rd Sample Input Select The MUX2 field is used during the third sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
7:6	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	0	2nd Sample Input Select The MUX1 field is used during the second sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
3:2	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	0	1st Sample Input Select The MUX0 field is used during the first sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.

## Register 12: ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044

This register contains the configuration information for each sample for a sequence executed with Sample Sequencer 0. When configuring a sample sequence, the END bit must be set at some point, whether it be after the first sample, last sample, or any sample in between.

This register is 32-bits wide and contains information for eight possible samples.

### ADC Sample Sequence Control 0 (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  The TS7 bit is used during the eighth sample of the sample sequence and specifies the input source of the sample. If set, the temperature sensor is read. Otherwise, the input pin specified by the <b>ADCSSMUX</b> register is read.
30	IE7	R/W	0	8th Sample Interrupt Enable  The IE7 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, otherwise it is not. It is legal to have multiple samples within a sequence generate interrupts.
29	END7	R/W	0	8th Sample is End of Sequence  The 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.

Bit/Field	Name	Type	Reset	Description
10	IE2	R/W	0	3rd Sample Interrupt Enable Same definition as IE7 but used during the third sample.
9	END2	R/W	0	3rd Sample is End of Sequence Same definition as END7 but used during the third sample.
8	D2	R/W	0	3rd Sample Diff Input Select Same definition as D7 but used during the third sample.
7	TS1	R/W	0	2nd Sample Temp Sensor Select Same definition as TS7 but used during the second sample.
6	IE1	R/W	0	2nd Sample Interrupt Enable Same definition as IE7 but used during the second sample.
5	END1	R/W	0	2nd Sample is End of Sequence Same definition as END7 but used during the second sample.
4	D1	R/W	0	2nd Sample Diff Input Select Same definition as D7 but used during the second sample.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as TS7 but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as IE7 but used during the first sample.
1	END0	R/W	0	1st Sample is End of Sequence Same definition as END7 but used during the first sample. Since this sequencer has only one entry, this bit must be set.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as D7 but used during the first sample.

**Register 13: ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048****Register 14: ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068****Register 15: ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088****Register 16: ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8**

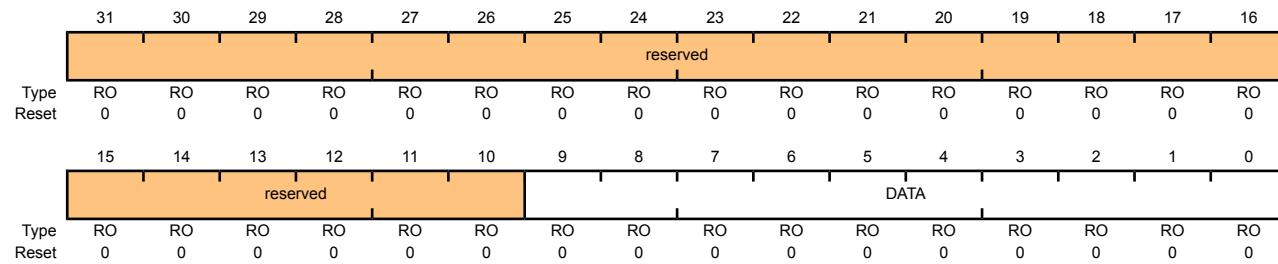
This register contains the conversion results for samples collected with the Sample Sequencer (the **ADCSSFIFO0** register is used for Sample Sequencer 0, **ADCSSFIFO1** for Sequencer 1, **ADCSSFIFO2** for Sequencer 2, and **ADCSSFIFO3** for Sequencer 3). Reads of this register return conversion result data in the order sample 0, sample 1, and so on, until the FIFO is empty. If the FIFO is not properly handled by software, overflow and underflow conditions are registered in the **ADCOSTAT** and **ADCUSTAT** registers.

## ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0)

Base 0x4003.8000

Offset 0x048

Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:0	DATA	RO	0x00	Conversion Result Data

**Register 17: ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C**

**Register 18: ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C**

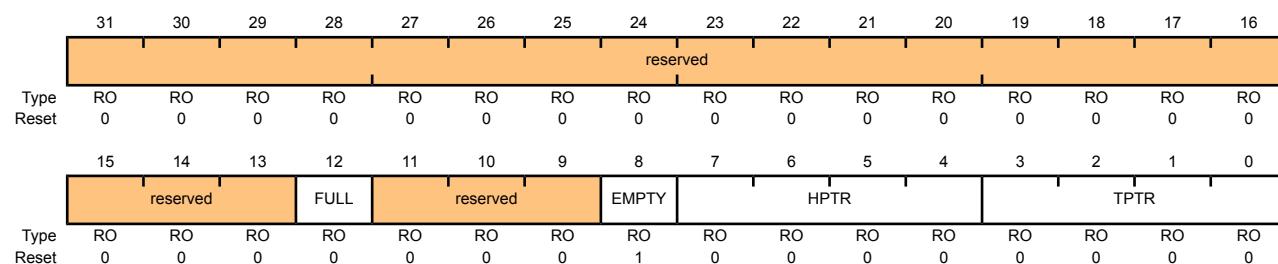
**Register 19: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C**

**Register 20: ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC**

This register provides a window into the Sample Sequencer, providing full/empty status information as well as the positions of the head and tail pointers. The reset value of 0x100 indicates an empty FIFO. The **ADCSSFSTAT0** register provides status on FIFO0, **ADCSSFSTAT1** on FIFO1, **ADCSSFSTAT2** on FIFO2, and **ADCSSFSTAT3** on FIFO3.

#### ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0)

Base 0x4003.8000  
Offset 0x04C  
Type RO, reset 0x0000.0100



Bit/Field	Name	Type	Reset	Description
31:13	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	FULL	RO	0	FIFO Full  When set, indicates that the FIFO is currently full.
11:9	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	EMPTY	RO	1	FIFO Empty  When set, indicates that the FIFO is currently empty.
7:4	HPTR	RO	0x00	FIFO Head Pointer  This field contains the current "head" pointer index for the FIFO, that is, the next entry to be written.
3:0	TPTR	RO	0x00	FIFO Tail Pointer  This field contains the current "tail" pointer index for the FIFO, that is, the next entry to be read.

## Register 21: ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060

## Register 22: ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080

This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 1 or 2. These registers are 16-bits wide and contain information for four possible samples. See the **ADCSSMUX0** register on page 284 for detailed bit descriptions.

### ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1)

Base 0x4003.8000  
Offset 0x060  
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	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	R/W	R/W												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<b>reserved</b>																
<b>MUX3</b>																
<b>reserved</b>																
<b>MUX2</b>																
<b>reserved</b>																
<b>MUX1</b>																
<b>reserved</b>																
<b>MUX0</b>																

Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13:12	MUX3	R/W	0	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	0	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	0	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	0	1st Sample Input Select

**Register 23: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064****Register 24: ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084**

These registers contain the configuration information for each sample for a sequence executed with Sample Sequencer 1 or 2. When configuring a sample sequence, the `END` bit must be set at some point, whether it be after the first sample, last sample, or any sample in between. This register is 16-bits wide and contains information for four possible samples. See the **ADCSSCTL0** register on page 286 for detailed bit descriptions.

## ADC Sample Sequence Control 1 (ADCSSCTL1)

Base 0x4003.8000

Offset 0x064

Type RO, reset 0x0000.0000

reserved																
Type	RO															
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	
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
31:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	TS3	R/W	0	4th Sample Temp Sensor Select Same definition as <code>TS7</code> but used during the fourth sample.
14	IE3	R/W	0	4th Sample Interrupt Enable Same definition as <code>IE7</code> but used during the fourth sample.
13	END3	R/W	0	4th Sample is End of Sequence Same definition as <code>END7</code> but used during the fourth sample.
12	D3	R/W	0	4th Sample Diff Input Select Same definition as <code>D7</code> but used during the fourth sample.
11	TS2	R/W	0	3rd Sample Temp Sensor Select Same definition as <code>TS7</code> but used during the third sample.
10	IE2	R/W	0	3rd Sample Interrupt Enable Same definition as <code>IE7</code> but used during the third sample.
9	END2	R/W	0	3rd Sample is End of Sequence Same definition as <code>END7</code> but used during the third sample.
8	D2	R/W	0	3rd Sample Diff Input Select Same definition as <code>D7</code> but used during the third sample.

Bit/Field	Name	Type	Reset	Description
7	TS1	R/W	0	2nd Sample Temp Sensor Select Same definition as TS7 but used during the second sample.
6	IE1	R/W	0	2nd Sample Interrupt Enable Same definition as IE7 but used during the second sample.
5	END1	R/W	0	2nd Sample is End of Sequence Same definition as END7 but used during the second sample.
4	D1	R/W	0	2nd Sample Diff Input Select Same definition as D7 but used during the second sample.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as TS7 but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as IE7 but used during the first sample.
1	END0	R/W	0	1st Sample is End of Sequence Same definition as END7 but used during the first sample. Since this sequencer has only one entry, this bit must be set.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as D7 but used during the first sample.

## Register 25: ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0

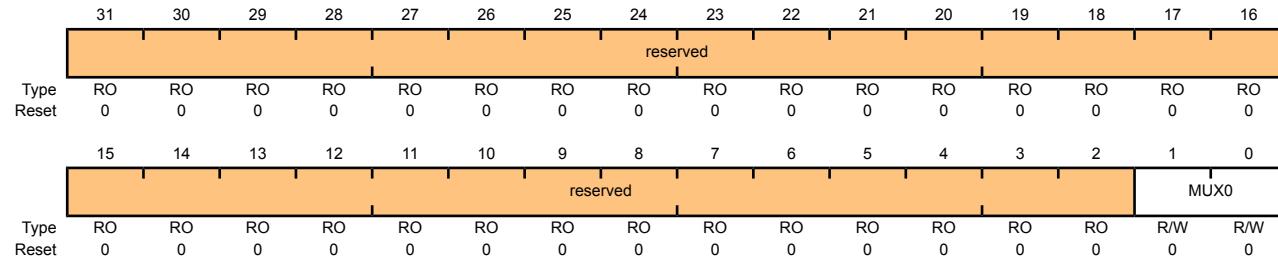
This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 3. This register is 4-bits wide and contains information for one possible sample. See the **ADCSSMUX0** register on page 284 for detailed bit descriptions.

### 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	0x00	Software should not rely on the value of 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

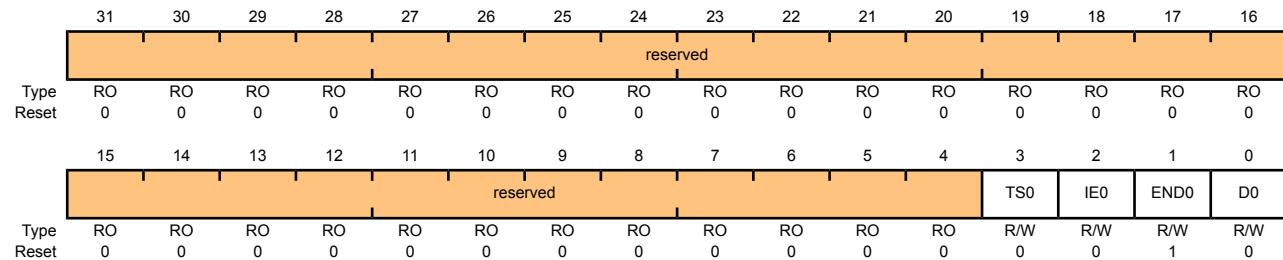
This register contains the configuration information for each sample for a sequence executed with Sample Sequencer 3. The `END` bit is always set since there is only one sample in this sequencer. This register is 4-bits wide and contains information for one possible sample. See the **ADCSSCTL0** register on page 286 for detailed bit descriptions.

### ADC Sample Sequence Control 3 (ADCSSCTL3)

Base 0x4003.8000

Offset 0x0A4

Type R/W, reset 0x0000.0002



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as <code>TS7</code> but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as <code>IE7</code> but used during the first sample.
1	END0	R/W	1	1st Sample is End of Sequence Same definition as <code>END7</code> but used during the first sample. Since this sequencer has only one entry, this bit must be set.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as <code>D7</code> but used during the first sample.

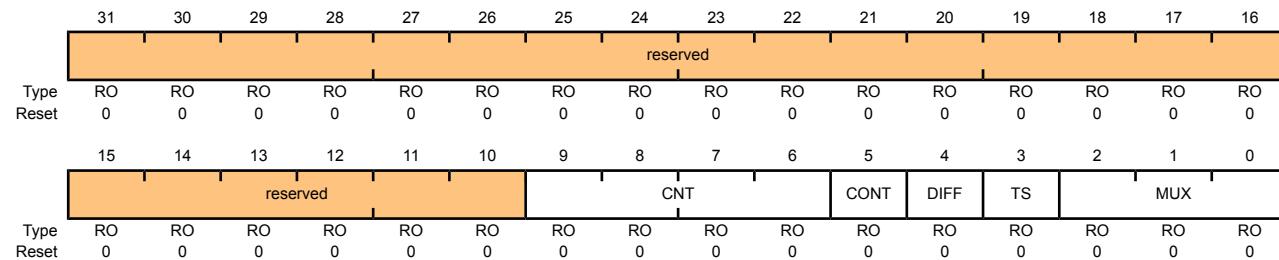
## Register 27: ADC Test Mode Loopback (ADCTMLB), offset 0x100

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.

### Read-Only Register

#### ADC Test Mode Loopback (ADCTMLB)

Base 0x4003.8000  
Offset 0x100  
Type RO, reset 0x0000.0000

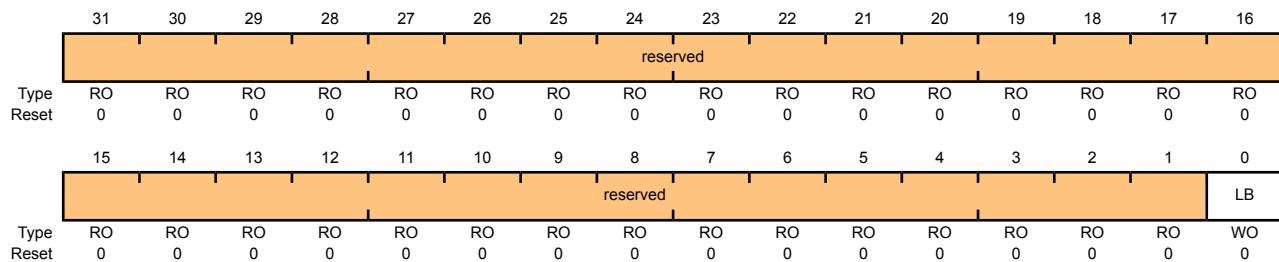


Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:6	CNT	RO	0x0	Continuous Sample Counter  Continuous 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	RO	0	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	RO	0	Differential Sample Indicator  When set, indicates that this is a differential sample.
3	TS	RO	0	Temp Sensor Sample Indicator  When set, indicates that this is a temperature sensor sample.
2:0	MUX	RO	0x0	Analog Input Indicator  Indicates which analog input is to be sampled.

## Write-Only Register

### ADC Test Mode Loopback (ADCTMLB)

Base 0x4003.8000  
Offset 0x100  
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	LB	WO	0	<p>Loopback Mode Enable</p> <p>When set, forces a loopback within the digital block to provide information on input and unique numbering.</p> <p>The 10-bit loopback data is defined as shown in the read for bits 9:0 above.</p>

## 13 Universal Asynchronous Receivers/Transmitters (UARTs)

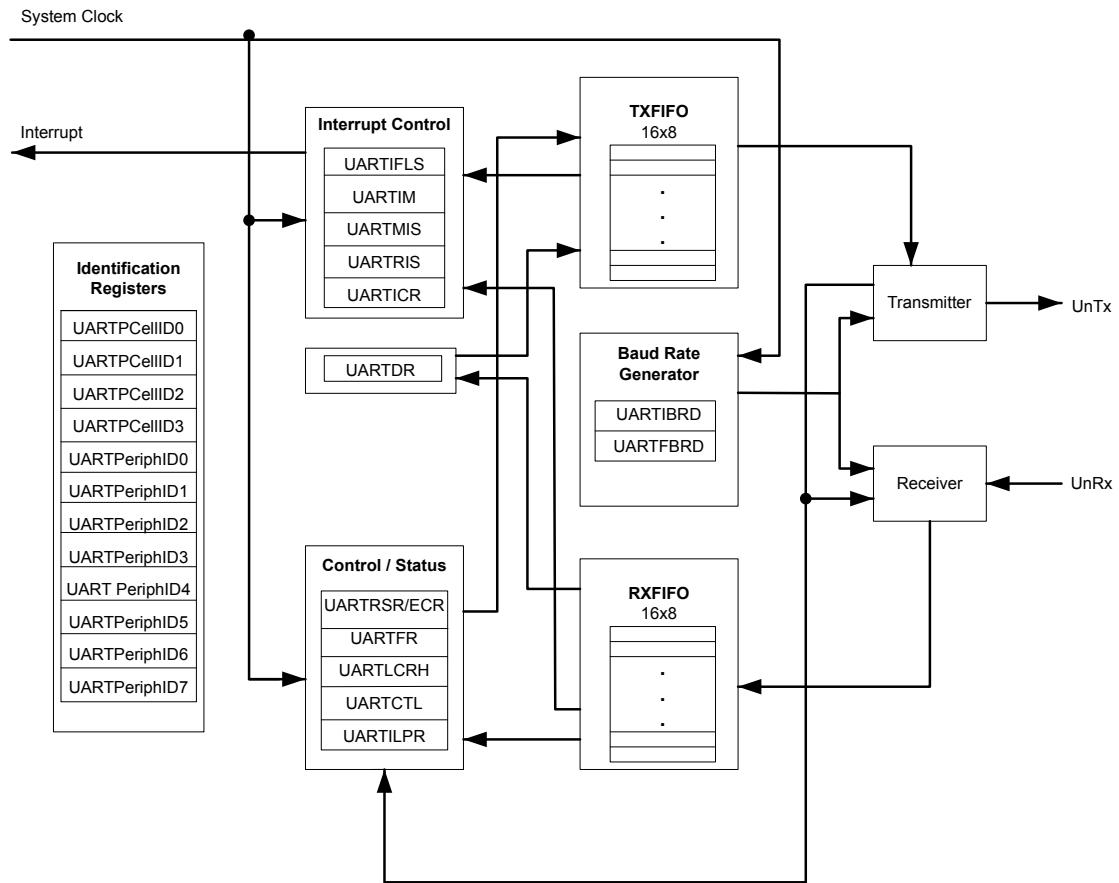
The Stellaris® Universal Asynchronous Receiver/Transmitter (UART) provides fully programmable, 16C550-type serial interface characteristics. The LM3S8962 controller is equipped with two UART modules.

Each UART has the following features:

- Separate transmit and receive FIFOs
- Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
- FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
- Programmable baud-rate generator allowing rates up to 3.125 Mbps
- Standard asynchronous communication bits for start, stop, and parity
- False start bit detection
- Line-break generation and detection
- Fully programmable serial interface characteristics:
  - 5, 6, 7, or 8 data bits
  - Even, odd, stick, or no-parity bit generation/detection
  - 1 or 2 stop bit generation
- IrDA serial-IR (SIR) encoder/decoder providing:
  - Programmable use of IrDA Serial InfraRed (SIR) or UART input/output
  - Support of IrDA SIR encoder/decoder functions for data rates up to 115.2 Kbps half-duplex
  - Support of normal 3/16 and low-power (1.41-2.23  $\mu$ s) bit durations
  - Programmable internal clock generator enabling division of reference clock by 1 to 256 for low-power mode bit duration

## 13.1 Block Diagram

Figure 13-1. UART Module Block Diagram



## 13.2 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 317). Transmit and receive are both enabled out of reset. Before any control registers are programmed, the UART must be disabled by clearing the UARTEN bit in **UARTCTL**. If the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

The UART peripheral also includes a serial IR (SIR) encoder/decoder block that can be connected to an infrared transceiver to implement an IrDA SIR physical layer. The SIR function is programmed using the **UARTCTL** register.

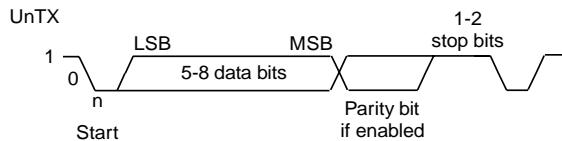
### 13.2.1 Transmit/Receive Logic

The transmit logic performs parallel-to-serial conversion on the data read from the transmit FIFO. The control logic outputs the serial bit stream beginning with a start bit, and followed by the data

bits (LSB first), parity bit, and the stop bits according to the programmed configuration in the control registers. See Figure 13-2 on page 300 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 13-2. UART Character Frame**



### 13.2.2 Baud-Rate Generation

The baud-rate divisor is a 22-bit number consisting of a 16-bit integer and a 6-bit fractional part. The number formed by these two values is used by the baud-rate generator to determine the bit period. Having a fractional baud-rate divider allows the UART to generate all the standard baud rates.

The 16-bit integer is loaded through the **UART Integer Baud-Rate Divisor (UARTIBRD)** register (see page 313) and the 6-bit fractional part is loaded with the **UART Fractional Baud-Rate Divisor (UARTFBRD)** register (see page 314). 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.):

$$\text{BRD} = \text{BRDI} + \text{BRDF} = \text{SysClk} / (16 * \text{Baud Rate})$$

The 6-bit fractional number (that is to be loaded into the DIVFRAC bit field in the **UARTFBRD** register) can be calculated by taking the fractional part of the baud-rate divisor, multiplying it by 64, and adding 0.5 to account for rounding errors:

$$\text{UARTFBRD}[\text{DIVFRAC}] = \text{integer}(\text{BRDF} * 64 + 0.5)$$

The UART generates an internal baud-rate reference clock at 16x the baud-rate (referred to as Baud16). This reference clock is divided by 16 to generate the transmit clock, and is used for error detection during receive operations.

Along with the **UART Line Control, High Byte (UARTLCRH)** register (see page 315), 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

### 13.2.3 Data Transmission

Data received or transmitted is stored in two 16-byte FIFOs, though the receive FIFO has an extra four bits per character for status information. For transmission, data is written into the transmit FIFO. If the UART is enabled, it causes a data frame to start transmitting with the parameters indicated in the **UARTLCRH** register. Data continues to be transmitted until there is no data left in the transmit FIFO. The **BUSY** bit in the **UART Flag (UARTFR)** register (see page 310) 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 299).

The start bit is valid if UnRx is still low on the eighth cycle of Baud16, otherwise a false start bit is detected and it is ignored. Start bit errors can be viewed in the **UART Receive Status (UARTRSR)** register (see page 308). If the start bit was valid, successive data bits are sampled on every 16th cycle of Baud16 (that is, one bit period later) according to the programmed length of the data characters. The parity bit is then checked if parity mode was enabled. Data length and parity are defined in the **UARTLCRH** register.

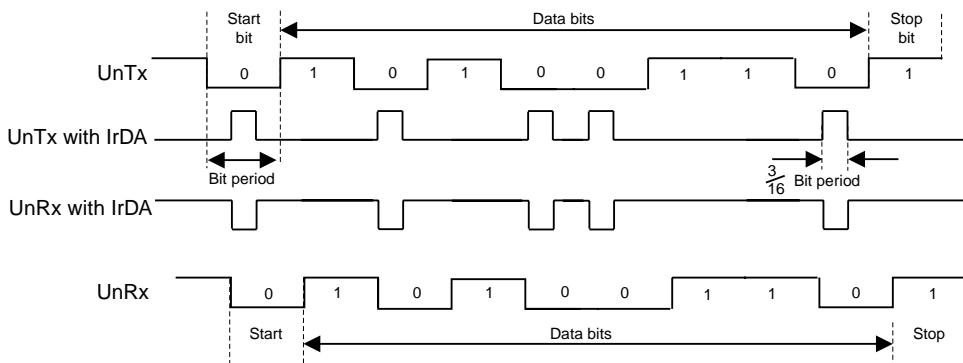
Lastly, a valid stop bit is confirmed if UnRx is High, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO, with any error bits associated with that word.

### 13.2.4 Serial IR (SIR)

The UART peripheral includes an IrDA serial-IR (SIR) encoder/decoder block. The IrDA SIR block provides functionality that converts between an asynchronous UART data stream, and half-duplex serial SIR interface. No analog processing is performed on-chip. The role of the SIR block is to provide a digital encoded output, and decoded input to the UART. The UART signal pins can be connected to an infrared transceiver to implement an IrDA SIR physical layer link. The SIR block has two modes of operation:

- In normal IrDA mode, a zero logic level is transmitted as high pulse of 3/16th duration of the selected baud rate bit period on the output pin, while logic one levels are transmitted as a static LOW signal. These levels control the driver of an infrared transmitter, sending a pulse of light for each zero. On the reception side, the incoming light pulses energize the photo transistor base of the receiver, pulling its output LOW. This drives the UART input pin LOW.
- In low-power IrDA mode, the width of the transmitted infrared pulse is set to three times the period of the internally generated IrLPBaud16 signal ( $1.63\ \mu s$ , assuming a nominal 1.8432 MHz frequency) by changing the appropriate bit in the **UARTCR** register.

Figure 13-3 on page 302 shows the UART transmit and receive signals, with and without IrDA modulation.

**Figure 13-3. IrDA Data Modulation**

In both normal and low-power IrDA modes:

- During transmission, the UART data bit is used as the base for encoding
- During reception, the decoded bits are transferred to the UART receive logic

The IrDA SIR physical layer specifies a half-duplex communication link, with a minimum 10 ms delay between transmission and reception. This delay must be generated by software because it is not automatically supported by the UART. The delay is required because the infrared receiver electronics might become biased, or even saturated from the optical power coupled from the adjacent transmitter LED. This delay is known as latency, or receiver setup time.

### 13.2.5 FIFO Operation

The UART has two 16-entry FIFOs; one for transmit and one for receive. Both FIFOs are accessed via the **UART Data (UARTDR)** register (see page 306). 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 315).

FIFO status can be monitored via the **UART Flag (UARTFR)** register (see page 310) 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 319). 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.

### 13.2.6 Interrupts

The UART can generate interrupts when the following conditions are observed:

- Overrun Error
- Break Error

- Parity Error
- Framing Error
- Receive Timeout
- Transmit (when condition defined in the TXIFLSEL bit in the **UARTIFLS** register is met)
- Receive (when condition defined in the RXIFLSEL bit in the **UARTIFLS** register is met)

All of the interrupt events are ORed together before being sent to the interrupt controller, so the UART can only generate a single interrupt request to the controller at any given time. Software can service multiple interrupt events in a single interrupt service routine by reading the **UART Masked Interrupt Status (UARTMIS)** register (see page 324).

The interrupt events that can trigger a controller-level interrupt are defined in the **UART Interrupt Mask (UARTIM)** register (see page 321) 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 323).

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 325).

The receive timeout interrupt is asserted when the receive FIFO is not empty, and no further data is received over a 32-bit period. The receive timeout interrupt is cleared either when the FIFO becomes empty through reading all the data (or by reading the holding register), or when a 1 is written to the corresponding bit in the **UARTICR** register.

### 13.2.7 Loopback Operation

The UART can be placed into an internal loopback mode for diagnostic or debug work. This is accomplished by setting the **LBE** bit in the **UARTCTL** register (see page 317). In loopback mode, data transmitted on UnTx is received on the UnRx input.

### 13.2.8 IrDA SIR block

The IrDA SIR block contains an IrDA serial IR (SIR) protocol encoder/decoder. When enabled, the SIR block uses the **UnTx** and **UnRx** pins for the SIR protocol, which should be connected to an IR transceiver.

The SIR block can receive and transmit, but it is only half-duplex so it cannot do both at the same time. Transmission must be stopped before data can be received. The IrDA SIR physical layer specifies a minimum 10-ms delay between transmission and reception.

## 13.3 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 for using a UART module. For this example, the system 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 300, 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 313) should be set to 10. The value to be loaded into the **UARTFBRD** register (see page 314) 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.

## 13.4 Register Map

Table 13-1 on page 304 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:** The UART must be disabled (see the **UARTEN** bit in the **UARTCTL** register on page 317) 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 13-1. UART Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	UARTDR	R/W	0x0000.0000	UART Data	306
0x004	UARTRSR/UARTECR	R/W	0x0000.0000	UART Receive Status/Error Clear	308
0x018	UARTFR	RO	0x0000.0090	UART Flag	310
0x020	UARTILPR	R/W	0x0000.0000	UART IrDA Low-Power Register	312
0x024	UARTIBRD	R/W	0x0000.0000	UART Integer Baud-Rate Divisor	313

Offset	Name	Type	Reset	Description	See page
0x028	UARTFBRD	R/W	0x0000.0000	UART Fractional Baud-Rate Divisor	314
0x02C	UARTLCRH	R/W	0x0000.0000	UART Line Control	315
0x030	UARTCTL	R/W	0x0000.0300	UART Control	317
0x034	UARTIFLS	R/W	0x0000.0012	UART Interrupt FIFO Level Select	319
0x038	UARTIM	R/W	0x0000.0000	UART Interrupt Mask	321
0x03C	UARTRIS	RO	0x0000.000F	UART Raw Interrupt Status	323
0x040	UARTMIS	RO	0x0000.0000	UART Masked Interrupt Status	324
0x044	UARTICR	W1C	0x0000.0000	UART Interrupt Clear	325
0xFD0	UARTPeriphID4	RO	0x0000.0000	UART Peripheral Identification 4	327
0xFD4	UARTPeriphID5	RO	0x0000.0000	UART Peripheral Identification 5	328
0xFD8	UARTPeriphID6	RO	0x0000.0000	UART Peripheral Identification 6	329
0xFDC	UARTPeriphID7	RO	0x0000.0000	UART Peripheral Identification 7	330
0xFE0	UARTPeriphID0	RO	0x0000.0011	UART Peripheral Identification 0	331
0xFE4	UARTPeriphID1	RO	0x0000.0000	UART Peripheral Identification 1	332
0xFE8	UARTPeriphID2	RO	0x0000.0018	UART Peripheral Identification 2	333
0xFEC	UARTPeriphID3	RO	0x0000.0001	UART Peripheral Identification 3	334
0xFF0	UARTPCellID0	RO	0x0000.000D	UART PrimeCell Identification 0	335
0xFF4	UARTPCellID1	RO	0x0000.00F0	UART PrimeCell Identification 1	336
0xFF8	UARTPCellID2	RO	0x0000.0005	UART PrimeCell Identification 2	337
0xFFC	UARTPCellID3	RO	0x0000.00B1	UART PrimeCell Identification 3	338

## 13.5 Register Descriptions

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

## Register 1: UART Data (UARTDR), offset 0x000

This register is the data register (the interface to the FIFOs).

When FIFOs are enabled, data written to this location is pushed onto the transmit FIFO. If FIFOs are disabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). A write to this register initiates a transmission from the UART.

For received data, if the FIFO is enabled, the data byte and the 4-bit status (break, frame, parity, and overrun) is pushed onto the 12-bit wide receive FIFO. If FIFOs are disabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO). The received data can be retrieved by reading this register.

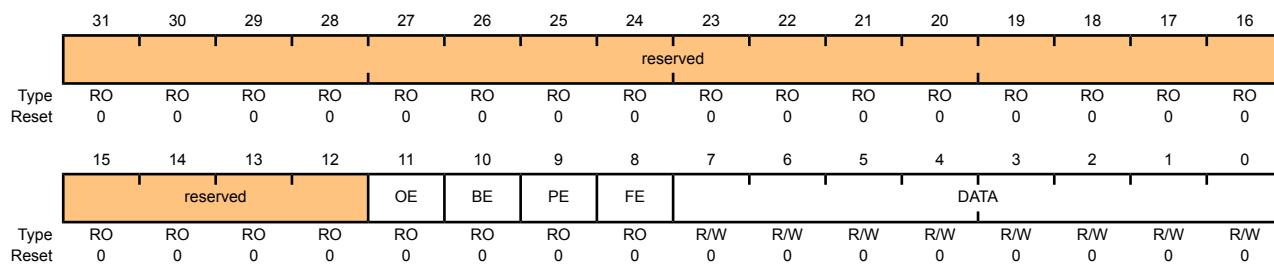
### UART Data (UARTDR)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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	<p>UART Parity Error</p> <p>This bit is set to 1 when the parity of the received data character does not match the parity defined by bits 2 and 7 of the <b>UARTLCRH</b> register.</p> <p>In FIFO mode, this error is associated with the character at the top of the FIFO.</p>
8	FE	RO	0	<p>UART Framing Error</p> <p>This bit is set to 1 when the received character does not have a valid stop bit (a valid stop bit is 1).</p>
7:0	DATA	R/W	0	<p>Data Transmitted or Received</p> <p>When written, the data that is to be transmitted via the UART. When read, the data that was received by the UART.</p>

## Register 2: UART Receive Status/Error Clear (UARTRSR/UARTECR), offset 0x004

The **UARTRSR/UARTECR** register is the receive status register/error clear register.

In addition to the **UARTDR** register, receive status can also be read from the **UARTRSR** register. If the status is read from this register, then the status information corresponds to the entry read from **UARTDR** prior to reading **UARTRSR**. The status information for overrun is set immediately when an overrun condition occurs.

The **UARTRSR** register cannot be written.

A write of any value to the **UARTECR** register clears the framing, parity, break, and overrun errors. All the bits are cleared to 0 on reset.

### Read-Only Receive Status (UARTRSR) Register

#### UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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	<p>UART Parity Error</p> <p>This bit is set to 1 when the parity of the received data character does not match the parity defined by bits 2 and 7 of the <b>UARTLCRH</b> register.</p> <p>This bit is cleared to 0 by a write to <b>UARTECR</b>.</p>
0	FE	RO	0	<p>UART Framing Error</p> <p>This bit is set to 1 when the received character does not have a valid stop bit (a valid stop bit is 1).</p> <p>This bit is cleared to 0 by a write to <b>UARTECR</b>.</p> <p>In FIFO mode, this error is associated with the character at the top of the FIFO.</p>

### Write-Only Error Clear (UARTECR) Register

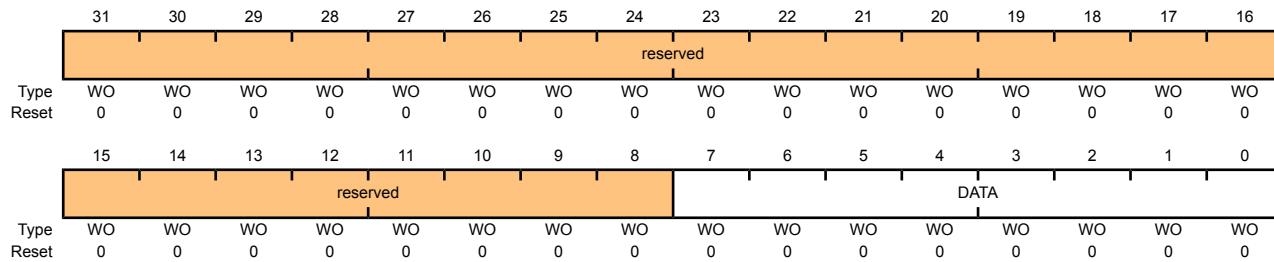
UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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	<p>Error Clear</p> <p>A write to this register of any data clears the framing, parity, break, and overrun flags.</p>

### Register 3: UART Flag (UARTFR), offset 0x018

The **UARTFR** register is the flag register. After reset, the TXFF, RXFF, and BUSY bits are 0, and TXFE and RXFE bits are 1.

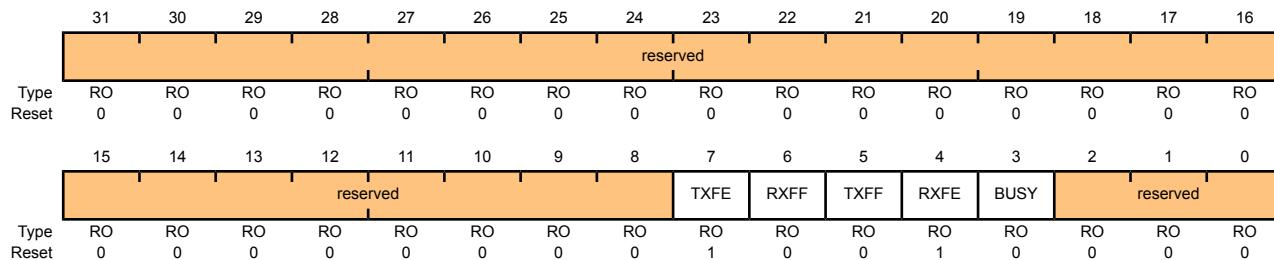
#### UART Flag (UARTFR)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

Offset 0x018

Type RO, reset 0x0000.0090



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TXFE	RO	1	UART Transmit FIFO Empty  The meaning of this bit depends on the state of the 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.

Bit/Field	Name	Type	Reset	Description
4	RXFE	RO	1	<p>UART Receive FIFO Empty</p> <p>The meaning of this bit depends on the state of the <b>FEN</b> bit in the <b>UARTLCRH</b> register.</p> <p>If the FIFO is disabled, this bit is set when the receive holding register is empty.</p> <p>If the FIFO is enabled, this bit is set when the receive FIFO is empty.</p>
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 IrDA Low-Power Register (UARTILPR), offset 0x020

The **UARTILPR** register is an 8-bit read/write register that stores the low-power counter divisor value used to generate the `IrLPBaud16` signal by dividing down the system clock (SysClk). All the bits are cleared to 0 when reset.

The `IrLPBaud16` internal signal is generated by dividing down the `UARTCLK` signal according to the low-power divisor value written to **UARTILPR**. The low-power divisor value is calculated as follows:

$$\text{ILPDVSR} = \text{SysClk} / F_{\text{IrLPBaud16}}$$

where  $F_{\text{IrLPBaud16}}$  is nominally 1.8432 MHz.

$\text{IrLPBaud16}$  is an internal signal used for SIR pulse generation when low-power mode is used. You must choose the divisor so that  $1.42 \text{ MHz} < F_{\text{IrLPBaud16}} < 2.12 \text{ MHz}$ , which results in a low-power pulse duration of  $1.41\text{--}2.11 \mu\text{s}$  (three times the period of `IrLPBaud16`). The minimum frequency of `IrLPBaud16` ensures that pulses less than one period of `IrLPBaud16` are rejected, but that pulses greater than  $1.4 \mu\text{s}$  are accepted as valid pulses.

**Note:** Zero is an illegal value. Programming a zero value results in no `IrLPBaud16` pulses being generated.

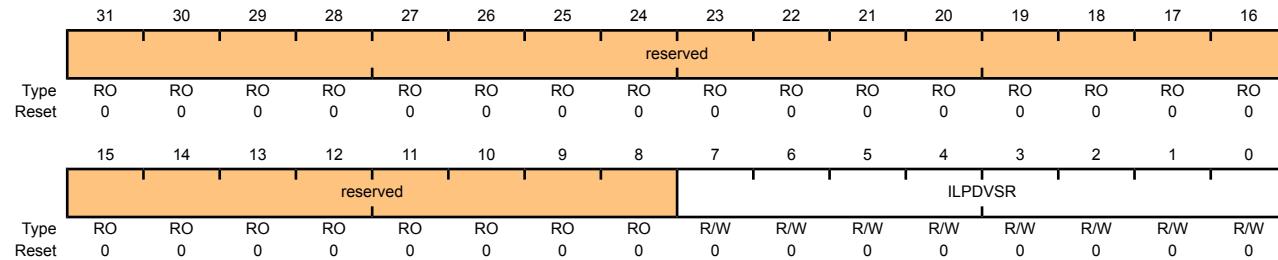
### UART IrDA Low-Power Register (UARTILPR)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

Offset 0x020

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ILPDVSR	R/W	0x00	IrDA Low-Power Divisor This is an 8-bit low-power divisor value.

## Register 5: UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024

The **UARTIBRD** register is the integer part of the baud-rate divisor value. All the bits are cleared on reset. The minimum possible divide ratio is 1 (when **UARTIBRD**=0), in which case the **UARTFBRD** register is ignored. When changing the **UARTIBRD** register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register. See “Baud-Rate Generation” on page 300 for configuration details.

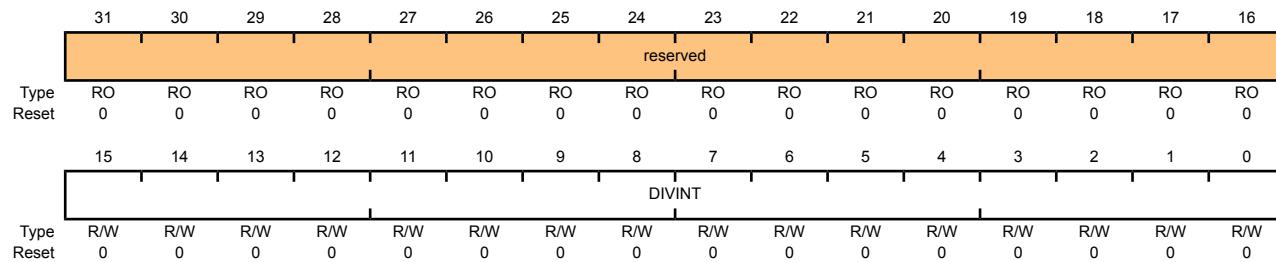
### UART Integer Baud-Rate Divisor (UARTIBRD)

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 6: UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028

The **UARTFBRD** register is the fractional part of the baud-rate divisor value. All the bits are cleared on reset. When changing the **UARTFBRD** register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register. See “Baud-Rate Generation” on page 300 for configuration details.

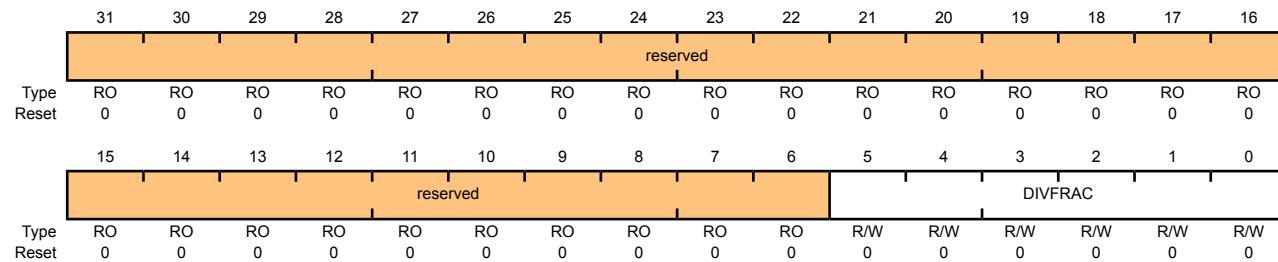
UART Fractional Baud-Rate Divisor (UARTFBRD)

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 7: UART Line Control (UARTLCRH), offset 0x02C

The **UARTLCRH** register is the line control register. Serial parameters such as data length, parity, and stop bit selection are implemented in this register.

When updating the baud-rate divisor (**UARTIBRD** and/or **UARTIFRD**), the **UARTLCRH** register must also be written. The write strobe for the baud-rate divisor registers is tied to the **UARTLCRH** register.

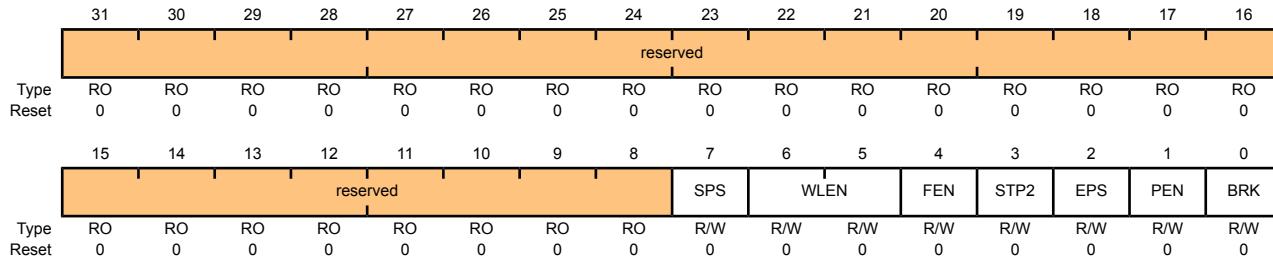
### UART Line Control (UARTLCRH)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

Offset 0x02C

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description										
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
7	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	WLLEN	R/W	0	UART Word Length The bits indicate the number of data bits transmitted or received in a frame as follows: <table border="1"><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0x3</td><td>8 bits</td></tr><tr><td>0x2</td><td>7 bits</td></tr><tr><td>0x1</td><td>6 bits</td></tr><tr><td>0x0</td><td>5 bits (default)</td></tr></tbody></table>	Value	Description	0x3	8 bits	0x2	7 bits	0x1	6 bits	0x0	5 bits (default)
Value	Description													
0x3	8 bits													
0x2	7 bits													
0x1	6 bits													
0x0	5 bits (default)													
4	FEN	R/W	0	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.										

Bit/Field	Name	Type	Reset	Description
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.
2	EPS	R/W	0	UART Even Parity Select  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 PEN bit.
1	PEN	R/W	0	UART Parity Enable  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	UART Send Break  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 8: UART Control (UARTCTL), offset 0x030

The **UARTCTL** register is the control register. All the bits are cleared on reset except for the Transmit Enable (TXE) and Receive Enable (RXE) bits, which are set to 1.

To enable the UART module, the **UARTEN** bit must be set to 1. If software requires a configuration change in the module, the **UARTEN** bit must be cleared before the configuration changes are written. If the UART is disabled during a transmit or receive operation, the current transaction is completed prior to the UART stopping.

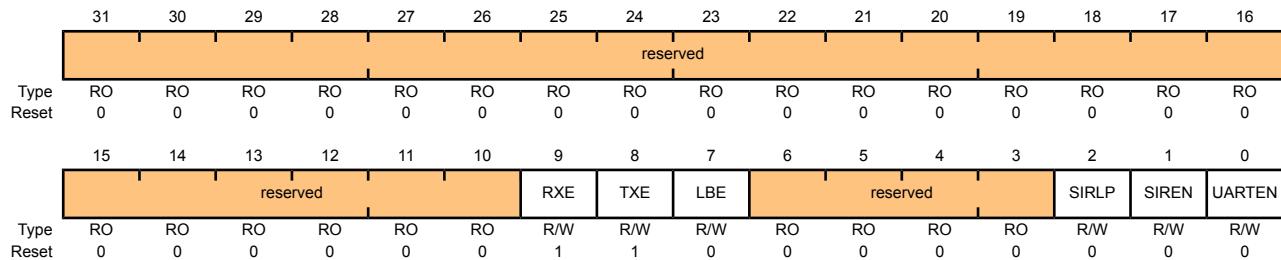
### UART Control (UARTCTL)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

Offset 0x030

Type R/W, reset 0x0000.0300



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.
7	LBE	R/W	0	UART Loop Back Enable  If this bit is set to 1, the <b>UnTX</b> path is fed through the <b>UnRX</b> path.
6:3	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.

Bit/Field	Name	Type	Reset	Description
2	SIRLP	R/W	0	<b>UART SIR Low Power Mode</b>  This bit selects the IrDA encoding mode. If this bit is cleared to 0, low-level bits are transmitted as an active High pulse with a width of 3/16th of the bit period. If this bit is set to 1, low-level bits are transmitted with a pulse width which is 3 times the period of the <code>IrLPBaud16</code> input signal, regardless of the selected bit rate. Setting this bit uses less power, but might reduce transmission distances. See page 312 for more information.
1	SIREN	R/W	0	<b>UART SIR Enable</b>  If this bit is set to 1, the IrDA SIR block is enabled, and the UART will transmit and receive data using SIR protocol.
0	UARTEN	R/W	0	<b>UART Enable</b>  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 9: UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034

The **UARTIFLS** register is the interrupt FIFO level select register. You can use this register to define the FIFO level at which the TXRIS and RXRIS bits in the **UARTRIS** register are triggered.

The interrupts are generated based on a transition through a level rather than being based on the level. That is, the interrupts are generated when the fill level progresses through the trigger level. For example, if the receive trigger level is set to the half-way mark, the interrupt is triggered as the module is receiving the 9th character.

Out of reset, the TXIFLSEL and RXIFLSEL bits are configured so that the FIFOs trigger an interrupt at the half-way mark.

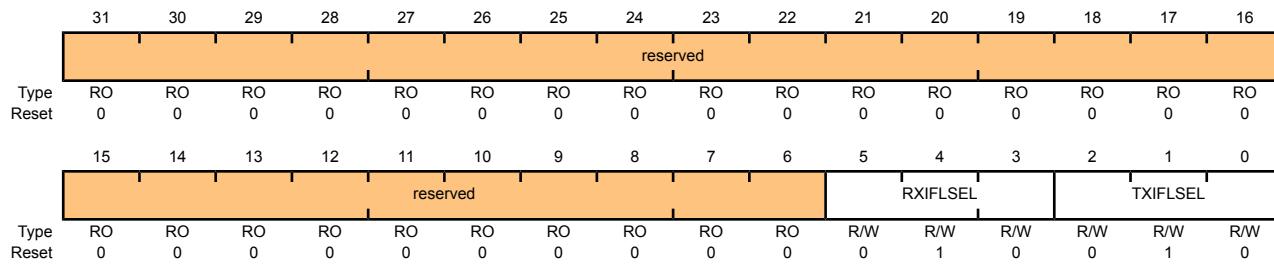
### UART Interrupt FIFO Level Select (UARTIFLS)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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 $\geq$ 1/8 full
0x1	RX FIFO $\geq$ 1/4 full
0x2	RX FIFO $\geq$ 1/2 full (default)
0x3	RX FIFO $\geq$ 3/8 full
0x4	RX FIFO $\geq$ 7/8 full
0x5-0x7	Reserved

Bit/Field	Name	Type	Reset	Description
2:0	TXIFLSEL	R/W	0x2	UART Transmit Interrupt FIFO Level Select The trigger points for the transmit interrupt are as follows:
Value Description				
0x0 TX FIFO $\leq$ 1/8 full				
0x1 TX FIFO $\leq$ 1/4 full				
0x2 TX FIFO $\leq$ 1/2 full (default)				
0x3 TX FIFO $\leq$ 3/4 full				
0x4 TX FIFO $\leq$ 7/8 full				
0x5-0x7 Reserved				

## Register 10: UART Interrupt Mask (UARTIM), offset 0x038

The **UARTIM** register is the interrupt mask set/clear register.

On a read, this register gives the current value of the mask on the relevant interrupt. Writing a 1 to a bit allows the corresponding raw interrupt signal to be routed to the interrupt controller. Writing a 0 prevents the raw interrupt signal from being sent to the interrupt controller.

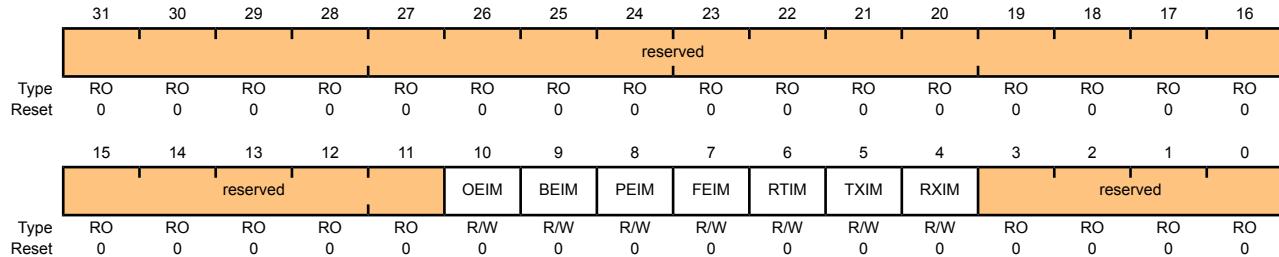
### UART Interrupt Mask (UARTIM)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

Offset 0x038

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEIM	R/W	0	UART Overrun Error Interrupt Mask On a read, the current mask for the <b>OEIM</b> interrupt is returned. Setting this bit to 1 promotes the <b>OEIM</b> interrupt to the interrupt controller.
9	BEIM	R/W	0	UART Break Error Interrupt Mask On a read, the current mask for the <b>BEIM</b> interrupt is returned. Setting this bit to 1 promotes the <b>BEIM</b> interrupt to the interrupt controller.
8	PEIM	R/W	0	UART Parity Error Interrupt Mask On a read, the current mask for the <b>PEIM</b> interrupt is returned. Setting this bit to 1 promotes the <b>PEIM</b> interrupt to the interrupt controller.
7	FEIM	R/W	0	UART Framing Error Interrupt Mask On a read, the current mask for the <b>FEIM</b> interrupt is returned. Setting this bit to 1 promotes the <b>FEIM</b> interrupt to the interrupt controller.
6	RTIM	R/W	0	UART Receive Time-Out Interrupt Mask On a read, the current mask for the <b>RTIM</b> interrupt is returned. Setting this bit to 1 promotes the <b>RTIM</b> interrupt to the interrupt controller.
5	TXIM	R/W	0	UART Transmit Interrupt Mask On a read, the current mask for the <b>TXIM</b> interrupt is returned. Setting this bit to 1 promotes the <b>TXIM</b> interrupt to the interrupt controller.

Bit/Field	Name	Type	Reset	Description
4	RXIM	R/W	0	UART Receive Interrupt Mask On a read, the current mask for the RXIM interrupt is returned. Setting this bit to 1 promotes the RXIM interrupt to the interrupt controller.
3:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 11: UART Raw Interrupt Status (UARTRIS), offset 0x03C

The **UARTRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt. A write has no effect.

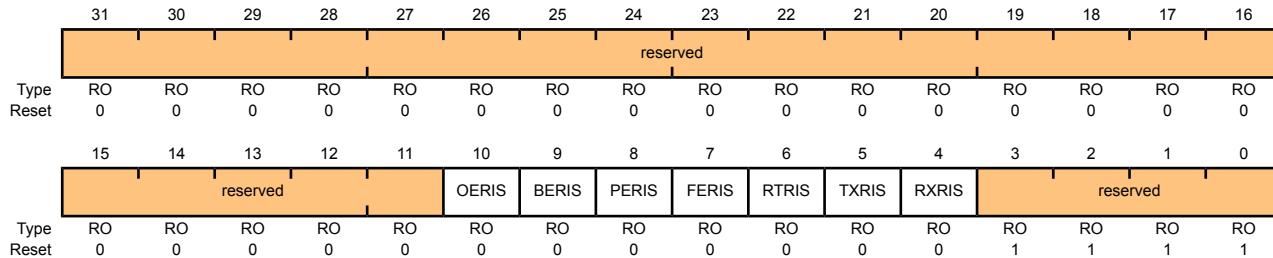
### UART Raw Interrupt Status (UARTRIS)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

Offset 0x03C

Type RO, reset 0x0000.000F



Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OERIS	RO	0	UART Overrun Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
9	BERIS	RO	0	UART Break Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
8	PERIS	RO	0	UART Parity Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
7	FERIS	RO	0	UART Framing Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
6	RTRIS	RO	0	UART Receive Time-Out Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
5	TXRIS	RO	0	UART Transmit Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
4	RXRIS	RO	0	UART Receive Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
3:0	reserved	RO	0xF	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 12: UART Masked Interrupt Status (UARTMIS), offset 0x040

The **UARTMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

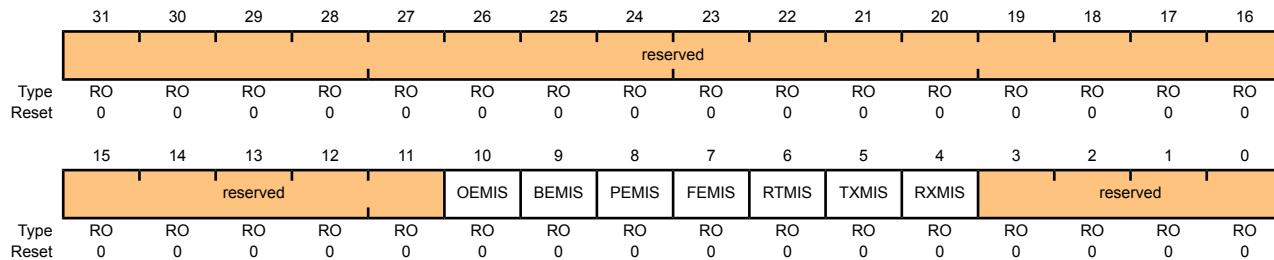
### UART Masked Interrupt Status (UARTMIS)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

Offset 0x040

Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEMIS	RO	0	UART Overrun Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
9	BEMIS	RO	0	UART Break Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
8	PEMIS	RO	0	UART Parity Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
7	FEMIS	RO	0	UART Framing Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
6	RTMIS	RO	0	UART Receive Time-Out Masked Interrupt Status Gives the masked interrupt state of this interrupt.
5	TXMIS	RO	0	UART Transmit Masked Interrupt Status Gives the masked interrupt state of this interrupt.
4	RXMIS	RO	0	UART Receive Masked Interrupt Status Gives the masked interrupt state of this interrupt.
3:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 13: UART Interrupt Clear (UARTICR), offset 0x044

The **UARTICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt (both raw interrupt and masked interrupt, if enabled) is cleared. A write of 0 has no effect.

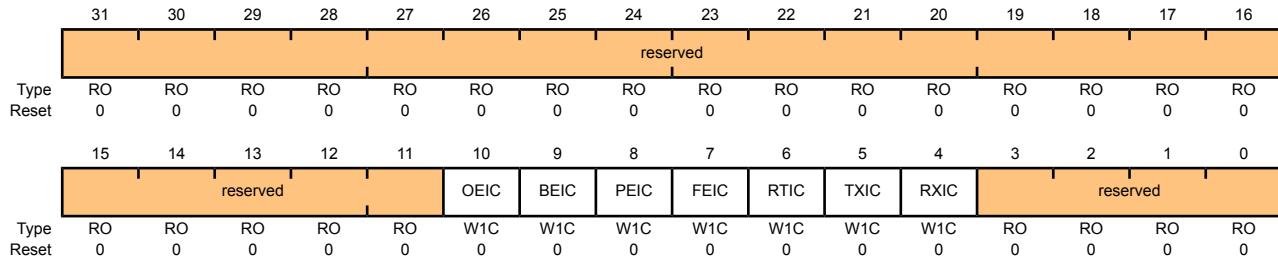
### UART Interrupt Clear (UARTICR)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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 **OEIC** 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 **BEIC** 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 **PEIC** values are defined as follows:

Value	Description
0	No effect on the interrupt.
1	Clears interrupt.

Bit/Field	Name	Type	Reset	Description
7	FEIC	W1C	0	Framing Error Interrupt Clear The FEIC values are defined as follows:  Value Description 0 No effect on the interrupt. 1 Clears interrupt.
6	RTIC	W1C	0	Receive Time-Out Interrupt Clear The RTIC values are defined as follows:  Value Description 0 No effect on the interrupt. 1 Clears interrupt.
5	TXIC	W1C	0	Transmit Interrupt Clear The TXIC values are defined as follows:  Value Description 0 No effect on the interrupt. 1 Clears interrupt.
4	RXIC	W1C	0	Receive Interrupt Clear The RXIC values are defined as follows:  Value Description 0 No effect on the interrupt. 1 Clears interrupt.
3:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 14: UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

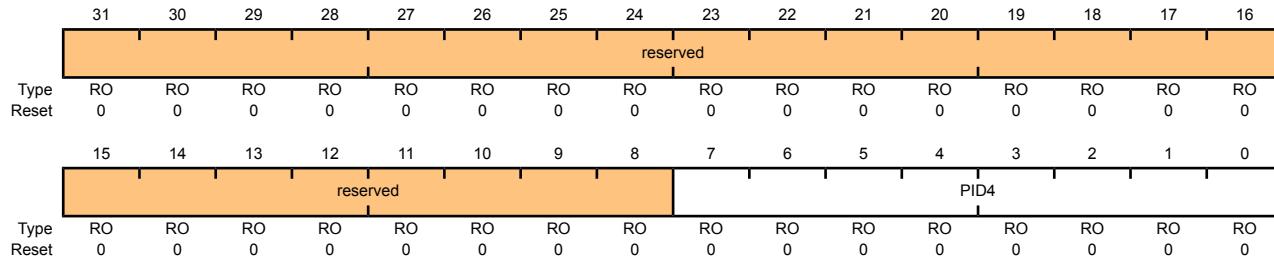
### UART Peripheral Identification 4 (UARTPeriphID4)

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 15: UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

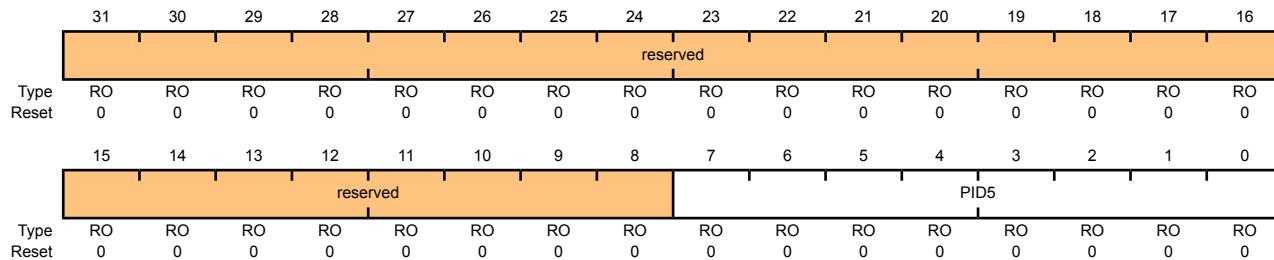
### UART Peripheral Identification 5 (UARTPeriphID5)

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 16: UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

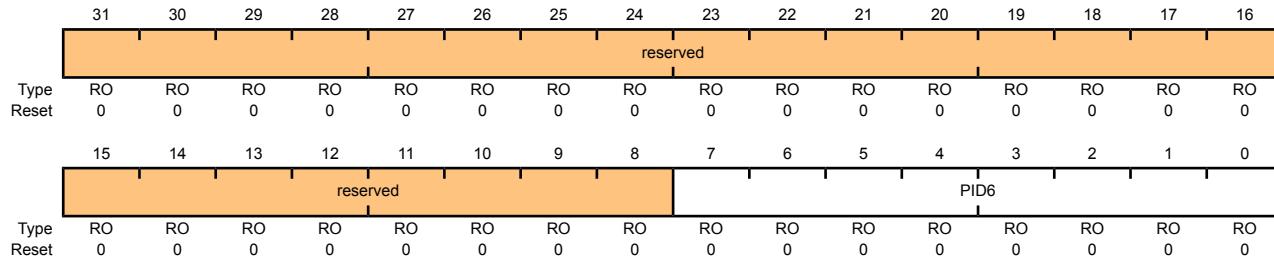
### UART Peripheral Identification 6 (UARTPeriphID6)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

Offset 0xFD8

Type RO, reset 0x0000.0000



## Register 17: UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

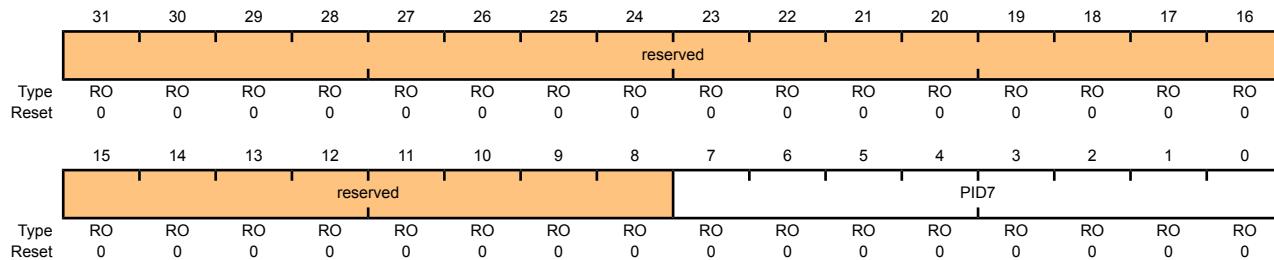
### UART Peripheral Identification 7 (UARTPeriphID7)

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 18: UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

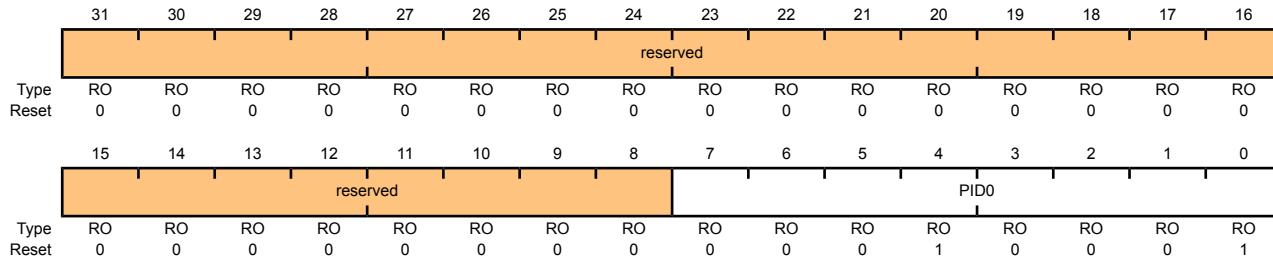
### UART Peripheral Identification 0 (UARTPeriphID0)

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 19: UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

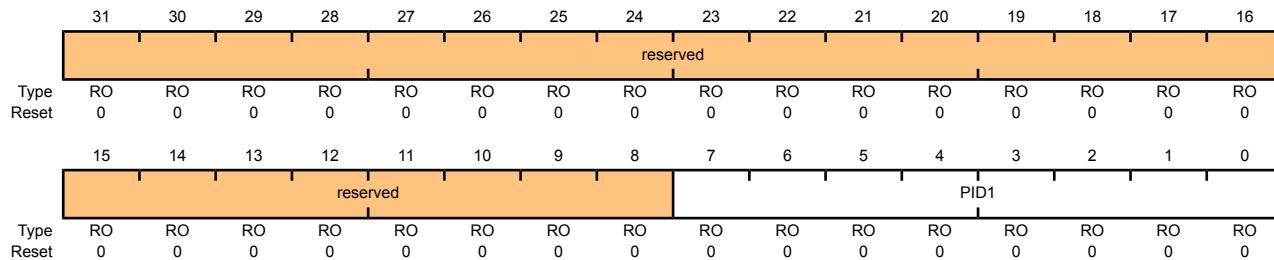
### UART Peripheral Identification 1 (UARTPeriphID1)

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 20: UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

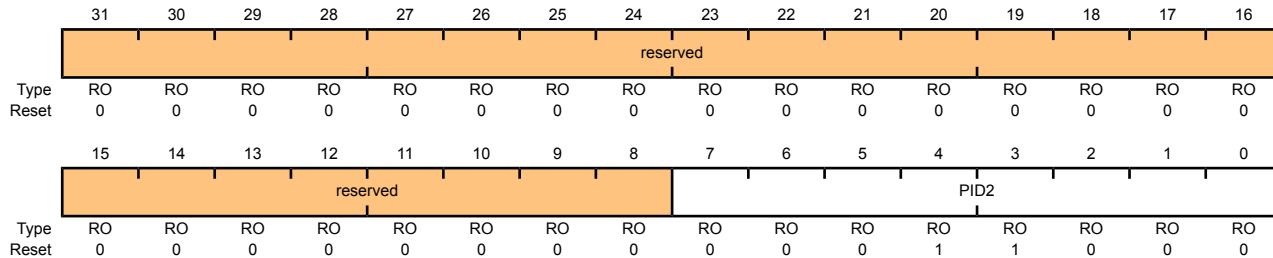
### UART Peripheral Identification 2 (UARTPeriphID2)

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 21: UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

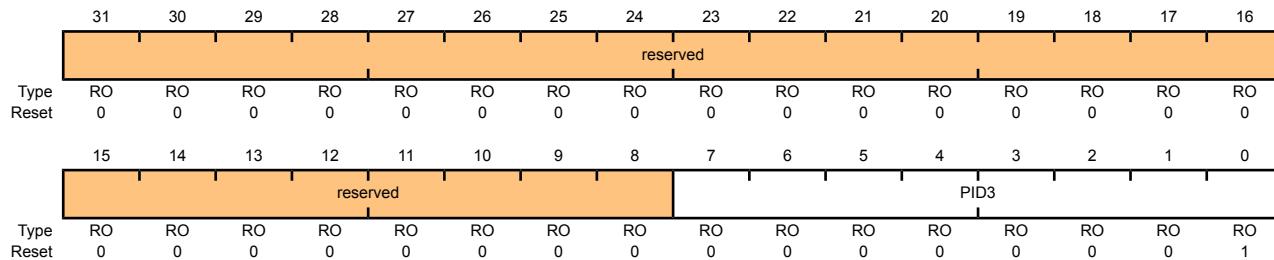
### UART Peripheral Identification 3 (UARTPeriphID3)

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 22: UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

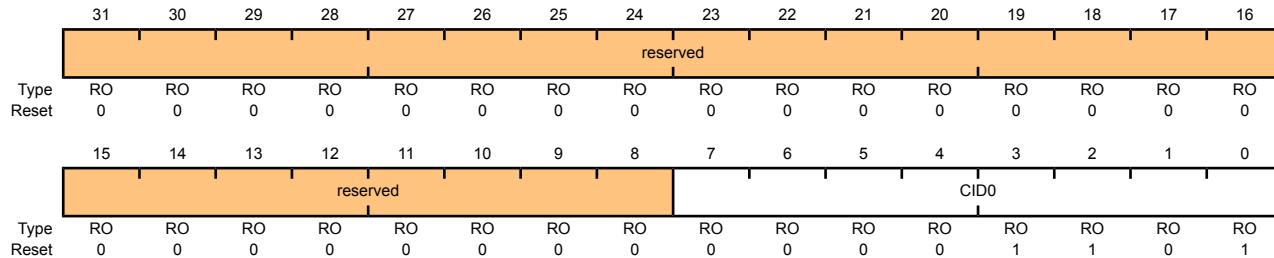
### UART PrimeCell Identification 0 (UARTPCellID0)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

Offset 0xFF0

Type RO, reset 0x0000.000D



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 23: UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

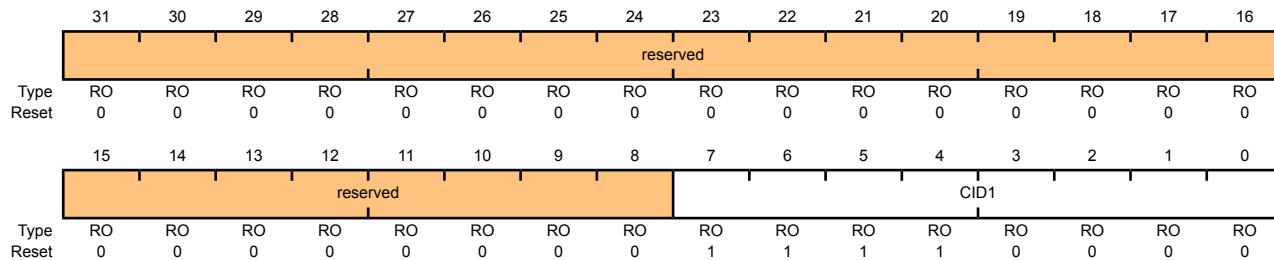
### UART PrimeCell Identification 1 (UARTPCellID1)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

Offset 0xFF4

Type RO, reset 0x0000.00F0



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 24: UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

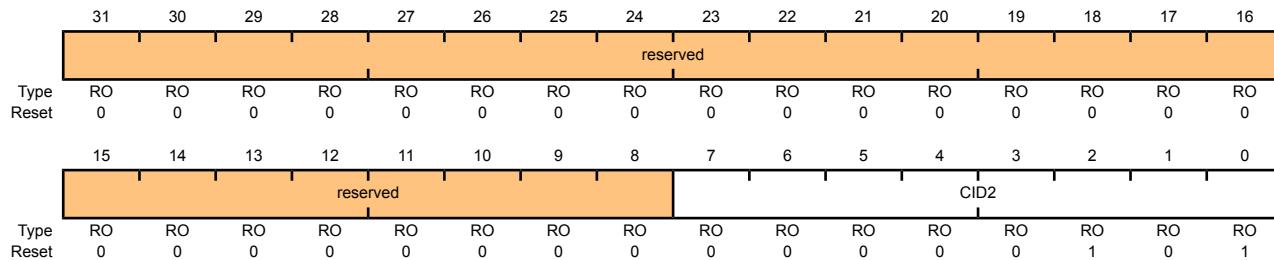
### UART PrimeCell Identification 2 (UARTPCellID2)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

Offset 0xFF8

Type RO, reset 0x0000.0005



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 25: UART PrimeCell Identification 3 (UARTPCellID3), offset 0xFFC

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

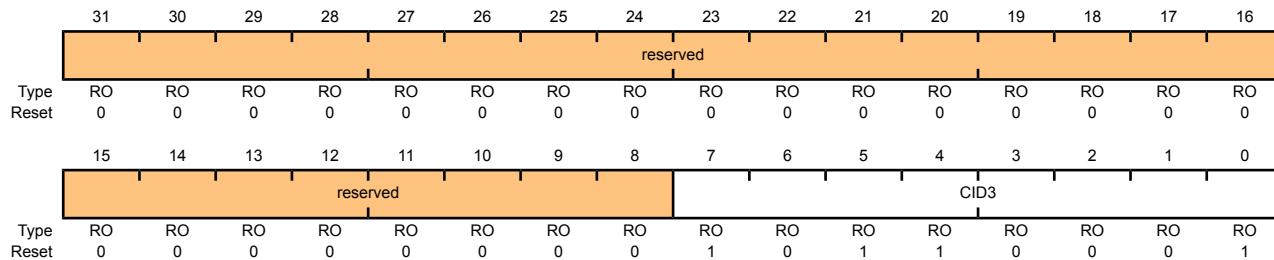
### UART PrimeCell Identification 3 (UARTPCellID3)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

Offset 0xFFC

Type RO, reset 0x0000.00B1



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.

## 14 Synchronous Serial Interface (SSI)

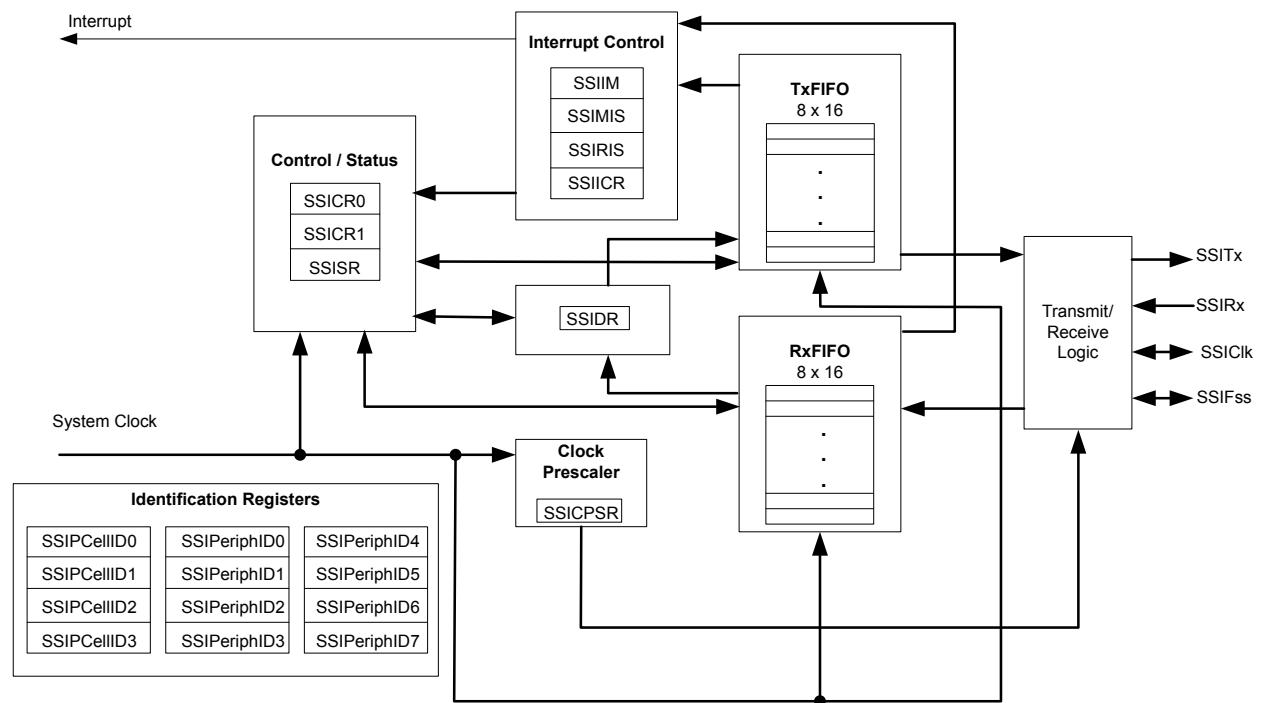
The Stellaris® Synchronous Serial Interface (SSI) is a master or slave interface for synchronous serial communication with peripheral devices that have either Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces.

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

### 14.1 Block Diagram

**Figure 14-1. SSI Module Block Diagram**



### 14.2 Functional Description

The SSI performs serial-to-parallel conversion on data received from a peripheral device. The CPU accesses data, control, and status information. The transmit and receive paths are buffered with

internal FIFO memories allowing up to eight 16-bit values to be stored independently in both transmit and receive modes.

### 14.2.1 Bit Rate Generation

The SSI includes a programmable bit rate clock divider and prescaler to generate the serial output clock. Bit rates are supported to 2 MHz and higher, although maximum bit rate is determined by peripheral devices.

The serial bit rate is derived by dividing down the 50-MHz input clock. 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 358). 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 351).

The frequency of the output clock **SSIClk** is defined by:

$$F_{SSIClk} = F_{SysClk} / (CPSDVSR * (1 + SCR))$$

Note that although the **SSIClk** transmit clock can theoretically be 25 MHz, the module may not be able to operate at that speed. For master mode, the system clock must be at least two times faster than the **SSIClk**. For slave mode, the system clock must be at least 12 times faster than the **SSIClk**.

See “Synchronous Serial Interface (SSI)” on page 586 to view SSI timing parameters.

### 14.2.2 FIFO Operation

#### 14.2.2.1 Transmit FIFO

The common transmit FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. The CPU writes data to the FIFO by writing the **SSI Data (SSIDR)** register (see page 355), 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.

#### 14.2.2.2 Receive FIFO

The common receive FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. Received data from the serial interface is stored in the buffer until read out by the CPU, which accesses the read FIFO by reading the **SSIDR** register.

When configured as a master or slave, serial data received through the **SSIRx** pin is registered prior to parallel loading into the attached slave or master receive FIFO, respectively.

### 14.2.3 Interrupts

The SSI can generate interrupts when the following conditions are observed:

- Transmit FIFO service
- Receive FIFO service
- Receive FIFO time-out
- Receive FIFO overrun

All of the interrupt events are ORed together before being sent to the interrupt controller, so the SSI can only generate a single interrupt request to the controller at any given time. You can mask each

of the four individual maskable interrupts by setting the appropriate bits in the **SSI Interrupt Mask (SSIIM)** register (see page 359). 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 361 and page 362, respectively).

#### 14.2.4 Frame Formats

Each data frame is between 4 and 16 bits long, depending on the size of data programmed, and is transmitted starting with the MSB. There are three basic frame types that can be selected:

- Texas Instruments synchronous serial
- Freescale SPI
- MICROWIRE

For all three formats, the serial clock (**SSIClk**) is held inactive while the SSI is idle, and **SSIClk** transitions at the programmed frequency only during active transmission or reception of data. The idle state of **SSIClk** is utilized to provide a receive timeout indication that occurs when the receive FIFO still contains data after a timeout period.

For Freescale SPI and MICROWIRE frame formats, the serial frame (**SSIFss**) pin is active Low, and is asserted (pulled down) during the entire transmission of the frame.

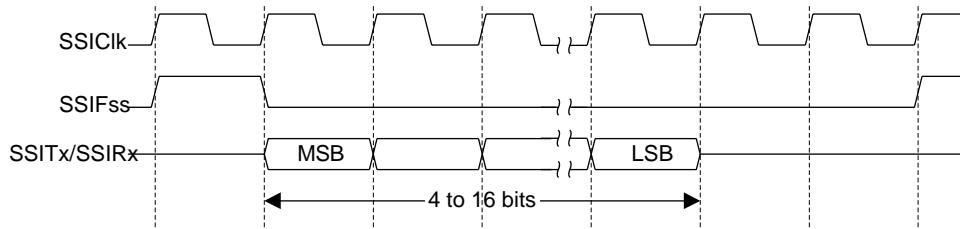
For Texas Instruments synchronous serial frame format, the **SSIFss** pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the SSI and the off-chip slave device drive their output data on the rising edge of **SSIClk**, and latch data from the other device on the falling edge.

Unlike the full-duplex transmission of the other two frame formats, the MICROWIRE format uses a special master-slave messaging technique, which operates at half-duplex. In this mode, when a frame begins, an 8-bit control message is transmitted to the off-chip slave. During this transmit, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the requested data. The returned data can be 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

##### 14.2.4.1 Texas Instruments Synchronous Serial Frame Format

Figure 14-2 on page 341 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

**Figure 14-2. TI Synchronous Serial Frame Format (Single Transfer)**

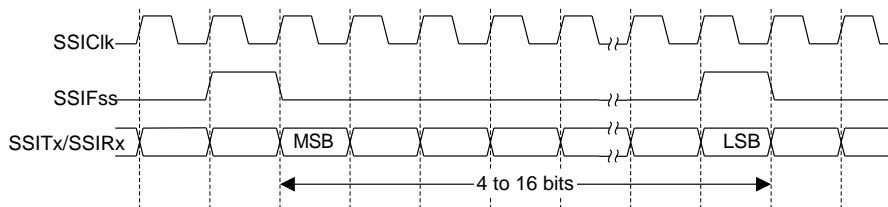


In this mode, SSIClk and SSIFss are forced Low, and the transmit data line SSITx is tristated whenever the SSI is idle. Once the bottom entry of the transmit FIFO contains data, SSIFss is pulsed High for one SSIClk period. The value to be transmitted is also transferred from the transmit FIFO to the 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 14-3 on page 342 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

**Figure 14-3. TI Synchronous Serial Frame Format (Continuous Transfer)**



#### 14.2.4.2 Freescale SPI Frame Format

The Freescale SPI interface is a four-wire interface where the SSIFss signal behaves as a slave select. The main feature of the Freescale SPI format is that the inactive state and phase of the SSIClk signal are programmable through the SPO and SPH bits within the **SSISCR0** control register.

##### SPO Clock Polarity Bit

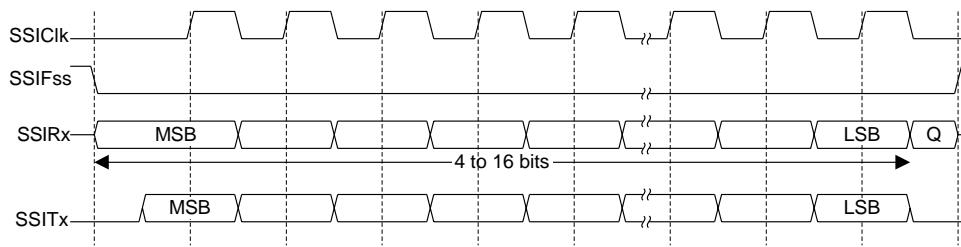
When the SPO clock polarity control bit is Low, it produces a steady state Low value on the SSIClk pin. If the SPO bit is High, a steady state High value is placed on the SSIClk pin when data is not being transferred.

##### SPH Phase Control Bit

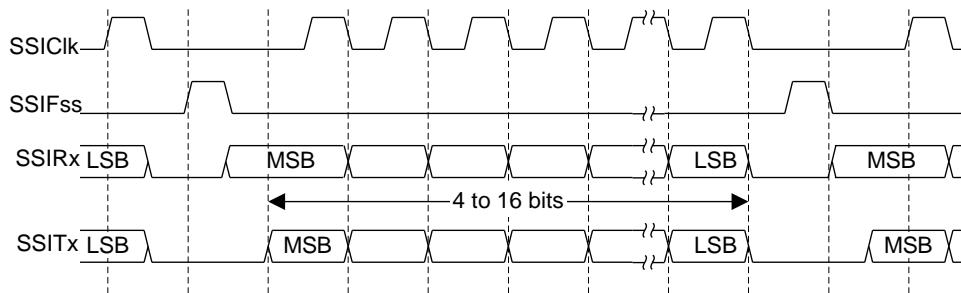
The SPH phase control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the SPH phase control bit is Low, data is captured on the first clock edge transition. If the SPH bit is High, data is captured on the second clock edge transition.

#### 14.2.4.3 Freescale SPI Frame Format with SPO=0 and SPH=0

Single and continuous transmission signal sequences for Freescale SPI format with SPO=0 and SPH=0 are shown in Figure 14-4 on page 343 and Figure 14-5 on page 343.

**Figure 14-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0**

**Note:** Q is undefined.

**Figure 14-5. 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.

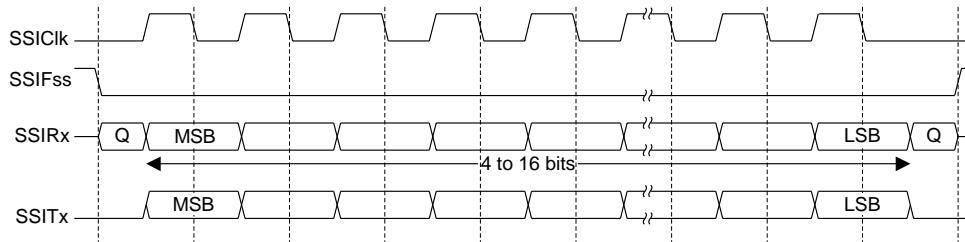
In the case of a single word transmission, after all bits of the data word have been transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

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.

#### 14.2.4.4 Freescale SPI Frame Format with SPO=0 and SPH=1

The transfer signal sequence for Freescale SPI format with SPO=0 and SPH=1 is shown in Figure 14-6 on page 344, which covers both single and continuous transfers.

**Figure 14-6. 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.

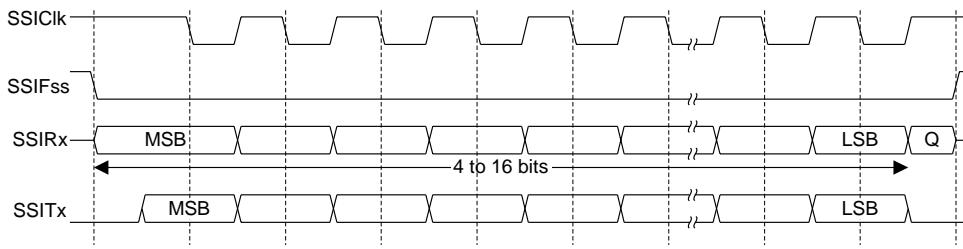
Data is then captured on the falling edges and propagated on the rising edges of the SSIClk signal.

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.

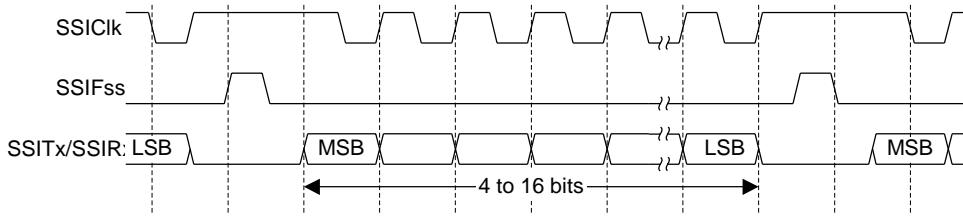
For continuous back-to-back transfers, the SSIFss pin is held Low between successive data words and termination is the same as that of the single word transfer.

#### 14.2.4.5 Freescale SPI Frame Format with SPO=1 and SPH=0

Single and continuous transmission signal sequences for Freescale SPI format with SPO=1 and SPH=0 are shown in Figure 14-7 on page 345 and Figure 14-8 on page 345.

**Figure 14-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0**

**Note:** Q is undefined.

**Figure 14-8. 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.

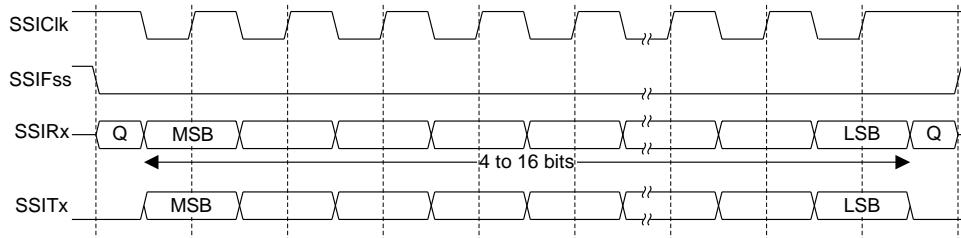
In the case of a single word transmission, after all bits of the data word are transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

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.

#### 14.2.4.6 Freescale SPI Frame Format with SPO=1 and SPH=1

The transfer signal sequence for Freescale SPI format with SPO=1 and SPH=1 is shown in Figure 14-9 on page 346, which covers both single and continuous transfers.

**Figure 14-9. 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.

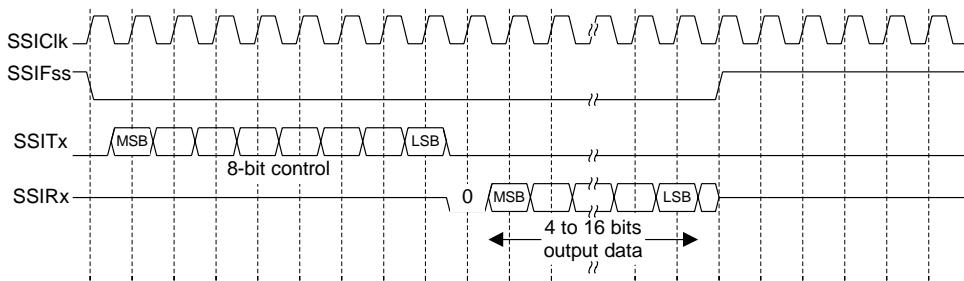
After all bits have been transferred, in the case of a single word transmission, the SSIFss line is returned to its idle high state one SSIClk period after the last bit has been captured.

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.

#### 14.2.4.7 MICROWIRE Frame Format

Figure 14-10 on page 347 shows the MICROWIRE frame format, again for a single frame. Figure 14-11 on page 348 shows the same format when back-to-back frames are transmitted.

**Figure 14-10. 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:

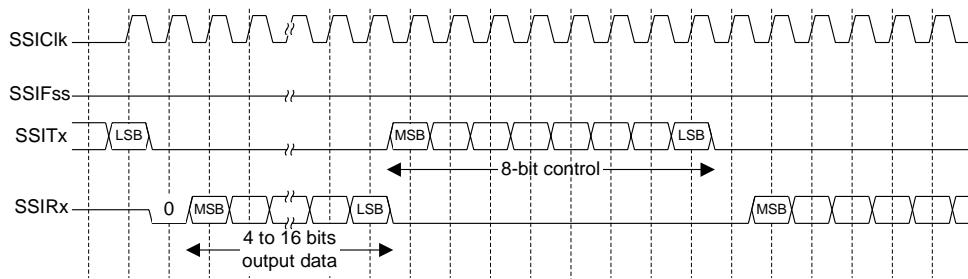
- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low

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.

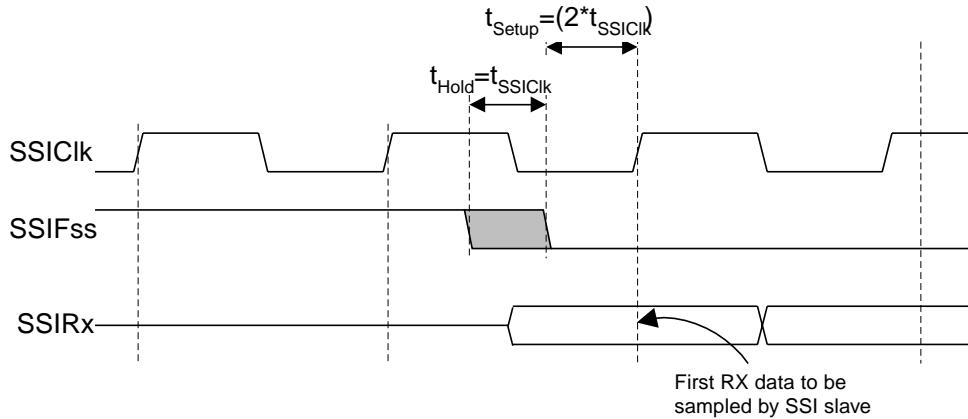
**Note:** The off-chip slave device can tristate the receive line either on the falling edge of SSIClk after the LSB has been latched by the receive shifter, or when the SSIFss pin goes High.

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 14-11. 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 14-12 on page 348 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 14-12. MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements**

## 14.3 Initialization and Configuration

To use the SSI, its peripheral clock must be enabled by setting the **SSI** bit in the **RCGC1** register.

For each of the frame formats, the SSI is configured using the following steps:

1. Ensure that the **SSE** bit in the **SSICR1** register is disabled before making any configuration changes.
2. Select whether the SSI is a master or slave:
  - a. For master operations, set the **SSICR1** register to 0x0000.0000.
  - b. For slave mode (output enabled), set the **SSICR1** register to 0x0000.0004.
  - c. For slave mode (output disabled), set the **SSICR1** register to 0x0000.000C.
3. Configure the clock prescale divisor by writing the **SSICPSR** register.

4. Write the **SSICR0** register with the following configuration:
  - Serial clock rate (SCR)
  - Desired clock phase/polarity, if using Freescale SPI mode (SPH and SPO)
  - The protocol mode: Freescale SPI, TI SSF, MICROWIRE (FRF)
  - The data size (DSS)
5. 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:

$$\begin{aligned} FSSI_{Clk} &= FSysClk / (CPSDVSR * (1 + SCR)) \\ 1 \times 10^6 &= 20 \times 10^6 / (CPSDVSR * (1 + SCR)) \end{aligned}$$

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.

## 14.4 Register Map

Table 14-1 on page 349 lists the SSI registers. The offset listed is a hexadecimal increment to the register's address, relative to that SSI module's base address:

- SSI0: 0x4000.8000

**Note:** The SSI must be disabled (see the SSE bit in the **SSICR1** register) before any of the control registers are reprogrammed.

**Table 14-1. SSI Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	SSICR0	R/W	0x0000.0000	SSI Control 0	351

Offset	Name	Type	Reset	Description	See page
0x004	SSICR1	R/W	0x0000.0000	SSI Control 1	353
0x008	SSIDR	R/W	0x0000.0000	SSI Data	355
0x00C	SSISR	RO	0x0000.0003	SSI Status	356
0x010	SSICPSR	R/W	0x0000.0000	SSI Clock Prescale	358
0x014	SSIIM	R/W	0x0000.0000	SSI Interrupt Mask	359
0x018	SSIRIS	RO	0x0000.0008	SSI Raw Interrupt Status	361
0x01C	SSIMIS	RO	0x0000.0000	SSI Masked Interrupt Status	362
0x020	SSIICR	W1C	0x0000.0000	SSI Interrupt Clear	363
0xFD0	SSIPeriphID4	RO	0x0000.0000	SSI Peripheral Identification 4	364
0xFD4	SSIPeriphID5	RO	0x0000.0000	SSI Peripheral Identification 5	365
0xFD8	SSIPeriphID6	RO	0x0000.0000	SSI Peripheral Identification 6	366
0xFDC	SSIPeriphID7	RO	0x0000.0000	SSI Peripheral Identification 7	367
0xFE0	SSIPeriphID0	RO	0x0000.0022	SSI Peripheral Identification 0	368
0xFE4	SSIPeriphID1	RO	0x0000.0000	SSI Peripheral Identification 1	369
0xFE8	SSIPeriphID2	RO	0x0000.0018	SSI Peripheral Identification 2	370
0xFEC	SSIPeriphID3	RO	0x0000.0001	SSI Peripheral Identification 3	371
0xFF0	SSIPCellID0	RO	0x0000.000D	SSI PrimeCell Identification 0	372
0xFF4	SSIPCellID1	RO	0x0000.00F0	SSI PrimeCell Identification 1	373
0xFF8	SSIPCellID2	RO	0x0000.0005	SSI PrimeCell Identification 2	374
0xFFC	SSIPCellID3	RO	0x0000.00B1	SSI PrimeCell Identification 3	375

## 14.5 Register Descriptions

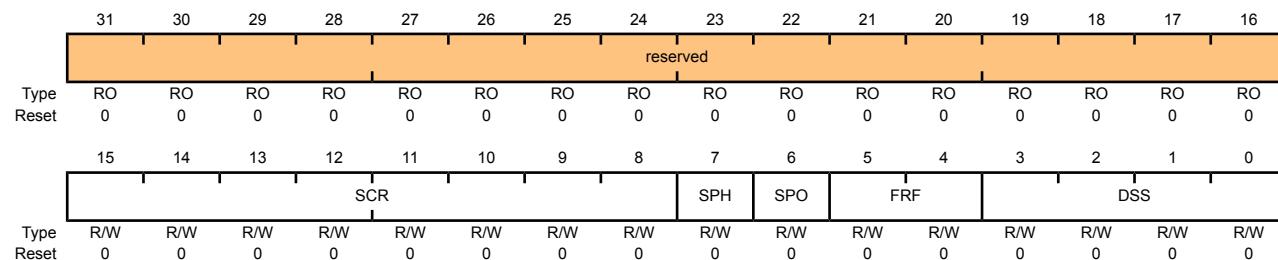
The remainder of this section lists and describes the SSI registers, in numerical order by address offset.

## Register 1: SSI Control 0 (SSICR0), offset 0x000

**SSICR0** is control register 0 and contains bit fields that control various functions within the SSI module. Functionality such as protocol mode, clock rate, and data size are configured in this register.

### SSI Control 0 (SSICR0)

SSI0 base: 0x4000.8000  
Offset 0x000  
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	SCR	R/W	0x0000	SSI Serial Clock Rate  The value SCR is used to generate the transmit and receive bit rate of the SSI. The bit rate is:  $BR = F_{SSIClk} / (CPSDVSR * (1 + SCR))$  where CPSDVSR is an even value from 2-254 programmed in the <b>SSICPSR</b> register, and SCR is a value from 0-255.
7	SPH	R/W	0	SSI Serial Clock Phase  This bit is only applicable to the Freescale SPI Format.  The 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.  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.
6	SPO	R/W	0	SSI Serial Clock Polarity  This bit is only applicable to the Freescale SPI Format.  When the SPO bit is 0, it produces a steady state Low value on the SSIClk pin. If SPO is 1, a steady state High value is placed on the SSIClk pin when data is not being transferred.

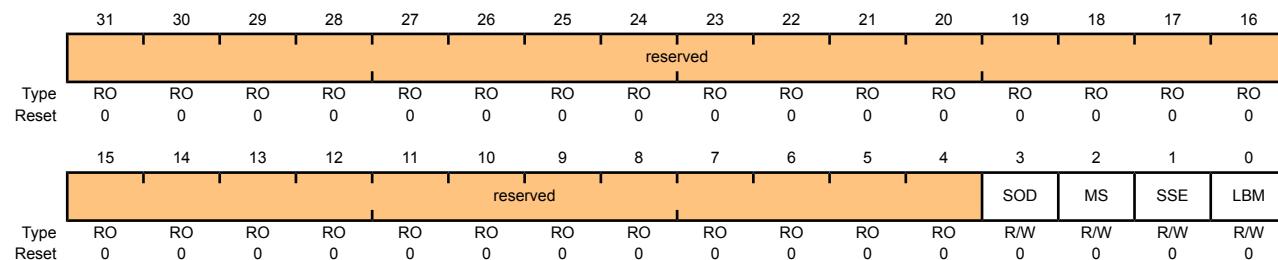
Bit/Field	Name	Type	Reset	Description																														
5:4	FRF	R/W	0x0	<p>SSI Frame Format Select</p> <p>The FRF values are defined as follows:</p> <table><thead><tr><th>Value</th><th>Frame Format</th></tr></thead><tbody><tr><td>0x0</td><td>Freescale SPI Frame Format</td></tr><tr><td>0x1</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></tbody></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																																	
3:0	DSS	R/W	0x00	<p>SSI Data Size Select</p> <p>The DSS values are defined as follows:</p> <table><thead><tr><th>Value</th><th>Data Size</th></tr></thead><tbody><tr><td>0x0-0x2</td><td>Reserved</td></tr><tr><td>0x3</td><td>4-bit data</td></tr><tr><td>0x4</td><td>5-bit data</td></tr><tr><td>0x5</td><td>6-bit data</td></tr><tr><td>0x6</td><td>7-bit data</td></tr><tr><td>0x7</td><td>8-bit data</td></tr><tr><td>0x8</td><td>9-bit data</td></tr><tr><td>0x9</td><td>10-bit data</td></tr><tr><td>0xA</td><td>11-bit data</td></tr><tr><td>0xB</td><td>12-bit data</td></tr><tr><td>0xC</td><td>13-bit data</td></tr><tr><td>0xD</td><td>14-bit data</td></tr><tr><td>0xE</td><td>15-bit data</td></tr><tr><td>0xF</td><td>16-bit data</td></tr></tbody></table>	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
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.</p> <p>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.</p> <p>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

**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.

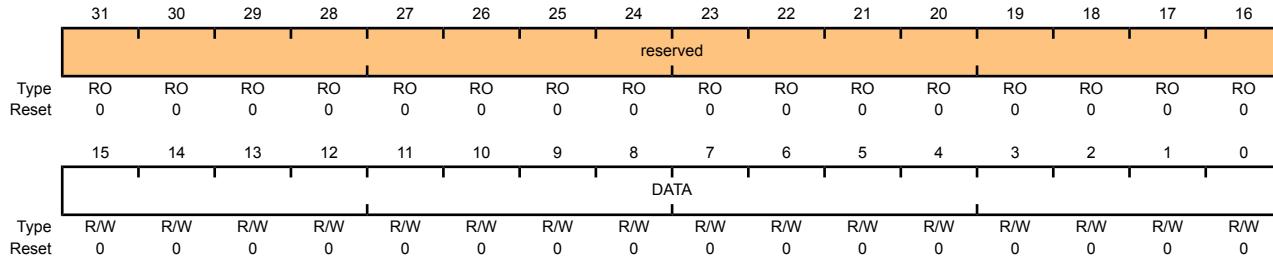
When the SSI is programmed for MICROWIRE frame format, the default size for transmit data is eight bits (the most significant byte is ignored). The receive data size is controlled by the programmer. The transmit FIFO and the receive FIFO are not cleared even when the **SSE** bit in the **SSICR1** register 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

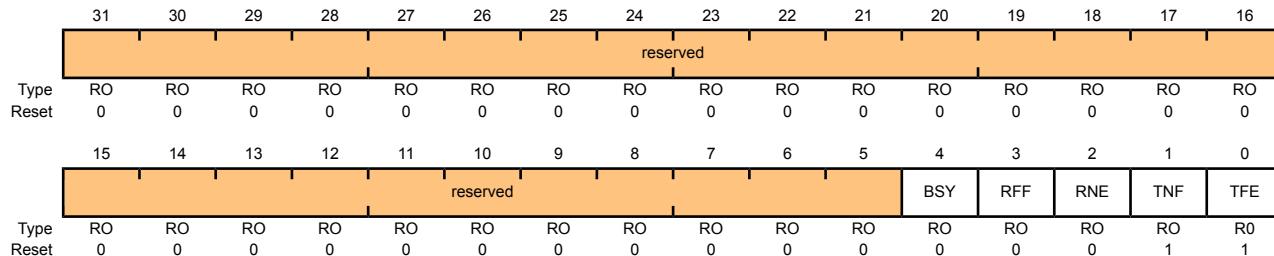
**SSISR** is a status register that contains bits that indicate the FIFO fill status and the SSI busy status.

### SSI Status (SSISR)

SSI0 base: 0x4000.8000

Offset 0x00C

Type RO, reset 0x0000.0003



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 **BSY** 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 **RFF** 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 **RNE** 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 **TNF** 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.

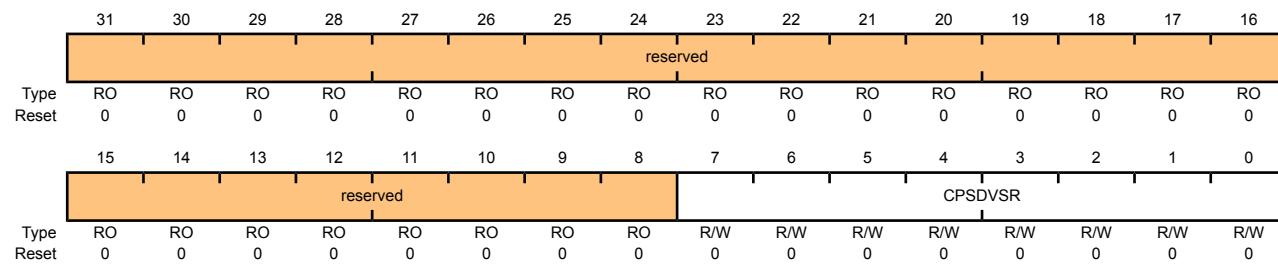
The value programmed into this register must be an even number between 2 and 254. The least-significant bit of the programmed number is hard-coded to zero. If an odd number is written to this register, data read back from this register has the least-significant bit as zero.

### 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.

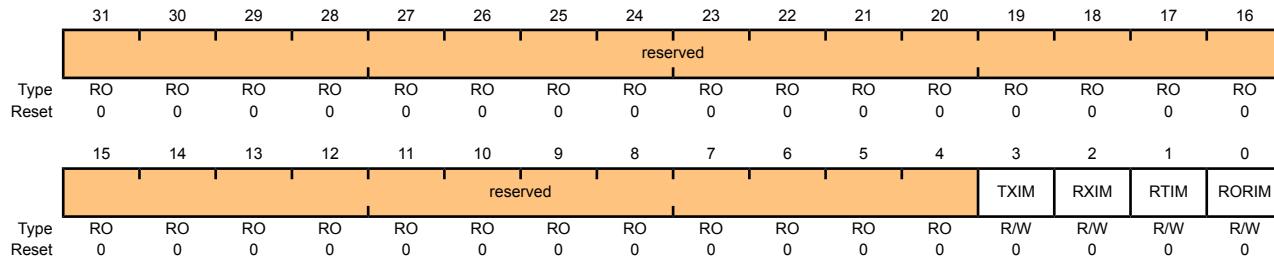
On a read, this register gives the current value of the mask on the relevant interrupt. A write of 1 to the particular bit sets the mask, enabling the interrupt to be read. A write of 0 clears the corresponding mask.

### SSI Interrupt Mask (SSIIM)

SSI0 base: 0x4000.8000

Offset 0x014

Type R/W, reset 0x0000.0000



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-full or less condition interrupt is masked. 1 TX FIFO half-full or less condition interrupt is not masked.
2	RXIM	R/W	0	SSI Receive FIFO Interrupt Mask The 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

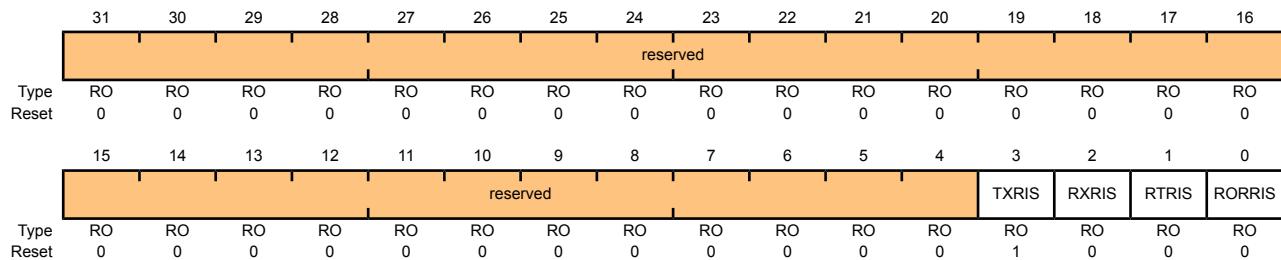
The **SSIRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect.

### 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 full or less, when set.
2	RXRIS	RO	0	SSI Receive FIFO Raw Interrupt Status Indicates that the receive FIFO is half full or more, when set.
1	RTRIS	RO	0	SSI Receive Time-Out Raw Interrupt Status Indicates that the receive time-out has occurred, when set.
0	RORRIS	RO	0	SSI Receive Overrun Raw Interrupt Status Indicates that the receive FIFO has overflowed, when set.

## Register 8: SSI Masked Interrupt Status (SSIMIS), offset 0x01C

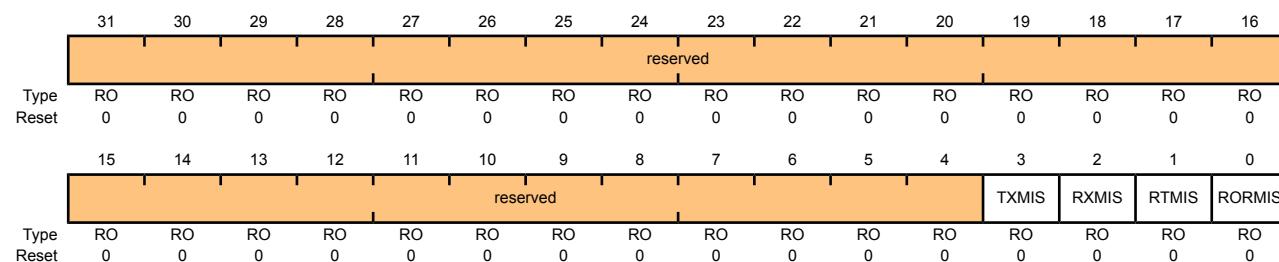
The **SSIMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

### SSI Masked Interrupt Status (SSIMIS)

SSI0 base: 0x4000.8000

Offset 0x01C

Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXMIS	RO	0	SSI Transmit FIFO Masked Interrupt Status Indicates that the transmit FIFO is half full or less, when set.
2	RXMIS	RO	0	SSI Receive FIFO Masked Interrupt Status Indicates that the receive FIFO is half full or more, when set.
1	RTMIS	RO	0	SSI Receive Time-Out Masked Interrupt Status Indicates that the receive time-out has occurred, when set.
0	RORMIS	RO	0	SSI Receive Overrun Masked Interrupt Status Indicates that the receive FIFO has overflowed, when set.

## Register 9: SSI Interrupt Clear (SSIICR), offset 0x020

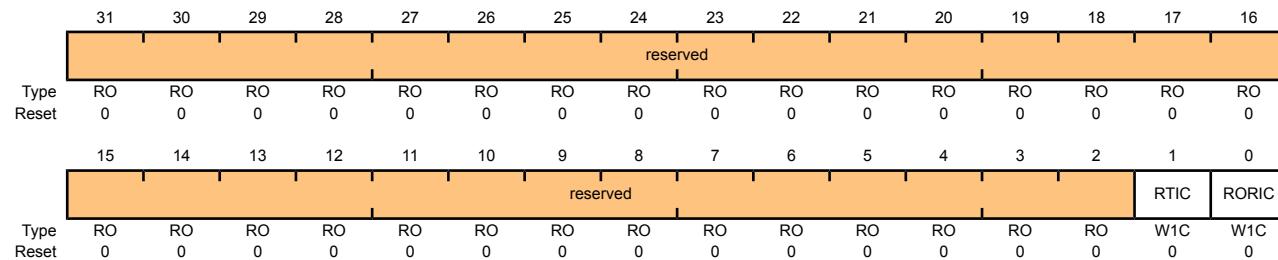
The **SSIICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

### SSI Interrupt Clear (SSIICR)

SSI0 base: 0x4000.8000

Offset 0x020

Type W1C, reset 0x0000.0000



#### 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

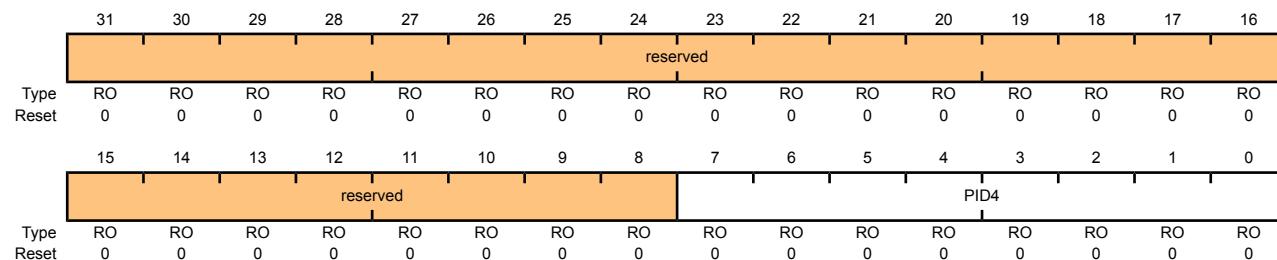
The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### 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

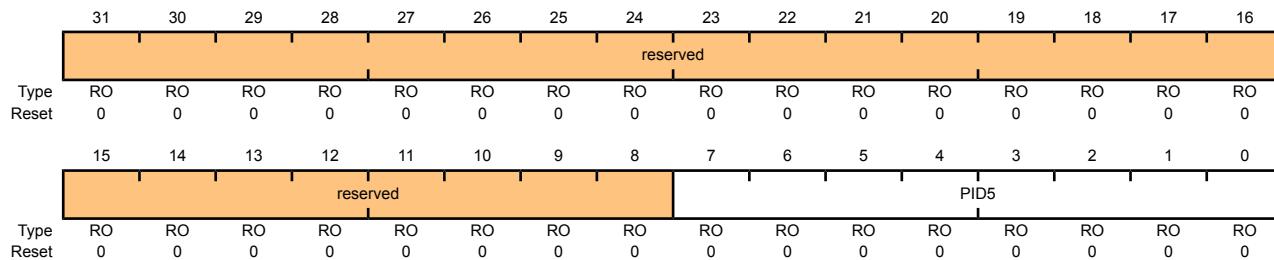
The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### 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

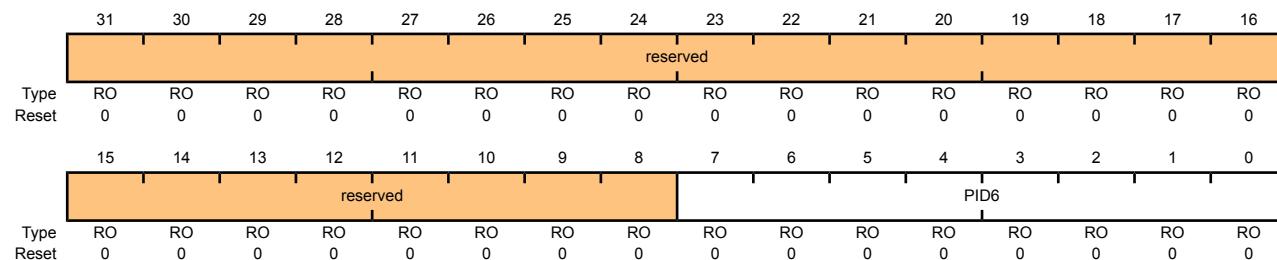
The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### 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

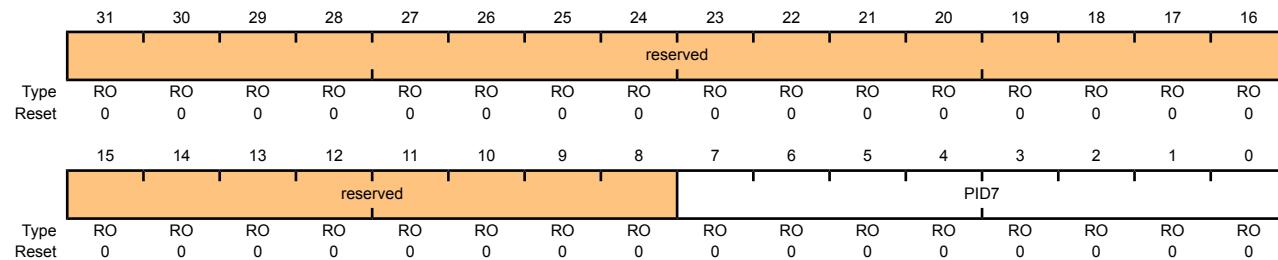
The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### 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

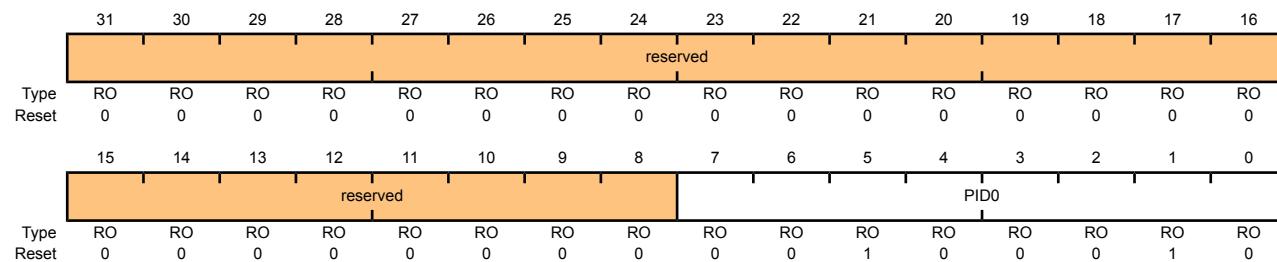
The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### 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

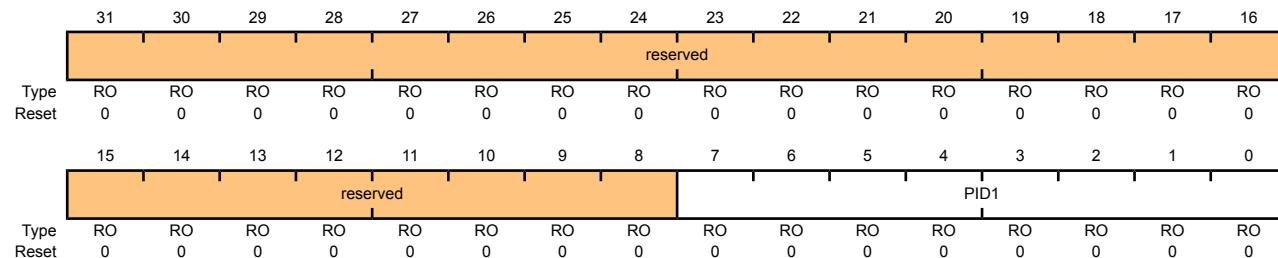
The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### 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

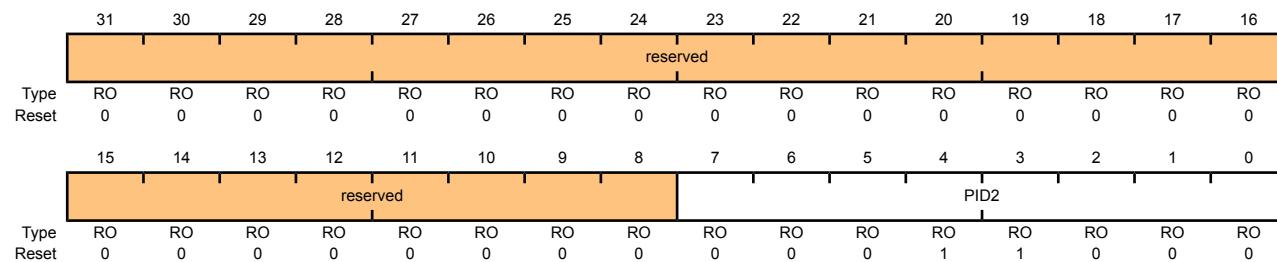
The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### 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

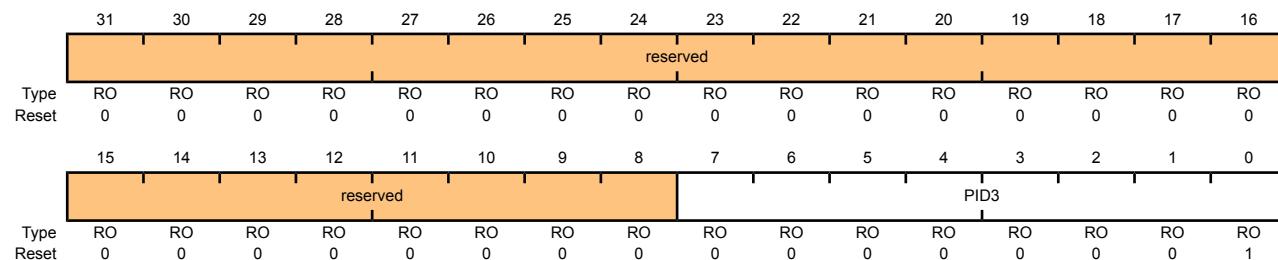
The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

### SSI Peripheral Identification 3 (SSIPeriphID3)

SSI0 base: 0x4000.8000

Offset 0xFEC

Type RO, reset 0x0000.0001



## Register 18: SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0

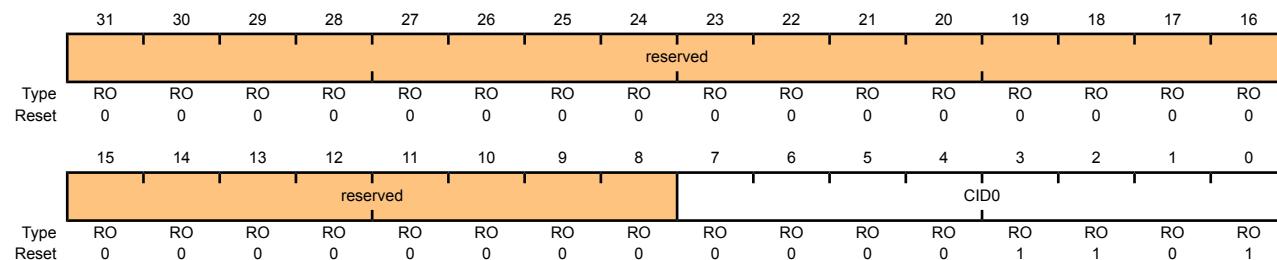
The **SSIPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

### 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

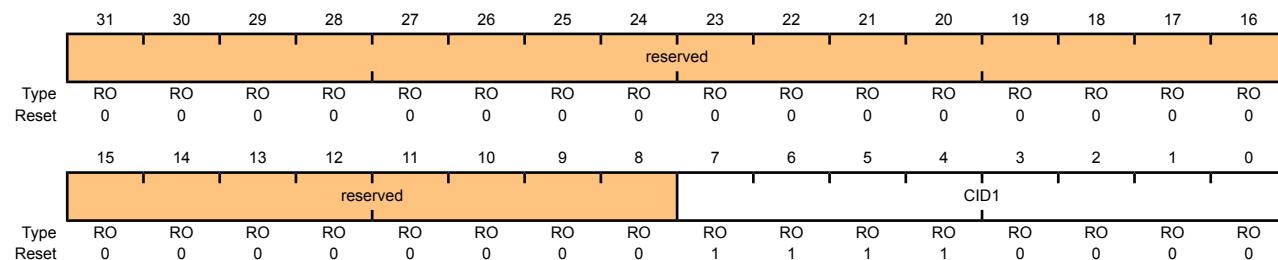
The **SSIPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

### 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

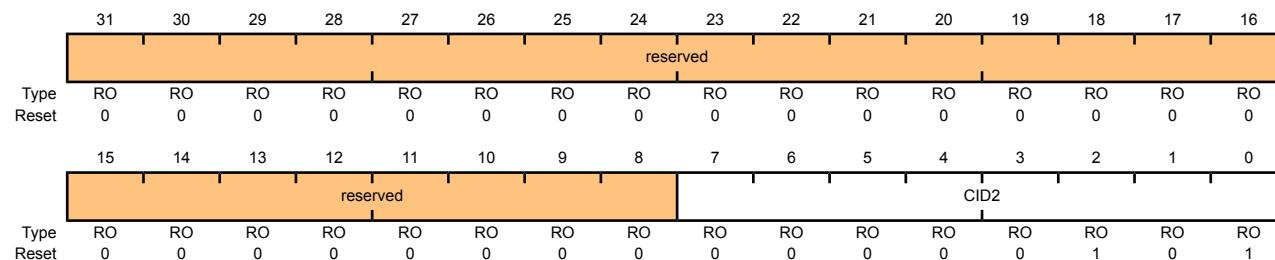
The **SSIPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

### 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

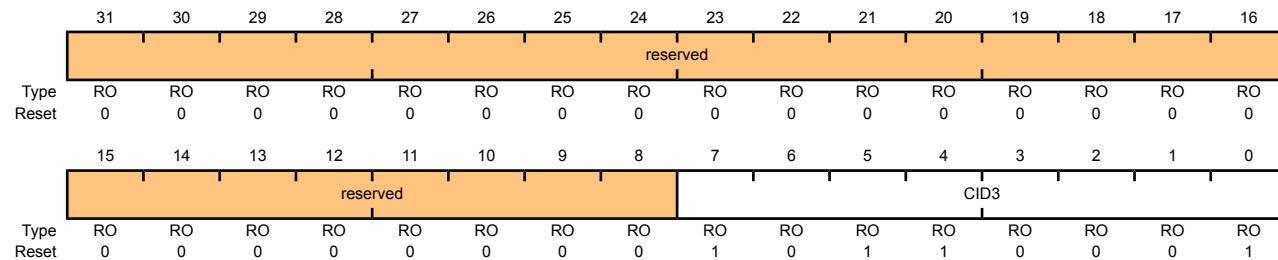
The **SSIPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

### 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.

## 15 Inter-Integrated Circuit (I<sup>2</sup>C) Interface

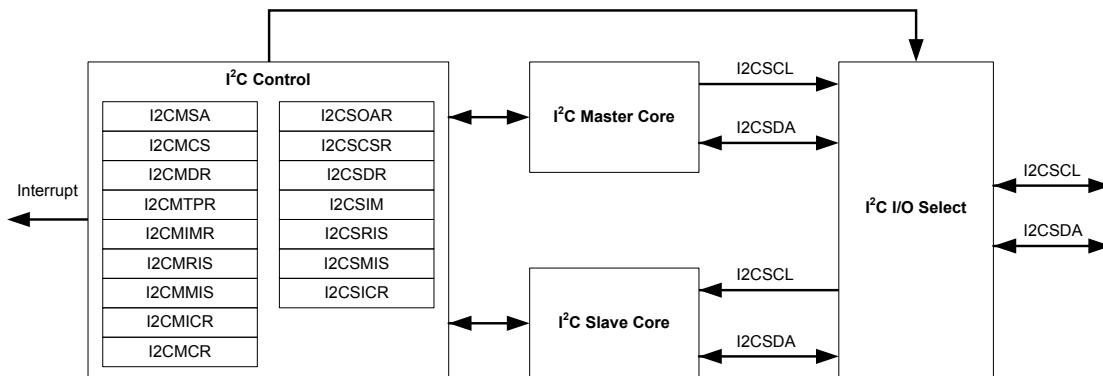
The Inter-Integrated Circuit (I<sup>2</sup>C) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL), and interfaces to external I<sup>2</sup>C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and 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 LM3S8962 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.

Devices on the I<sup>2</sup>C bus can be designated as either a master or a slave. The Stellaris® I<sup>2</sup>C module supports both sending and receiving data as either a master or a slave, and also supports the simultaneous operation as both a master and a slave. There are a total of four I<sup>2</sup>C modes: Master Transmit, Master Receive, Slave Transmit, and Slave Receive. The Stellaris® 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) and the I<sup>2</sup>C slave generates interrupts when data has been sent or requested by a master.

### 15.1 Block Diagram

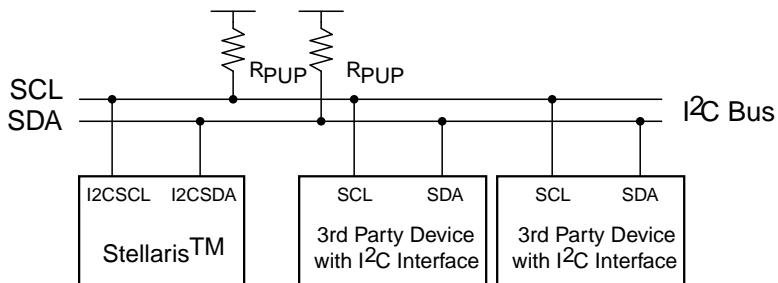
Figure 15-1. I<sup>2</sup>C Block Diagram



### 15.2 Functional Description

The 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 15-2 on page 377.

See "I<sup>2</sup>C" on page 582 for I<sup>2</sup>C timing diagrams.

**Figure 15-2. I<sup>2</sup>C Bus Configuration**

### 15.2.1 I<sup>2</sup>C Bus Functional Overview

The I<sup>2</sup>C bus uses only two signals: SDA and SCL, named I<sup>2</sup>CSDA and I<sup>2</sup>CSCL on Stellaris<sup>®</sup> microcontrollers. SDA is the bi-directional serial data line and SCL is the bi-directional serial clock line. The bus is considered idle when both lines are high.

Every transaction on the I<sup>2</sup>C bus is nine bits long, consisting of eight data bits and a single acknowledge bit. The number of bytes per transfer (defined as the time between a valid START and STOP condition, described in “START and STOP Conditions” on page 377) 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.

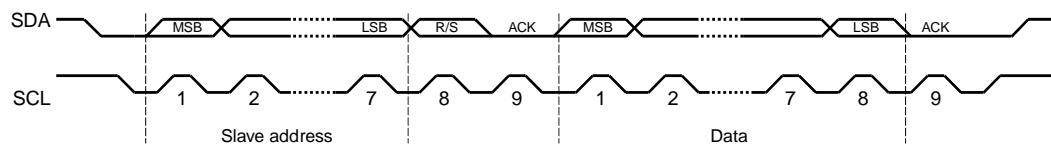
#### 15.2.1.1 START and STOP Conditions

The protocol of the I<sup>2</sup>C bus defines two states to begin and end a transaction: START and STOP. A high-to-low transition on the SDA line while the SCL is high is defined as a START condition, and a low-to-high transition on the SDA line while SCL is high is defined as a STOP condition. The bus is considered busy after a START condition and free after a STOP condition. See Figure 15-3 on page 377.

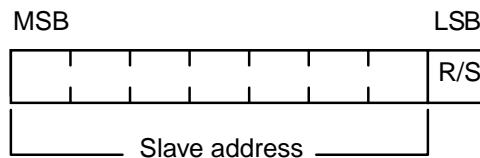
**Figure 15-3. START and STOP Conditions**

#### 15.2.1.2 Data Format with 7-Bit Address

Data transfers follow the format shown in Figure 15-4 on page 378. 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.

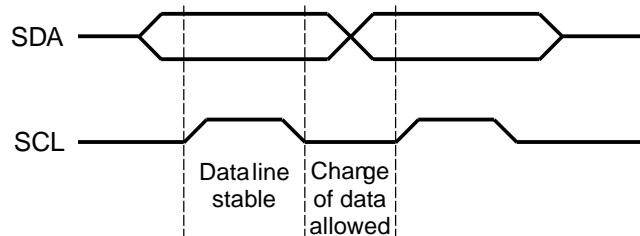
**Figure 15-4. Complete Data Transfer with a 7-Bit Address**

The first seven bits of the first byte make up the slave address (see Figure 15-5 on page 378). 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 15-5. R/S Bit in First Byte**

### 15.2.1.3 Data Validity

The data on the SDA line must be stable during the high period of the clock, and the data line can only change when SCL is low (see Figure 15-6 on page 378).

**Figure 15-6. Data Validity During Bit Transfer on the  $I^2C$  Bus**

### 15.2.1.4 Acknowledge

All bus transactions have a required acknowledge clock cycle that is generated by the master. During the acknowledge cycle, the transmitter (which can be the master or slave) releases the SDA line. To acknowledge the transaction, the receiver must pull down SDA during the acknowledge clock cycle. The data sent out by the receiver during the acknowledge cycle must comply with the data validity requirements described in "Data Validity" on page 378.

When a slave receiver does not acknowledge the slave address, SDA must be left high by the slave so that the master can generate a STOP condition and abort the current transfer. If the master device is acting as a receiver during a transfer, it is responsible for acknowledging each transfer made by the slave. Since the master controls the number of bytes in the transfer, it signals the end of data to the slave transmitter by not generating an acknowledge on the last data byte. The slave transmitter must then release SDA to allow the master to generate the STOP or a repeated START condition.

### 15.2.1.5 Arbitration

A master may start a transfer only if the bus is idle. It's possible for two or more masters to generate a START condition within minimum hold time of the START condition. In these situations, an arbitration scheme takes place on the SDA line, while SCL is high. During arbitration, the first of the competing master devices to place a '1' (high) on SDA while another master transmits a '0' (low) will switch off its data output stage and retire until the bus is idle again.

Arbitration can take place over several bits. Its first stage is a comparison of address bits, and if both masters are trying to address the same device, arbitration continues on to the comparison of data bits.

### 15.2.2 Available Speed Modes

The I<sup>2</sup>C clock rate is determined by the parameters: CLK\_PRD, TIMER\_PRD, SCL\_LP, and SCL\_HP. where:

CLK\_PRD is the system clock period

SCL\_LP is the low phase of SCL (fixed at 6)

SCL\_HP is the high phase of SCL (fixed at 4)

TIMER\_PRD is the programmed value in the **I<sup>2</sup>C Master Timer Period (I2CMTPR)** register (see page 396).

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:

$$1/T = 333 \text{ KHz}$$

Table 15-1 on page 379 gives examples of timer period, system clock, and speed mode (Standard or Fast).

**Table 15-1. Examples of I<sup>2</sup>C Master Timer Period versus Speed Mode**

System Clock	Timer Period	Standard Mode	Timer Period	Fast Mode
4 Mhz	0x01	100 Kbps	-	-
6 Mhz	0x02	100 Kbps	-	-
12.5 Mhz	0x06	89 Kbps	0x01	312 Kbps
16.7 Mhz	0x08	93 Kbps	0x02	278 Kbps
20 Mhz	0x09	100 Kbps	0x02	333 Kbps
25 Mhz	0x0C	96.2 Kbps	0x03	312 Kbps
33Mhz	0x10	97.1 Kbps	0x04	330 Kbps
40Mhz	0x13	100 Kbps	0x04	400 Kbps

System Clock	Timer Period	Standard Mode	Timer Period	Fast Mode
50Mhz	0x18	100 Kbps	0x06	357 Kbps

### 15.2.3 Interrupts

The I<sup>2</sup>C can generate interrupts when the following conditions are observed:

- Master transaction completed
- Master transaction error
- Slave transaction received
- Slave transaction requested

There is a separate interrupt signal for the I<sup>2</sup>C master and I<sup>2</sup>C modules. While both modules can generate interrupts for multiple conditions, only a single interrupt signal is sent to the interrupt controller.

#### 15.2.3.1 I<sup>2</sup>C Master Interrupts

The I<sup>2</sup>C master module generates an interrupt when a transaction completes (either transmit or receive), or when an error occurs during a transaction. To enable the I<sup>2</sup>C master interrupt, software must write a '1' to the **I<sup>2</sup>C Master Interrupt Mask (I2CMIMR)** register. When an interrupt condition is met, software must check the **ERROR** bit in the **I<sup>2</sup>C Master Control/Status (I2CMCS)** register to verify that an error didn't occur during the last transaction. An error condition is asserted if the last transaction wasn't acknowledged by the slave or if the master was forced to give up ownership of the bus due to a lost arbitration round with another master. If an error is not detected, the application can proceed with the transfer. The interrupt is cleared by writing a '1' to the **I<sup>2</sup>C Master Interrupt Clear (I2CMICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I<sup>2</sup>C Master Raw Interrupt Status (I2CMRIS)** register.

#### 15.2.3.2 I<sup>2</sup>C Slave Interrupts

The slave module generates interrupts as it receives requests from an I<sup>2</sup>C master. To enable the I<sup>2</sup>C slave interrupt, write a '1' to the **I<sup>2</sup>C Slave Interrupt Mask (I2CSIMR)** register. Software determines whether the module should write (transmit) or read (receive) data from the **I<sup>2</sup>C Slave Data (I2CSDR)** register, by checking the **RREQ** and **TREQ** bits of the **I<sup>2</sup>C Slave Control/Status (I2CSCSR)** register. If the slave module is in receive mode and the first byte of a transfer is received, the **FBR** bit is set along with the **RREQ** bit. The interrupt is cleared by writing a '1' to the **I<sup>2</sup>C Slave Interrupt Clear (I2CSICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I<sup>2</sup>C Slave Raw Interrupt Status (I2CSRIS)** register.

#### 15.2.4 Loopback Operation

The I<sup>2</sup>C modules can be placed into an internal loopback mode for diagnostic or debug work. This is accomplished by setting the **LPBK** bit in the **I<sup>2</sup>C Master Configuration (I2CMCR)** register. In loopback mode, the SDA and SCL signals from the master and slave modules are tied together.

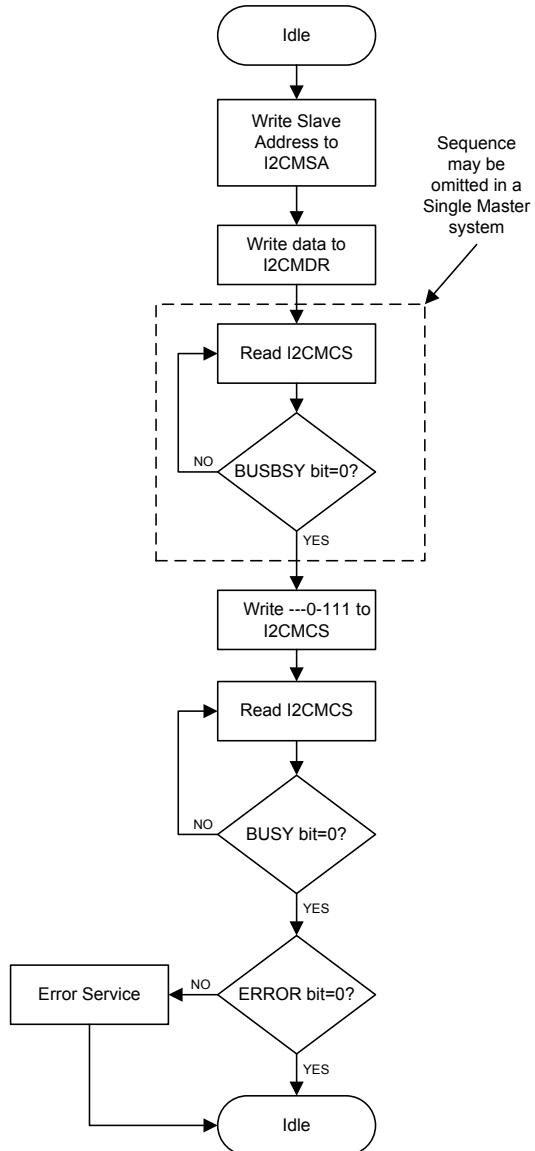
## 15.2.5 Command Sequence Flow Charts

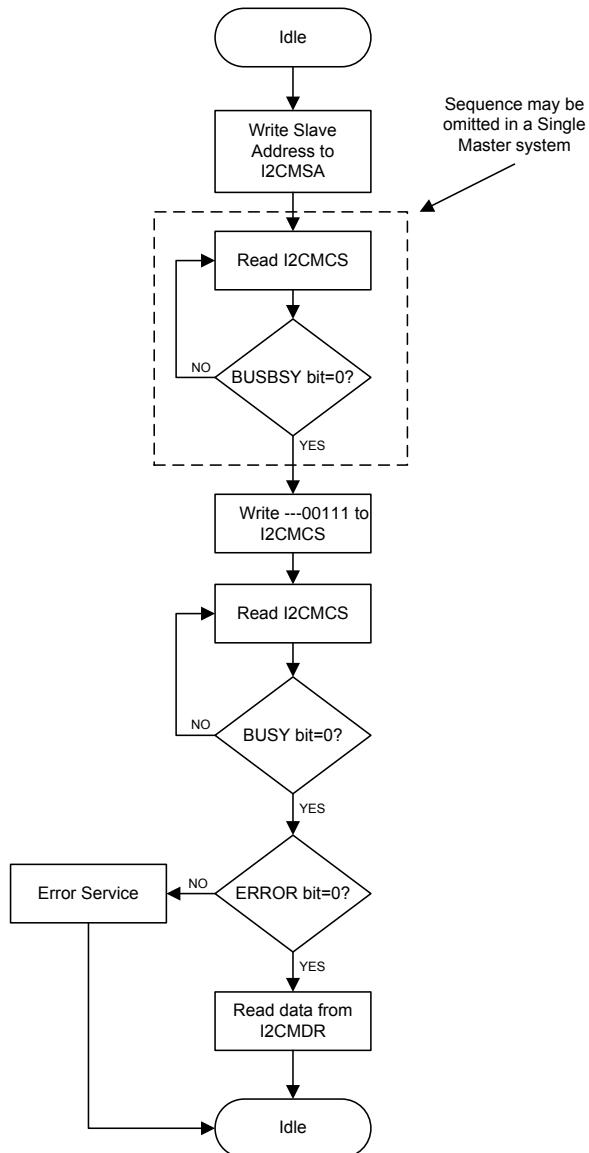
This section details the steps required to perform the various I<sup>2</sup>C transfer types in both master and slave mode.

### 15.2.5.1 I<sup>2</sup>C Master Command Sequences

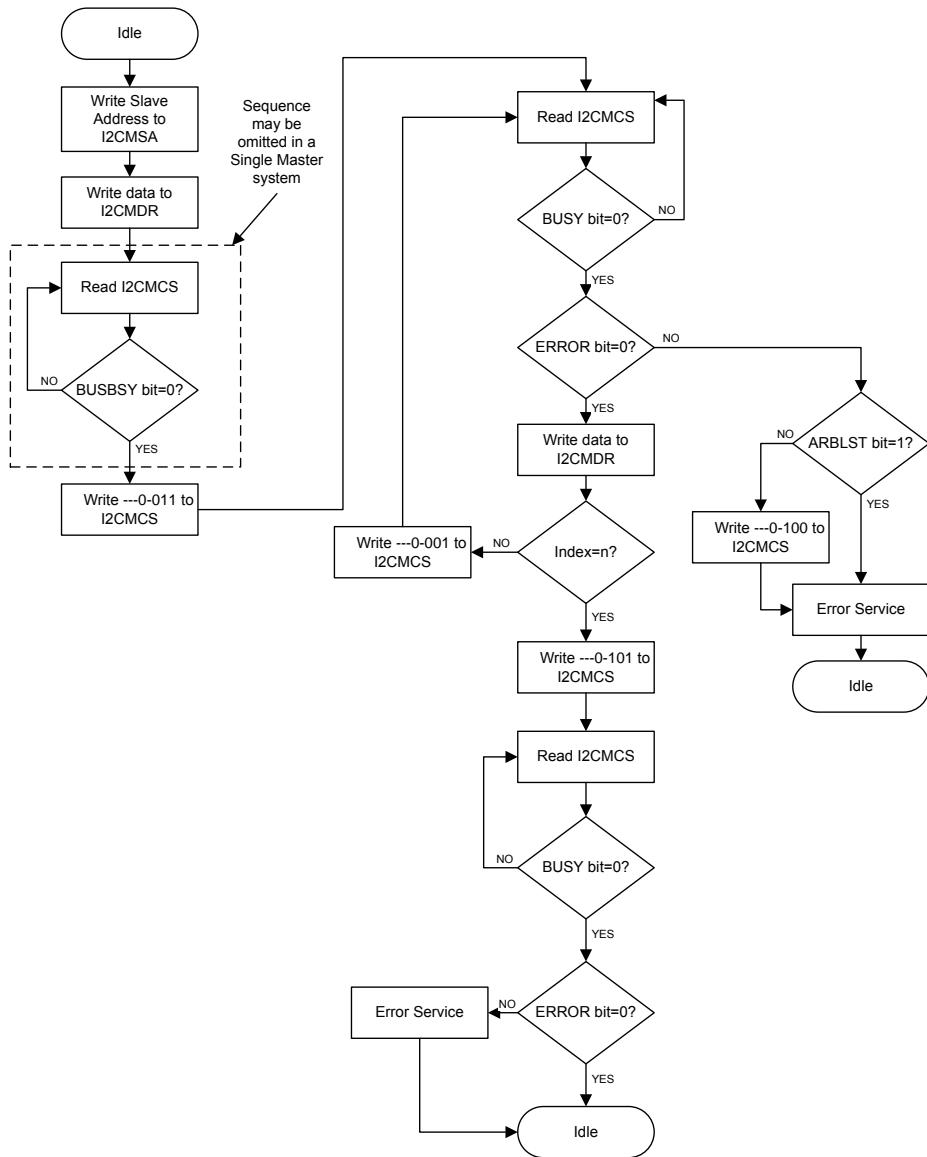
The figures that follow show the command sequences available for the I<sup>2</sup>C master.

**Figure 15-7. Master Single SEND**

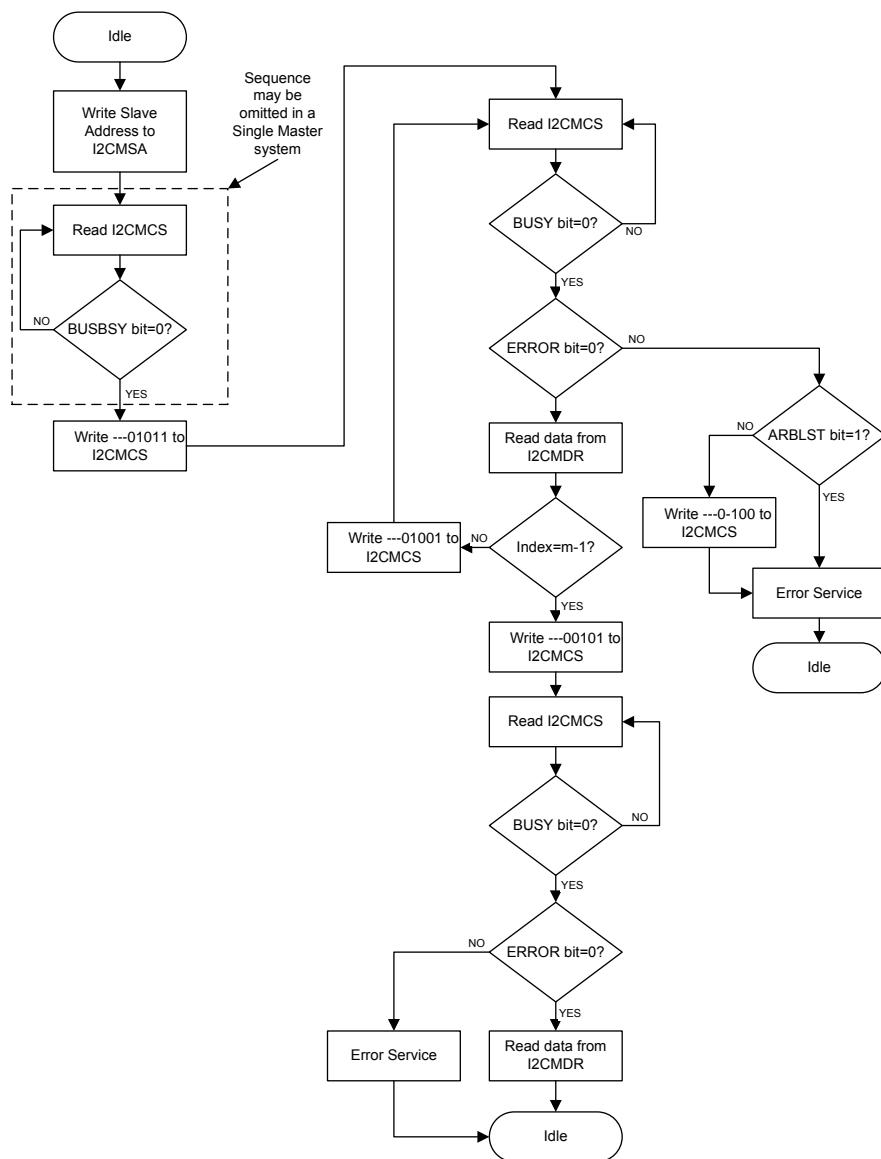


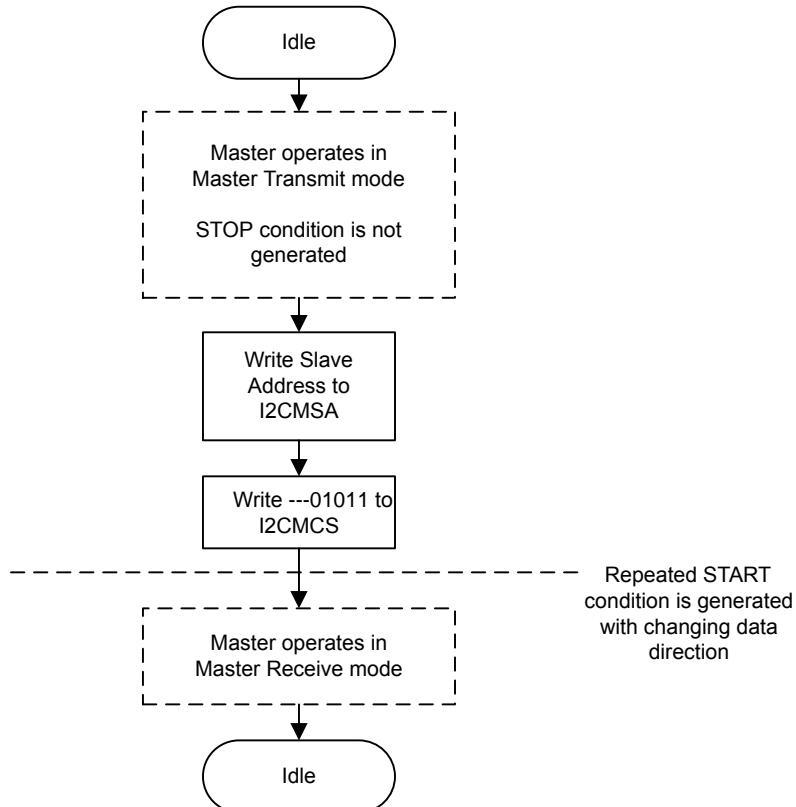
**Figure 15-8. Master Single RECEIVE**

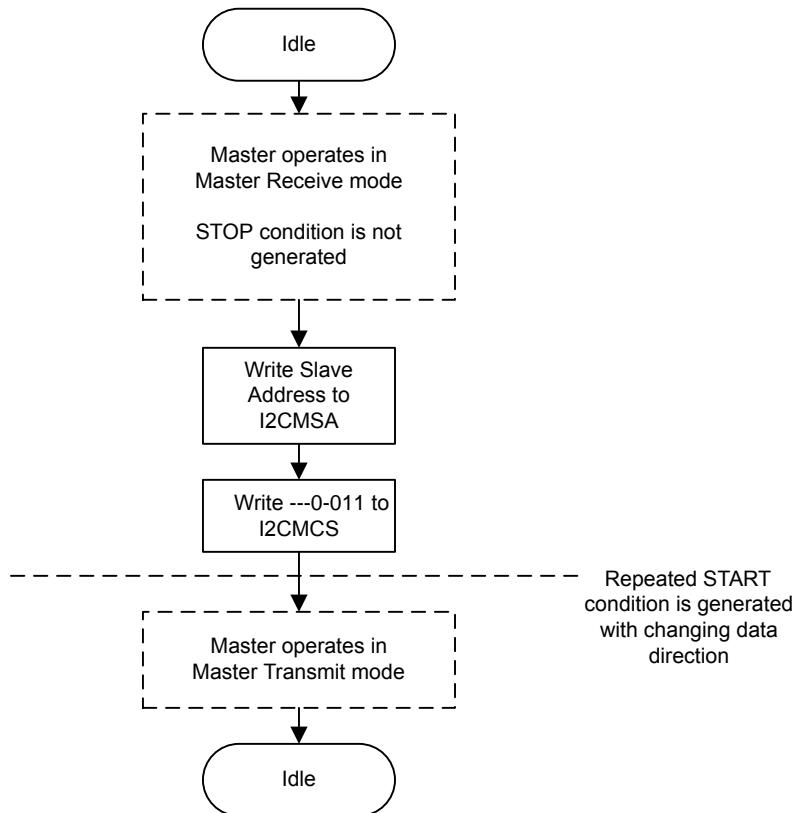
**Figure 15-9. Master Burst SEND**



**Figure 15-10. Master Burst RECEIVE**

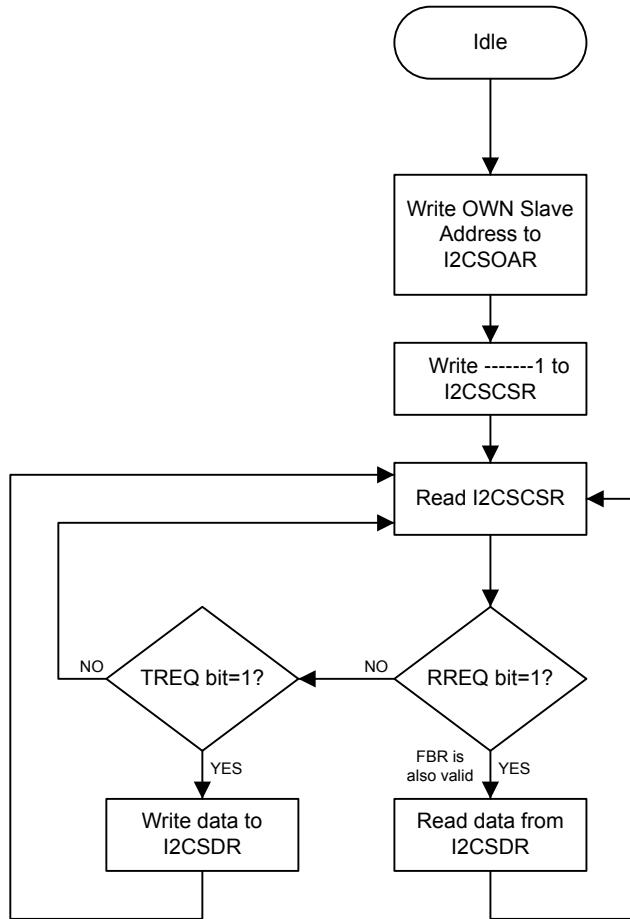


**Figure 15-11. Master Burst RECEIVE after Burst SEND**

**Figure 15-12. Master Burst SEND after Burst RECEIVE**

### 15.2.5.2 $I^2C$ Slave Command Sequences

Figure 15-13 on page 387 presents the command sequence available for the  $I^2C$  slave.

**Figure 15-13. Slave Command Sequence**

### 15.3 Initialization and Configuration

The following example shows how to configure the I<sup>2</sup>C module to send a single byte as a master. This assumes the system clock is 20 MHz.

1. Enable the I<sup>2</sup>C clock by writing a value of 0x0000.1000 to the **RCGC1** register in the System Control module.
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module.
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register. Also, be sure to enable the same pins for Open Drain operation.
4. Initialize the I<sup>2</sup>C Master by writing the **I2CMCR** register with a value of 0x0000.0020.
5. Set the desired SCL clock speed of 100 Kbps by writing the **I2CMTPR** register with the correct value. The value written to the **I2CMTPR** register represents the number of system clock periods in one SCL clock period. The TPR value is determined by the following equation:

```

TPR = (System Clock / (2 * (SCL_LP + SCL_HP) * SCL_CLK)) - 1;
TPR = (20MHz / (2 * (6 + 4) * 100000)) - 1;
TPR = 9

```

Write the **I2CMTPR** register with the value of 0x0000.0009.

6. Specify the slave address of the master and that the next operation will be a Send by writing the **I2CMSA** register with a value of 0x0000.0076. This sets the slave address to 0x3B.
7. Place data (byte) to be sent in the data register by writing the **I2CMDR** register with the desired data.
8. Initiate a single byte send of the data from Master to Slave by writing the **I2CMCS** register with a value of 0x0000.0007 (STOP, START, RUN).
9. Wait until the transmission completes by polling the **I2CMCS** register's **BUSBSY** bit until it has been cleared.

## 15.4 I<sup>2</sup>C Register Map

Table 15-2 on page 388 lists the I<sup>2</sup>C registers. All addresses given are relative to the I<sup>2</sup>C base addresses for the master and slave:

- I<sup>2</sup>C Master 0: 0x4002.0000
- I<sup>2</sup>C Slave 0: 0x4002.0800

**Table 15-2. Inter-Integrated Circuit (I<sup>2</sup>C) Interface Register Map**

Offset	Name	Type	Reset	Description	See page
<b>I<sup>2</sup>C Master</b>					
0x000	I2CMSA	R/W	0x0000.0000	I2C Master Slave Address	390
0x004	I2CMCS	R/W	0x0000.0000	I2C Master Control/Status	391
0x008	I2CMDR	R/W	0x0000.0000	I2C Master Data	395
0x00C	I2CMTPR	R/W	0x0000.0001	I2C Master Timer Period	396
0x010	I2CMIMR	R/W	0x0000.0000	I2C Master Interrupt Mask	397
0x014	I2CMRIS	RO	0x0000.0000	I2C Master Raw Interrupt Status	398
0x018	I2CMMIS	RO	0x0000.0000	I2C Master Masked Interrupt Status	399
0x01C	I2CMICR	WO	0x0000.0000	I2C Master Interrupt Clear	400
0x020	I2CMCR	R/W	0x0000.0000	I2C Master Configuration	401
<b>I<sup>2</sup>C Slave</b>					
0x000	I2CSOAR	R/W	0x0000.0000	I2C Slave Own Address	403
0x004	I2CSCSR	RO	0x0000.0000	I2C Slave Control/Status	404
0x008	I2CSDR	R/W	0x0000.0000	I2C Slave Data	406
0x00C	I2CSIMR	R/W	0x0000.0000	I2C Slave Interrupt Mask	407

Offset	Name	Type	Reset	Description	See page
0x010	I2CSRIS	RO	0x0000.0000	I2C Slave Raw Interrupt Status	408
0x014	I2CSMIS	RO	0x0000.0000	I2C Slave Masked Interrupt Status	409
0x018	I2CSICR	WO	0x0000.0000	I2C Slave Interrupt Clear	410

## 15.5 Register Descriptions (I<sup>2</sup>C Master)

The remainder of this section lists and describes the I<sup>2</sup>C master registers, in numerical order by address offset. See also “Register Descriptions (I<sup>2</sup>C Slave)” on page 402.

## Register 1: I<sup>2</sup>C Master Slave Address (I2CMSA), offset 0x000

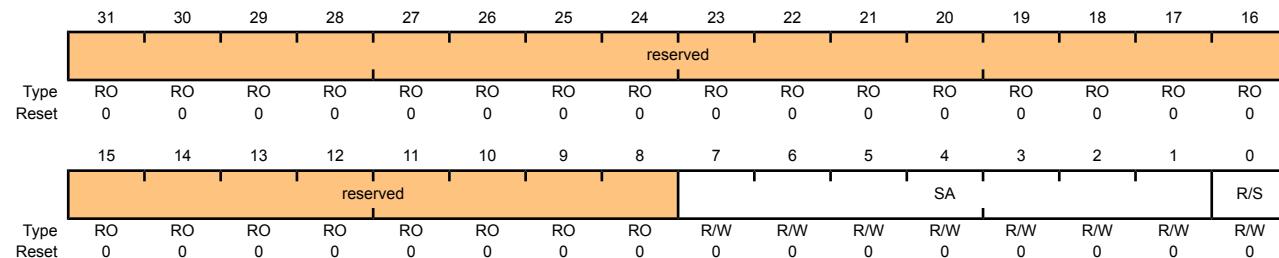
This register consists of eight bits: seven address bits (A6-A0), and a Receive/Send bit, which determines if the next operation is a Receive (High), or Send (Low).

### I2C Master Slave Address (I2CMSA)

I2C Master 0 base: 0x4002.0000

Offset 0x000

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:1	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.

### Read-Only Status Register

#### I<sup>2</sup>C Master Control/Status (I2CMCS)

I<sup>2</sup>C Master 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
Type	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	BUSBSY	IDLE	ARBLST	DATACK	ADRACK	ERROR	BUSY								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	BUSBSY	RO	0	Bus Busy This bit specifies the state of the I <sup>2</sup> C bus. If set, the bus is busy; otherwise, the bus is idle. The bit changes based on the START and STOP conditions.
5	IDLE	RO	0	I <sup>2</sup> C Idle This bit specifies the I <sup>2</sup> C controller state. If set, the controller is idle; otherwise the controller is not idle.
4	ARBLST	RO	0	Arbitration Lost This bit specifies the result of bus arbitration. If set, the controller lost arbitration; otherwise, the controller won arbitration.

Bit/Field	Name	Type	Reset	Description
3	DATAACK	RO	0	Acknowledge Data  This bit specifies the result of the last data operation. If set, the transmitted data was not acknowledged; otherwise, the data was acknowledged.
2	ADRACK	RO	0	Acknowledge Address  This bit specifies the result of the last address operation. If set, the transmitted address was not acknowledged; otherwise, the address was acknowledged.
1	ERROR	RO	0	Error  This bit specifies the result of the last bus operation. If set, an error occurred on the last operation; otherwise, no error was detected. The error can be from the slave address not being acknowledged, the transmit data not being acknowledged, or because the controller lost arbitration.
0	BUSY	RO	0	I <sup>2</sup> C Busy  This bit specifies the state of the controller. If set, the controller is busy; otherwise, the controller is idle. When the BUSY bit is set, the other status bits are not valid.

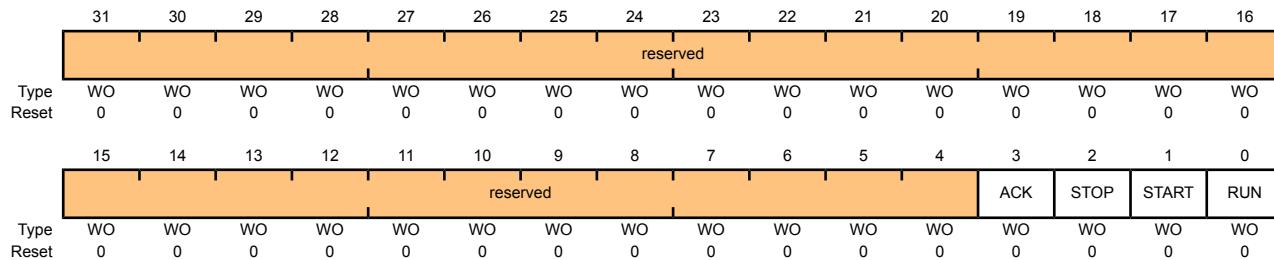
## Write-Only Control Register

### I<sup>2</sup>C Master Control/Status (I2CMCS)

I<sup>2</sup>C Master 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 15-3 on page 393.
2	STOP	WO	0	Generate STOP  When set, causes the generation of the STOP condition. See field decoding in Table 15-3 on page 393.

Bit/Field	Name	Type	Reset	Description
1	START	WO	0	Generate START When set, causes the generation of a START or repeated START condition. See field decoding in Table 15-3 on page 393.
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 15-3 on page 393.

**Table 15-3. Write Field Decoding for I2CMCS[3:0] Field (Sheet 1 of 3)**

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
		R/S	ACK	STOP	START	
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.

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
	R/S	ACK	STOP	START	RUN	
Master Receive	X	0	0	0	1	RECEIVE operation with negative ACK (master remains in Master Receive state).
	X	X	1	0	0	STOP condition (master goes to Idle state). <sup>b</sup>
	X	0	1	0	1	RECEIVE followed by STOP condition (master goes to Idle state).
	X	1	0	0	1	RECEIVE operation (master remains in Master Receive state).
	X	1	1	0	1	Illegal.
	1	0	0	1	1	Repeated START condition followed by RECEIVE operation with a negative ACK (master remains in Master Receive state).
	1	0	1	1	1	Repeated START condition followed by RECEIVE and STOP condition (master goes to Idle state).
	1	1	0	1	1	Repeated START condition followed by RECEIVE (master remains in Master Receive state).
	0	X	0	1	1	Repeated START condition followed by SEND (master goes to Master Transmit state).
	0	X	1	1	1	Repeated START condition followed by SEND and STOP condition (master goes to Idle state).
All other combinations not listed are non-operations.					NOP.	

a. An X in a table cell indicates the bit can be 0 or 1.

b. In Master Receive mode, a STOP condition should be generated only after a Data Negative Acknowledge executed by the master or an Address Negative Acknowledge executed by the slave.

## Register 3: I<sup>2</sup>C Master Data (I2CMDR), offset 0x008

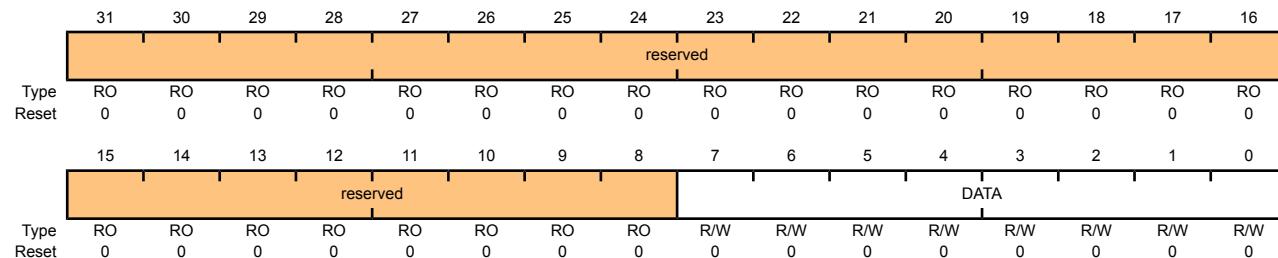
This register contains the data to be transmitted when in the Master Transmit state, and the data received when in the Master Receive state.

### I2C Master Data (I2CMDR)

I2C Master 0 base: 0x4002.0000

Offset 0x008

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x00	Data Transferred Data transferred during transaction.

## Register 4: I<sup>2</sup>C Master Timer Period (I2CMTPR), offset 0x00C

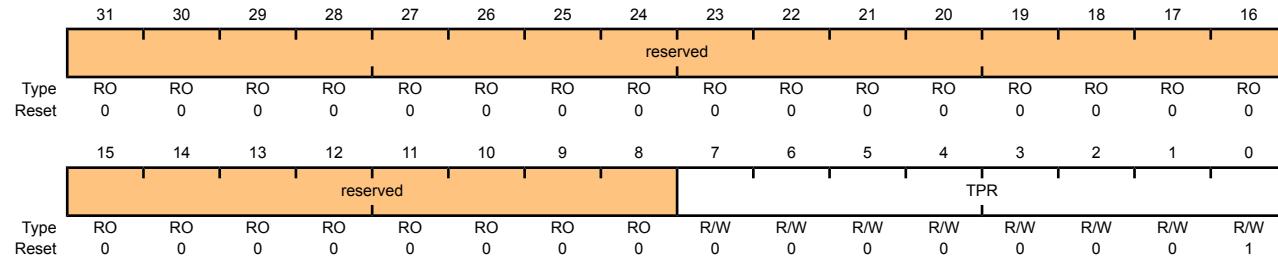
This register specifies the period of the SCL clock.

### I2C Master Timer Period (I2CMTPR)

I2C Master 0 base: 0x4002.0000

Offset 0x00C

Type R/W, 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	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 255).</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 Master 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 Master 0 base: 0x4002.0000

Offset 0x014

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	RIS	RO	0	Raw Interrupt Status This bit specifies the raw interrupt state (prior to masking) of the I <sup>2</sup> C master block. If set, an interrupt is pending; otherwise, an interrupt is not pending.

## Register 7: I<sup>2</sup>C Master Masked Interrupt Status (I2CMMIS), offset 0x018

This register specifies whether an interrupt was signaled.

### I2C Master Masked Interrupt Status (I2CMMIS)

I2C Master 0 base: 0x4002.0000

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

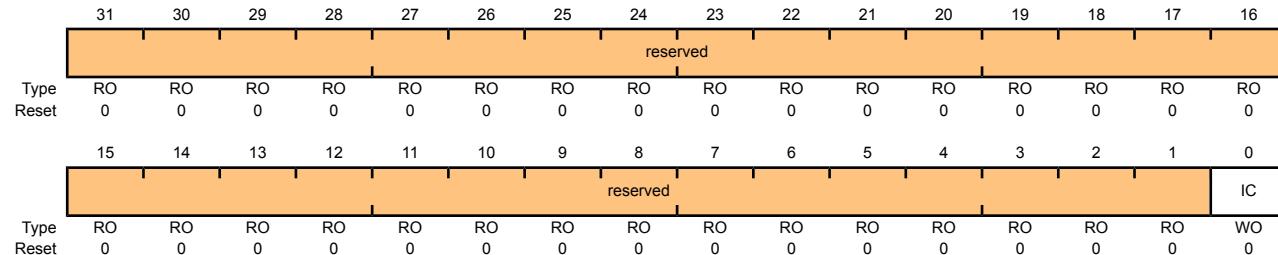
This register clears the raw interrupt.

### I2C Master Interrupt Clear (I2CMICR)

I2C Master 0 base: 0x4002.0000

Offset 0x01C

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	IC	WO	0	Interrupt Clear  This bit controls the clearing of the raw interrupt. A write of 1 clears the interrupt; otherwise, a write of 0 has no affect on the interrupt state. A read of this register returns no meaningful data.

## Register 9: I<sup>2</sup>C Master Configuration (I2CMCR), offset 0x020

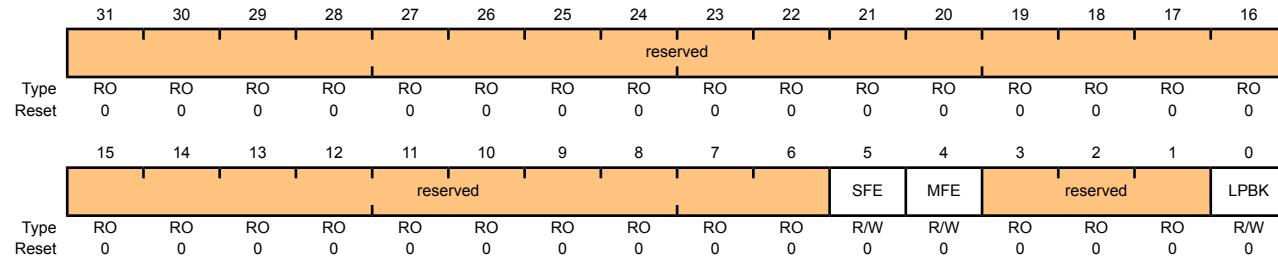
This register configures the mode (Master or Slave) and sets the interface for test mode loopback.

### I2C Master Configuration (I2CMCR)

I2C Master 0 base: 0x4002.0000

Offset 0x020

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SFE	R/W	0	I <sup>2</sup> C Slave Function Enable  This bit specifies whether the interface may operate in Slave mode. If set, Slave mode is enabled; otherwise, Slave mode is disabled.
4	MFE	R/W	0	I <sup>2</sup> C Master Function Enable  This bit specifies whether the interface may operate in Master mode. If set, Master mode is enabled; otherwise, Master mode is disabled and the interface clock is disabled.
3:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	LPBK	R/W	0	I <sup>2</sup> C Loopback  This bit specifies whether the interface is operating normally or in Loopback mode. If set, the device is put in a test mode loopback configuration; otherwise, the device operates normally.

## **15.6 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 389.

## Register 10: I<sup>2</sup>C Slave Own Address (I2CSOAR), offset 0x000

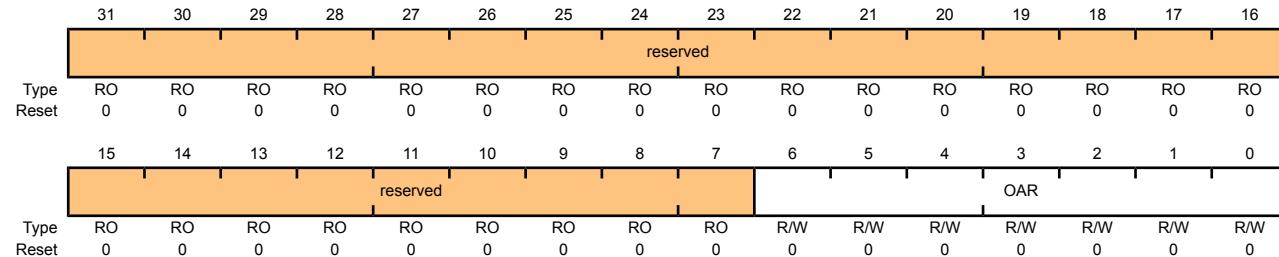
This register consists of seven address bits that identify the Stellaris® I<sup>2</sup>C device on the I<sup>2</sup>C bus.

### I<sup>2</sup>C Slave Own Address (I2CSOAR)

I<sup>2</sup>C Slave 0 base: 0x4002.0800

Offset 0x000

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	OAR	R/W	0x00	I <sup>2</sup> C Slave Own Address This field specifies bits A6 through A0 of the slave address.

## Register 11: I<sup>2</sup>C Slave Control/Status (I2CSCCSR), offset 0x004

This register accesses one control bit when written, and three status bits when read.

The read-only Status register consists of three bits: the FBR, RREQ, and TREQ bits. The First Byte Received (FBR) bit is set only after the Stellaris® 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.

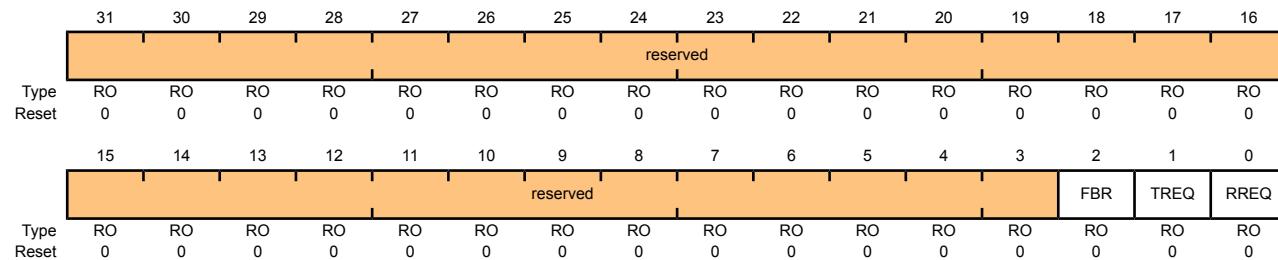
### Read-Only Status Register

#### I<sup>2</sup>C Slave Control/Status (I2CSCCSR)

I<sup>2</sup>C Slave 0 base: 0x4002.0800

Offset 0x004

Type RO, 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	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 I2CSDR 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 I2CSDR 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 I2CSDR register. Otherwise, no receive data is outstanding.

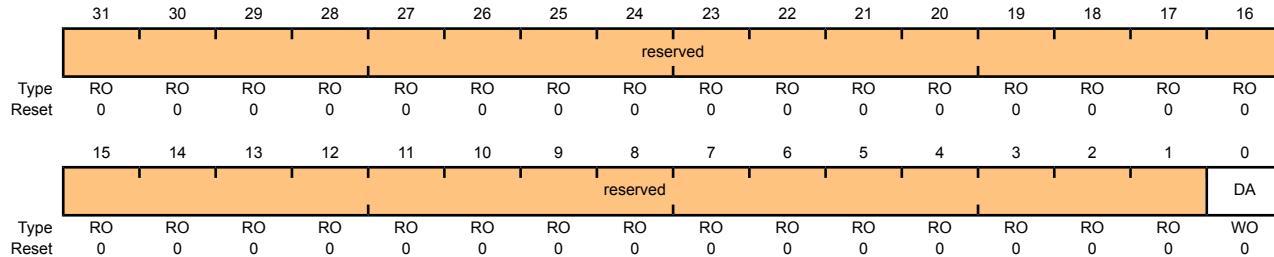
## Write-Only Control Register

### I<sup>2</sup>C Slave Control/Status (I2CSCCSR)

I<sup>2</sup>C Slave 0 base: 0x4002.0800

Offset 0x004

Type WO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	DA	WO	0	Device Active
		Value	Description	
		0	Disables the I <sup>2</sup> C slave operation.	
		1	Enables the I <sup>2</sup> C slave operation.	

## Register 12: I<sup>2</sup>C Slave Data (I2CSDR), offset 0x008

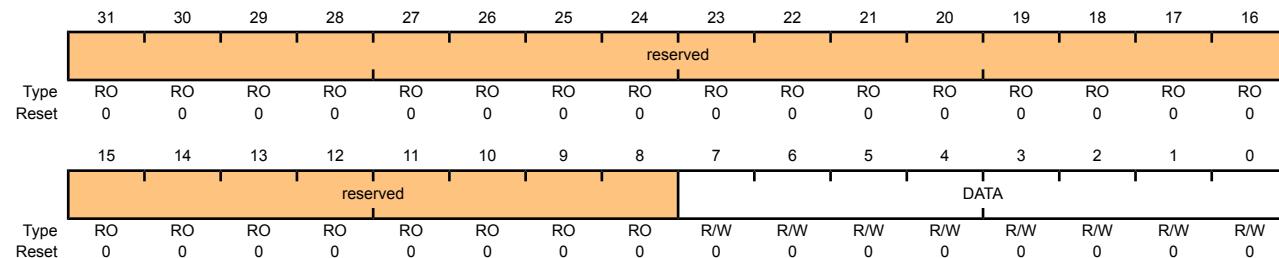
This register contains the data to be transmitted when in the Slave Transmit state, and the data received when in the Slave Receive state.

### I2C Slave Data (I2CSDR)

I2C Slave 0 base: 0x4002.0800

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	0x0	Data for Transfer This field contains the data for transfer during a slave receive or transmit operation.

## Register 13: I<sup>2</sup>C Slave Interrupt Mask (I2CSIMR), offset 0x00C

This register controls whether a raw interrupt is promoted to a controller interrupt.

### I2C Slave Interrupt Mask (I2CSIMR)

I2C Slave 0 base: 0x4002.0800

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															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 14: I<sup>2</sup>C Slave Raw Interrupt Status (I2CSRIS), offset 0x010

This register specifies whether an interrupt is pending.

### I2C Slave Raw Interrupt Status (I2CSRIS)

I2C Slave 0 base: 0x4002.0800

Offset 0x010

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	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 slave block. If set, an interrupt is pending; otherwise, an interrupt is not pending.

## Register 15: I<sup>2</sup>C Slave Masked Interrupt Status (I2CSMIS), offset 0x014

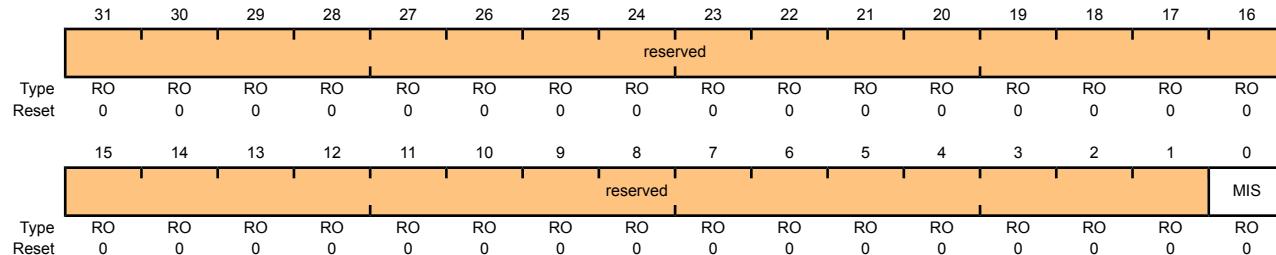
This register specifies whether an interrupt was signaled.

### I2C Slave Masked Interrupt Status (I2CSMIS)

I2C Slave 0 base: 0x4002.0800

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	MIS	RO	0	Masked Interrupt Status  This bit specifies the raw interrupt state (after masking) of the I <sup>2</sup> C slave block. If set, an interrupt was signaled; otherwise, an interrupt has not been generated since the bit was last cleared.

## Register 16: I<sup>2</sup>C Slave Interrupt Clear (I2CSICR), offset 0x018

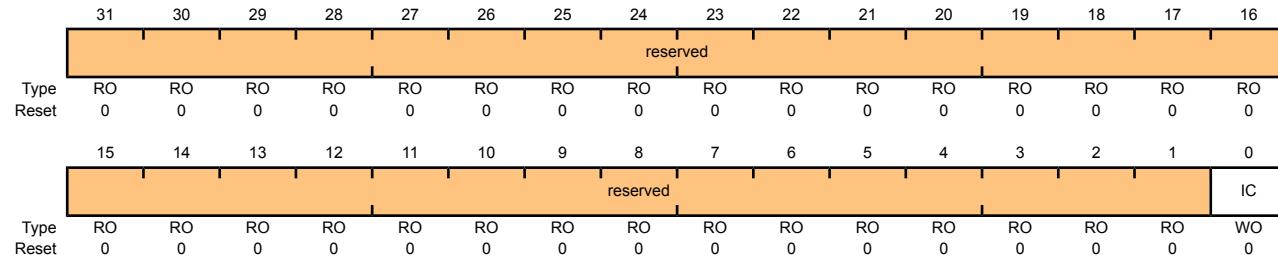
This register clears the raw interrupt.

### I<sup>2</sup>C Slave Interrupt Clear (I2CSICR)

I<sup>2</sup>C Slave 0 base: 0x4002.0800

Offset 0x018

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	IC	WO	0	<p>Clear Interrupt</p> <p>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.</p>

# 16 Controller Area Network (CAN) Module

## 16.1 Controller Area Network Overview

Controller Area Network (CAN) is a multicast shared serial bus standard for connecting electronic control units (ECUs). CAN was specifically designed to be robust in electromagnetically noisy environments and can utilize a differential balanced line like RS-485 or a more robust twisted-pair wire. Originally created for automotive purposes, it is also used in many embedded control applications (such as industrial and medical). Bit rates up to 1 Mbps are possible at network lengths below 40 meters. Decreased bit rates allow longer network distances (for example, 125 Kbps at 500 m).

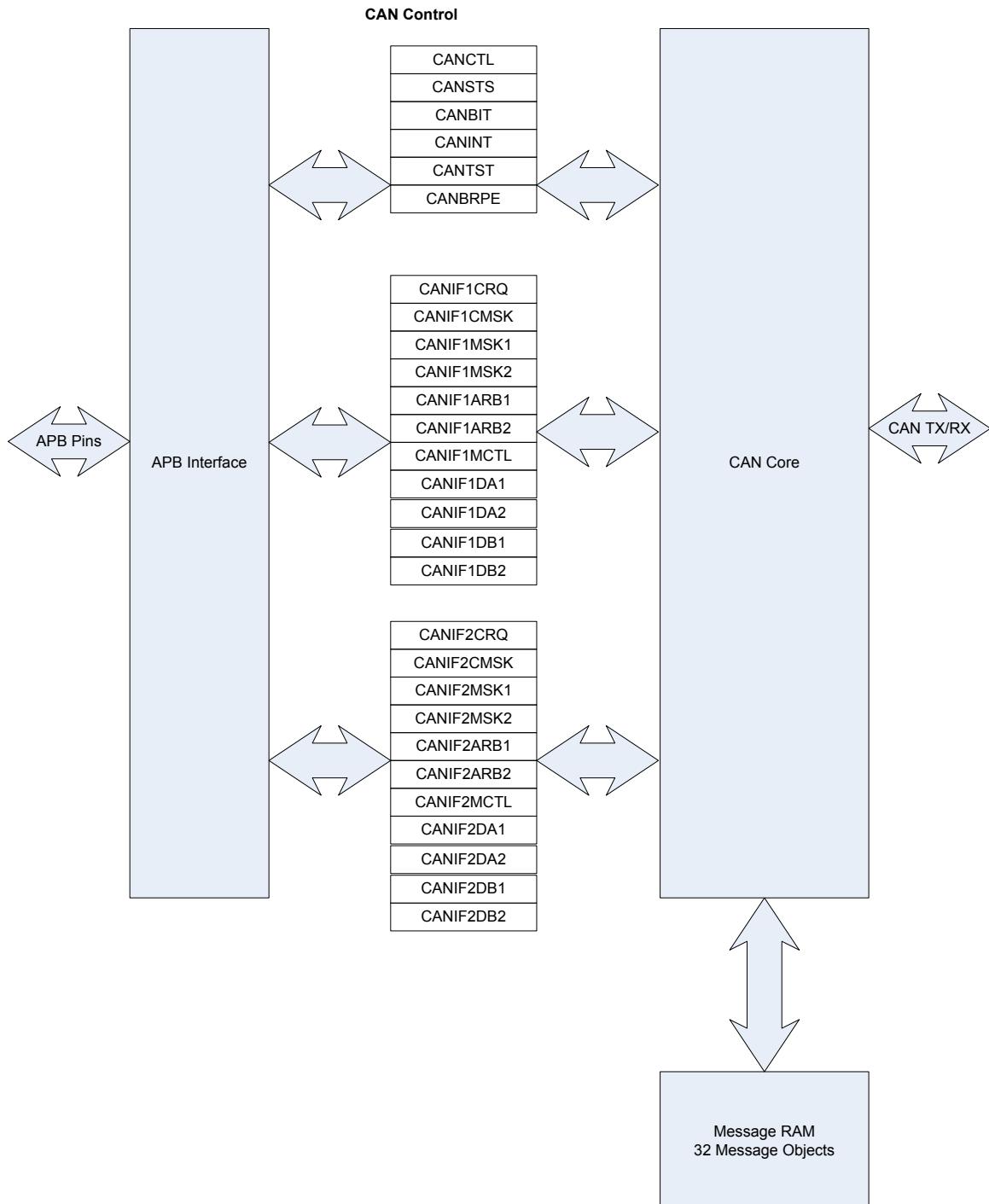
## 16.2 Controller Area Network Features

The Stellaris® CAN module supports the following features:

- CAN protocol version 2.0 part A/B
- Bit rates up to 1 Mbps
- 32 message objects
- Each message object has its own identifier mask
- Maskable interrupt
- Disable Automatic Retransmission mode for Time Triggered CAN (TTCAN) applications
- Programmable Loopback mode for self-test operation
- Programmable FIFO mode
- Gluelessly attach to an external CAN PHY through the CAN0Tx and CAN0Rx pins

## 16.3 Controller Area Network Block Diagram

Figure 16-1. CAN Module Block Diagram



## 16.4 Controller Area Network Functional Description

The CAN module conforms to the CAN protocol version 2.0 (parts A and B). Message transfers that include data, remote, error, and overload frames with an 11-bit identifier (standard) or a 29-bit identifier (extended) are supported. Transfer rates can be programmed up to 1 Mbps.

The CAN module consists of three major parts:

- CAN protocol controller and message handler
- Message memory
- CAN register interface

The protocol controller transfers and receives the serial data from the CAN bus and passes the data on to the message handler. The message handler then loads this information into the appropriate message object based on the current filtering and identifiers in the message object memory. The message handler is also responsible for generating interrupts based on events on the CAN bus.

The message object memory is a set of 32 identical memory blocks that hold the current configuration, status, and actual data for each message object. These are accessed via the CAN message object register interface. The message memory is not directly accessible in the Stellaris® memory map, so the Stellaris® CAN controller provides an interface to communicate with the message memory.

The CAN message object register interface provides two register sets for communicating with the message objects. Since there is no direct access to the message object memory, these two interfaces must be used to read or write to each message object. The two message object interfaces allow parallel access to the CAN controller message objects when multiple objects may have new information that needs to be processed.

### 16.4.1 Initialization

The software initialization is started by setting the **INIT** bit in the **CAN Control (CANCTL)** register, with software or by a hardware reset, or by going bus-off, which occurs when the transmitter's error counter exceeds a count of 255. While **INIT** is set, all message transfers to and from the CAN bus are stopped and the status of the CAN transmit output is recessive (High). Entering the initialization state does not change the configuration of the CAN controller, the message objects, or the error counters. However, some configuration registers are only accessible when in the initialization state.

To initialize the CAN controller, set the **CAN Bit Timing (CANBIT)** register and configure each message object. If a message object is not needed, it is sufficient to set it as not valid by clearing the **MsgVal** bit in the **CANIFnARB2** register. Otherwise, the whole message object has to be initialized, as the fields of the message object may not have valid information causing unexpected results. Access to the **CAN Bit Timing (CANBIT)** register and to the **CAN Baud Rate Prescalar Extension (CANBRPE)** register to configure the bit timing are enabled when both the **INIT** and **CCE** bits in the **CANCTL** register are set. To leave the initialization state, the **INIT** bit must be cleared. Afterwards, the internal Bit Stream Processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (Bus Idle) before it takes part in bus activities and starts message transfers. The initialization of the message objects is independent of being in the initialization state and can be done on the fly, but message objects should all be configured to particular identifiers or set to not valid before the BSP starts the message transfer. To change the configuration of a message object during normal operation, set the **MsgVal** bit in the **CANIFnARB2** register to 0 (not valid). When the configuration is completed, **MsgVal** is set to 1 again (valid).

### 16.4.2 Operation

Once the CAN module is initialized and the INIT bit in the **CANCTL** register is reset to 0, the CAN module synchronizes itself to the CAN bus and starts the message transfer. As messages are received, they are stored in their appropriate message objects if they pass the message handler's filtering. The whole message (including all arbitration bits, data-length code, and eight data bytes) is stored in the message object. If the Identifier Mask (the Msk bits in the **CANIFnMSKn** registers) is used, the arbitration bits which are masked to "don't care" may be overwritten in the message object.

The CPU may read or write each message any time via the CAN Interface Registers (**CANIFnCRQ**, **CANIFnCMSK**, **CANIFnMSKn**, **CANIFnARBn**, **CANIFnMCTL**, **CANIFnDAn**, and **CANIFnDBn**). The message handler guarantees data consistency in case of concurrent accesses.

The transmission of message objects are under the control of the software that is managing the CAN hardware. These can be message objects used for one-time data transfers, or permanent message objects used to respond in a more periodic manner. Permanent message objects have all arbitration and control set up, and only the data bytes are updated. To start the transmission, the TxRqst bit in the **CANTXRQn** register and the NewDat bit in the **CANNWDAn** register are set. If several transmit messages are assigned to the same message object (when the number of message objects is not sufficient), the whole message object has to be configured before the transmission of this message is requested.

The transmission of any number of message objects may be requested at the same time; they are transmitted according to their internal priority, which is based on the message identifier for the message object. Messages may be updated or set to not valid any time, even when their requested transmission is still pending. The old data is discarded when a message is updated before its pending transmission has started. Depending on the configuration of the message object, the transmission of a message may be requested autonomously by the reception of a remote frame with a matching identifier.

There are two sets of CAN Interface Registers (**CANIFn1x** and **CANIFn2x**), which are used to access the Message Objects in the Message RAM. The CAN controller coordinates transfers to and from the Message RAM to and from the registers. The function of the two sets are independent and identical and can be used to queue transactions.

### 16.4.3 Transmitting Message Objects

If the internal transmit shift register of the CAN module is ready for loading, and if there is no data transfer between the CAN Interface Registers and message RAM, the valid message object with the highest priority and that has a pending transmission request is loaded into the transmit shift register by the message handler and the transmission is started. The message object's NewDat bit is reset and can be viewed in the **CANNWDAn** register. After a successful transmission, and if no new data was written to the message object since the start of the transmission, the TxRqst bit in the **CANIFnCMSK** register is reset. If the TxIE bit in the **CANIFnMCTL** register is set, the IntPnd bit in the **CANIFnMCTL** register is set after a successful transmission. If the CAN module has lost the arbitration or if an error occurred during the transmission, the message is re-transmitted as soon as the CAN bus is free again. If, meanwhile, the transmission of a message with higher priority has been requested, the messages are transmitted in the order of their priority.

### 16.4.4 Configuring a Transmit Message Object

Table 16-1 on page 415 specifies the bit settings for a transmit message object.

**Table 16-1. Transmit Message Object Bit Settings**

Register	CANIFnARB2	CANIFnCMSK			CANIFnMCTL	CANIFnARB2	CANIFnMCTL						
Bit	MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
Value	1	appl	appl	appl	1	1	0	0	0	appl	0	appl	0

The `Xtd` and `ID` bit fields in the **CANIFnARBn** registers are set by an application. They define the identifier and type of the outgoing message. If an 11-bit Identifier (Standard Frame) is used, it is programmed to bits [28:18] of **CANIFnARB1**, as bits 17:0 of **CANIFnARBn** are not used by the CAN controller for 11-bit identifiers.

If the `TxIE` bit is set, the `IntPnd` bit is set after a successful transmission of the message object.

If the `RmtEn` bit is set, a matching received Remote Frame causes the `TxRqst` bit to be set and the Remote Frame is autonomously answered by a Data Frame with the data from the message object.

The `DLC` bit in the **CANIFnMCTL** register is set by an application. `TxRqst` and `RmtEn` may not be set before the data is valid.

The CAN mask registers (`Msk` bits in **CANIFnMSKn**, `UMask` bit in **CANIFnMCTL** register, and `MXtd` and `MDir` bits in **CANIFnMSK2** register) may be used (`UMask=1`) to allow groups of Remote Frames with similar identifiers to set the `TxRqst` bit. The `Dir` bit should not be masked.

#### 16.4.5 Updating a Transmit Message Object

The CPU may update the data bytes of a Transmit Message Object any time via the CAN Interface Registers and neither the `MsgVal` nor the `TxRqst` bits have to be reset before the update.

Even if only a part of the data bytes are to be updated, all four bytes of the corresponding **CANIFnDAn** or **CANIFnDBn** register have to be valid before the content of that register is transferred to the message object. Either the CPU has to write all four bytes into the **CANIFnDAn** or **CANIFnDBn** register or the message object is transferred to the **CANIFnDAn** or **CANIFnDBn** register before the CPU writes the new data bytes.

In order to only update the data in a message object, the `WR`, `NewDat`, `DataA`, and `DataB` bits are written to the **CAN IFn Command Mask (CANIFnMSKn)** register, followed by writing the **CAN IFn Data** registers, and then the number of the message object is written to the **CAN IFn Command Request (CANIFnCRQ)** register, to update the data bytes and the `TxRqst` bit at the same time.

To prevent the reset of `TxRqst` at the end of a transmission that may already be in progress while the data is updated, `NewDat` has to be set together with `TxRqst`. When `NewDat` is set together with `TxRqst`, `NewDat` is reset as soon as the new transmission has started.

#### 16.4.6 Accepting Received Message Objects

When the arbitration and control field (`ID + Xtd + RmtEn + DLC`) of an incoming message is completely shifted into the CAN module, the message handling capability of the module starts scanning the message RAM for a matching valid message object. To scan the message RAM for a matching message object, the Acceptance Filtering unit is loaded with the arbitration bits from the core. Then the arbitration and mask fields (including `MsgVal`, `UMask`, `NewDat`, and `EoB`) of message object 1 are loaded into the Acceptance Filtering unit and compared with the arbitration field from the shift register. This is repeated with each following message object until a matching message object is found or until the end of the message RAM is reached. If a match occurs, the scanning is stopped and the message handler proceeds depending on the type of frame received.

### 16.4.7 Receiving a Data Frame

The message handler stores the message from the CAN module receive shift register into the respective message object in the message RAM. It stores the data bytes, all arbitration bits, and the Data Length Code into the corresponding message object. This is implemented to keep the data bytes connected with the identifier even if arbitration mask registers are used. The CANIFnMCTL.NewDat bit is set to indicate that new data has been received. The CPU should reset CANIFnMCTL.NewDat when it reads the message object to indicate to the controller that the message has been received and the buffer is free to receive more messages. If the CAN controller receives a message and the CANIFnMCTL.NewDat bit was already set, the MsgLst bit is set to indicate that the previous data was lost. If the CANIFnMCTL.RxIE bit is set, the CANIFnMCTL.IntPnd bit is set, causing the **CANINT** interrupt register to point to the message object that just received a message. The CANIFnMCTL.TxRqst bit of this message object is reset to prevent the transmission of a Remote Frame, while the requested Data Frame has just been received.

### 16.4.8 Receiving a Remote Frame

When a Remote Frame is received, three different configurations of the matching message object have to be considered:

- Dir = 1 (direction = transmit), RmtEn = 1, UMask = 1 or 0

At the reception of a matching Remote Frame, the TxRqst bit of this message object is set. The rest of the message object remains unchanged.

- Dir = 1 (direction = transmit), RmtEn = 0, UMask = 0

At the reception of a matching Remote Frame, the TxRqst bit of this message object remains unchanged; the Remote Frame is ignored. This remote frame is disabled and will not automatically respond or indicate that the remote frame ever happened.

- Dir = 1 (direction = transmit), RmtEn = 0, UMask = 1

At the reception of a matching Remote Frame, the TxRqst bit of this message object is reset. The arbitration and control field (ID + Xtd + RmtEn + DLC) from the shift register is stored into the message object in the message RAM and the NewDat bit of this message object is set. The data field of the message object remains unchanged; the Remote Frame is treated similar to a received Data Frame. This is useful for a remote data request from another CAN device for which the Stellaris® controller does not have readily available data. The software must fill the data and answer the frame manually.

### 16.4.9 Receive/Transmit Priority

The receive/transmit priority for the message objects is controlled by the message number. Message object 1 has the highest priority, while message object 32 has the lowest priority. If more than one transmission request is pending, the message objects are transmitted in order based on the message object with the lowest message number. This should not be confused with the message identifier as that priority is enforced by the CAN bus. This means that if message object 1 and message object 2 both have valid messages that need to be transmitted, message object 1 will always be transmitted first regardless of the message identifier in the message object itself.

### 16.4.10 Configuring a Receive Message Object

Table 16-2 on page 417 specifies the bit settings for a transmit message object.

**Table 16-2. Receive Message Object Bit Settings**

Register	CANIFnARB2	CANIFnCMSK			CANIFnMCTL	CANIFnARB2	CANIFnMCTL						
Bit	MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
Value	1	appl	appl	appl	1	0	0	0	appl	0	0	0	0

The `Xtd` and `ID` bit fields in the **CANIFnARBn** registers are set by an application. They define the identifier and type of accepted received messages. If an 11-bit Identifier (Standard Frame) is used, it is programmed to bits [28:18] of **CANIFnARB1**, and bits [17:0] are ignored by the CAN controller. When a Data Frame with an 11-bit Identifier is received, bits [17:0] are set to 0.

If the `RxIE` bit is set, the `IntPnd` bit is set when a received Data Frame is accepted and stored in the message object.

When the message handler stores a Data Frame in the message object, it stores the received Data Length Code and eight data bytes. If the Data Length Code is less than 8, the remaining bytes of the message object are overwritten by nonspecified values.

The CAN mask registers (`Msk` bits in **CANIFnMSKn**, `UMask` bit in **CANIFnMCTL** register, and `MXtd` and `MDir` bits in **CANIFnMSK2** register) may be used (`UMask=1`) to allow groups of Data Frames with similar identifiers to be accepted. The `Dir` bit should not be masked in typical applications.

#### 16.4.11 Handling of Received Message Objects

The CPU may read a received message any time via the CAN Interface registers because the data consistency is guaranteed by the message handler state machine.

Typically, the CPU first writes 0x007F to the **CAN IFn Command Mask (CANIFnCMSK)** register and then writes the number of the message object to the **CAN IFn Command Request (CANIFnCRQ)** register. That combination transfers the whole received message from the message RAM into the Message Buffer registers (**CANIFnMSKn**, **CANIFnARBn**, and **CANIFnMCTL**). Additionally, the `NewDat` and `IntPnd` bits are cleared in the message RAM, acknowledging that the message has been read and clearing the pending interrupt being generated by this message object.

If the message object uses masks for acceptance filtering, the arbitration bits show which of the matching messages has been received.

The actual value of `NewDat` shows whether a new message has been received since the last time this message object was read. The actual value of `MsgLst` shows whether more than one message has been received since the last time this message object was read. `MsgLst` is not automatically reset.

Using a Remote Frame, the CPU may request new data from another CAN node on the CAN bus. Setting the `TxRqst` bit of a receive object causes the transmission of a Remote Frame with the receive object's identifier. This Remote Frame triggers the other CAN node to start the transmission of the matching Data Frame. If the matching Data Frame is received before the Remote Frame could be transmitted, the `TxRqst` bit is automatically reset. This prevents the possible loss of data when the other device on the CAN bus has already transmitted the data, slightly earlier than expected.

#### 16.4.12 Handling of Interrupts

If several interrupts are pending, the **CAN Interrupt (CANINT)** register points to the pending interrupt with the highest priority, disregarding their chronological order. An interrupt remains pending until the CPU has cleared it.

The Status Interrupt has the highest priority. Among the message interrupts, the message object's interrupt priority decreases with increasing message number. A message interrupt is cleared by clearing the message object's IntPnd bit. The Status Interrupt is cleared by reading the **CAN Status (CANSTS)** register.

The interrupt identifier IntId in the **CANINT** register indicates the cause of the interrupt. When no interrupt is pending, the register holds the value to 0. If the value of **CANINT** is different from 0, then there is an interrupt pending. If the IE bit is set in the **CANCTL** register, the interrupt line to the CPU is active. The interrupt line remains active until **CANINT** is 0, all interrupt sources have been cleared, (the cause of the interrupt is reset), or until IE is reset, which disables interrupts from the CAN controller.

The value 0x8000 in the **CANINT** register indicates that an interrupt is pending because the CAN module has updated, but not necessarily changed, the **CANSTS** register (Error Interrupt or Status Interrupt). This indicates that there is either a new Error Interrupt or a new Status Interrupt. A write access can clear the RxOK, TxOK, and LEC flags in the **CANSTS** register, however, only a read access to the **CANSTS** register will clear the source of the status interrupt.

IntId points to the pending message interrupt with the highest interrupt priority. The SIE bit in the **CANCTL** register controls whether a change of the status register may cause an interrupt. The EIE bit in the **CANCTL** register controls whether any interrupt from the CAN controller actually generates an interrupt to the microcontroller's interrupt controller. The **CANINT** interrupt register is updated even when the IE bit is set to zero.

There are two possibilities when handling the source of a message interrupt. The first is to read the IntId bit in the **CANINT** interrupt register to determine the highest priority interrupt that is pending, and the second is to read the **CAN Message Interrupt Pending (CANMSGnINT)** register to see all of the message objects that have pending interrupts.

An interrupt service routine reading the message that is the source of the interrupt may read the message and reset the message object's IntPnd at the same time by setting the ClrIntPnd bit in the **CANIFn Command Mask (CANIFnCMSK)** register. When the IntPnd bit is cleared, the **CANINT** register will contain the message number for the next message object with a pending interrupt.

#### 16.4.13 Bit Timing Configuration Error Considerations

Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly. In many cases, the CAN bit synchronization amends a faulty configuration of the CAN bit timing to such a degree that only occasionally an error frame is generated. In the case of arbitration, however, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point may cause one of the transmitters to become error passive. The analysis of such sporadic errors requires a detailed knowledge of the CAN bit synchronization inside a CAN node and of the CAN nodes' interaction on the CAN bus.

#### 16.4.14 Bit Time and Bit Rate

The CAN system supports bit rates in the range of lower than 1 Kbps up to 1000 Kbps. Each member of the CAN network has its own clock generator. The timing parameter of the bit time can be configured individually for each CAN node, creating a common bit rate even though the CAN nodes' oscillator periods may be different.

Because of small variations in frequency caused by changes in temperature or voltage and by deteriorating components, these oscillators are not absolutely stable. As long as the variations

remain inside a specific oscillator's tolerance range, the CAN nodes are able to compensate for the different bit rates by periodically resynchronizing to the bit stream.

According to the CAN specification, the bit time is divided into four segments (see Figure 16-2 on page 419): the Synchronization Segment, the Propagation Time Segment, the Phase Buffer Segment 1, and the Phase Buffer Segment 2. Each segment consists of a specific, programmable number of time quanta (see Table 16-3 on page 419). The length of the time quantum ( $t_q$ ), which is the basic time unit of the bit time, is defined by the CAN controller's system clock ( $f_{sys}$ ) and the Baud Rate Prescaler (BRP):

$$t_q = \text{BRP} / f_{sys}$$

The CAN module's system clock  $f_{sys}$  is the frequency of its CAN module clock (CAN\_CLK) input.

The Synchronization Segment Sync\_Seg is that part of the bit time where edges of the CAN bus level are expected to occur; the distance between an edge that occurs outside of Sync\_Seg and the Sync\_Seg is called the *phase error* of that edge.

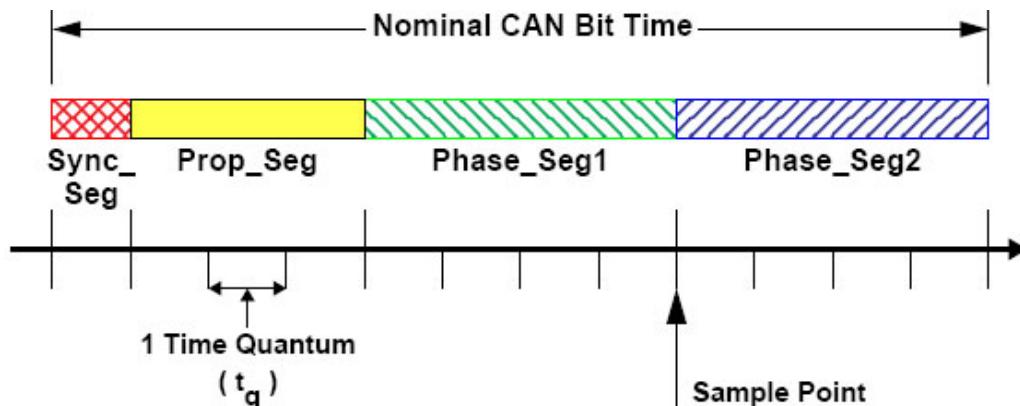
The Propagation Time Segment Prop\_Seg is intended to compensate for the physical delay times within the CAN network.

The Phase Buffer Segments Phase\_Seg1 and Phase\_Seg2 surround the Sample Point.

The (Re-)Synchronization Jump Width (SJW) defines how far a resynchronization may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.

A given bit rate may be met by different bit-time configurations, but for the proper function of the CAN network, the physical delay times and the oscillator's tolerance range have to be considered.

**Figure 16-2. CAN Bit Time**



**Table 16-3. CAN Protocol Ranges<sup>a</sup>**

Parameter	Range	Remark
BRP	[1 .. 32]	Defines the length of the time quantum $t_q$
Sync_Seg	1 $t_q$	Fixed length, synchronization of bus input to system clock
Prop_Seg	[1 .. 8] $t_q$	Compensates for the physical delay times
Phase_Seg1	[1 .. 8] $t_q$	May be lengthened temporarily by synchronization
Phase_Seg2	[1 .. 8] $t_q$	May be shortened temporarily by synchronization

Parameter	Range	Remark
SJW	[1 .. 4] $t_q$	May not be longer than either Phase Buffer Segment

a. This table describes the minimum programmable ranges required by the CAN protocol.

The bit timing configuration is programmed in two register bytes in the **CANBIT** register. The sum of **Prop\_Seg** and **Phase\_Seg1** (as **TSEG1**) is combined with **Phase\_Seg2** (as **TSEG2**) in one byte, and **SJW** and **BRP** are combined in the other byte.

In these bit timing registers, the four components **TSEG1**, **TSEG2**, **SJW**, and **BRP** have to be programmed to a numerical value that is one less than its functional value; so instead of values in the range of [1..n], values in the range of [0..n-1] are programmed. That way, for example, **SJW** (functional range of [1..4]) is represented by only two bits. Therefore, the length of the bit time is (programmed values):

$$[TSEG1 + TSEG2 + 3] \ t_q$$

or (functional values):

$$[Sync\_Seg + Prop\_Seg + Phase\_Seg1 + Phase\_Seg2] \ t_q$$

The data in the bit timing registers are the configuration input of the CAN protocol controller. The Baud Rate Prescalar (configured by **BRP**) defines the length of the time quantum, the basic time unit of the bit time; the Bit Timing Logic (configured by **TSEG1**, **TSEG2**, and **SJW**) defines the number of time quanta in the bit time.

The processing of the bit time, the calculation of the position of the Sample Point, and occasional synchronizations are controlled by the CAN controller and are evaluated once per time quantum.

The CAN controller translates messages to and from frames. It generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the error management, and decides which type of synchronization is to be used. It is evaluated at the Sample Point and processes the sampled bus input bit. The time after the Sample Point that is needed to calculate the next bit to be sent (that is, the data bit, CRC bit, stuff bit, error flag, or idle) is called the Information Processing Time (IPT).

The IPT is application-specific but may not be longer than 2  $t_q$ ; the CAN's IPT is 0  $t_q$ . Its length is the lower limit of the programmed length of **Phase\_Seg2**. In case of synchronization, **Phase\_Seg2** may be shortened to a value less than IPT, which does not affect bus timing.

### 16.4.15 Calculating the Bit Timing Parameters

Usually, the calculation of the bit timing configuration starts with a desired bit rate or bit time. The resulting bit time (1/bit rate) must be an integer multiple of the system clock period.

The bit time may consist of 4 to 25 time quanta. Several combinations may lead to the desired bit time, allowing iterations of the following steps.

The first part of the bit time to be defined is the **Prop\_Seg**. Its length depends on the delay times measured in the system. A maximum bus length as well as a maximum node delay has to be defined for expandable CAN bus systems. The resulting time for **Prop\_Seg** is converted into time quanta (rounded up to the nearest integer multiple of  $t_q$ ).

The **Sync\_Seg** is 1  $t_q$  long (fixed), which leaves (bit time - **Prop\_Seg** - 1)  $t_q$  for the two Phase Buffer Segments. If the number of remaining  $t_q$  is even, the Phase Buffer Segments have the same length, that is, **Phase\_Seg2** = **Phase\_Seg1**, else **Phase\_Seg2** = **Phase\_Seg1** + 1.

The minimum nominal length of Phase\_Seg2 has to be regarded as well. Phase\_Seg2 may not be shorter than the CAN controller's Information Processing Time, which is, depending on the actual implementation, in the range of [0..2] tq.

The length of the Synchronization Jump Width is set to its maximum value, which is the minimum of 4 and Phase\_Seg1.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formula given below:

$$(1 - df) \times f_{nom} \leq f_{osc} \leq (1 + df) \times f_{nom}$$

where:

- df = maximum tolerance of oscillator frequency
- f<sub>osc</sub> = actual oscillator frequency
- f<sub>nom</sub> = nominal oscillator frequency

Maximum frequency tolerance must take into account the following formulas:

$$\begin{aligned} df &\leq (\text{Phase\_Seg1}, \text{Phase\_Seg2})_{\min} / 2 \times (13 \times t_{bit} - \text{Phase\_Seg2}) \\ df_{max} &= 2 \times df \times f_{nom} \end{aligned}$$

where:

- Phase\_Seg1 and Phase\_Seg2 are from Table 16-3 on page 419
- t<sub>bit</sub> = Bit Time
- df<sub>max</sub> = maximum difference between two oscillators

If more than one configuration is possible, that configuration allowing the highest oscillator tolerance range should be chosen.

CAN nodes with different system clocks require different configurations to come to the same bit rate. The calculation of the propagation time in the CAN network, based on the nodes with the longest delay times, is done once for the whole network.

The CAN system's oscillator tolerance range is limited by the node with the lowest tolerance range.

The calculation may show that bus length or bit rate have to be decreased or that the oscillator frequencies' stability has to be increased in order to find a protocol-compliant configuration of the CAN bit timing.

The resulting configuration is written into the **CAN Bit Timing (CANBIT)** register :

(Phase\_Seg2-1) & (Phase\_Seg1+Prop\_Seg-1) & (SynchronizationJumpWidth-1) & (Prescaler-1)

#### 16.4.15.1 Example for Bit Timing at High Baud Rate

In this example, the frequency of CAN\_CLK is 10 MHz, BRP is 0, and the bit rate is 1 Mbps.

```
tq 100 ns = tCAN_CLK
delay of bus driver 50 ns
delay of receiver circuit 30 ns
delay of bus line (40m) 220 ns
```

```

tProp 600 ns = 6 × tq
tSJW 100 ns = 1 × tq
tTSeg1 700 ns = tProp + tSJW
tTSeg2 200 ns = Information Processing Time + 1 × tq
tSync-Seg 100 ns = 1 × tq
bit time 1000 ns = tSync-Seg + tTSeg1 + tTSeg2
tolerance for CAN_CLK 0.39 % =
    min(PB1,PB2)/ 2 × (13 x bit time - PB2) =
        0.1us/ 2 × (13x 1us - 2us)

```

In the above example, the concatenated bit time parameters are (2-1)3&(7-1)4&(1-1)2&(1-1)6, and **CANBIT** is programmed to 0x1600.

#### 16.4.15.2 Example for Bit Timing at Low Baud Rate

In this example, the frequency of CAN\_CLK is 2 MHz, BRP is 1, and the bit rate is 100 Kbps.

```

tq 1 ms = 2 × tCAN_CLK
delay of bus driver 200 ns
delay of receiver circuit 80 ns
delay of bus line (40m) 220 ns
tProp 1 ms = 1 × tq
tSJW 4 ms = 4 × tq
tTSeg1 5 ms = tProp + tSJW
tTSeg2 4 ms = Information Processing Time + 3 × tq
tSync-Seg 1 ms = 1 × tq
bit time 10 ms = tSync-Seg + tTSeg1 + tTSeg2
tolerance for CAN_CLK 1.58 % =
    min(PB1,PB2)/ 2 x (13 x bit time - PB2) =
        4us/ 2 x (13 x 10us - 4us)

```

In this example, the concatenated bit time parameters are (4-1)3&(5-1)4&(4-1)2&(2-1)6, and **CANBIT** is programmed to 0x34C1.

### 16.5 Controller Area Network Register Map

Table 16-4 on page 422 lists the registers. All addresses given are relative to the CAN base address of:

- CAN0: 0x4004.0000

All accesses are on word (32-bit) boundaries.

**Table 16-4. CAN Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	CANCTL	R/W	0x0000.0001	CAN Control	425
0x004	CANSTS	R/W	0x0000.0000	CAN Status	427
0x008	CANERR	RO	0x0000.0000	CAN Error Counter	430
0x00C	CANBIT	R/W	0x0000.2301	CAN Bit Timing	431
0x010	CANINT	RO	0x0000.0000	CAN Interrupt	433

Offset	Name	Type	Reset	Description	See page
0x014	CANTST	R/W	0x0000.0000	CAN Test	434
0x018	CANBRPE	R/W	0x0000.0000	CAN Baud Rate Prescalar Extension	436
0x020	CANIF1CRQ	R/W	0x0000.0001	CAN IF1 Command Request	437
0x024	CANIF1CMSK	R/W	0x0000.0000	CAN IF1 Command Mask	438
0x028	CANIF1MSK1	R/W	0x0000.FFFF	CAN IF1 Mask 1	441
0x02C	CANIF1MSK2	R/W	0x0000.FFFF	CAN IF1 Mask 2	442
0x030	CANIF1ARB1	R/W	0x0000.0000	CAN IF1 Arbitration 1	443
0x034	CANIF1ARB2	R/W	0x0000.0000	CAN IF1 Arbitration 2	444
0x038	CANIF1MCTL	R/W	0x0000.0000	CAN IF1 Message Control	445
0x03C	CANIF1DA1	R/W	0x0000.0000	CAN IF1 Data A1	447
0x040	CANIF1DA2	R/W	0x0000.0000	CAN IF1 Data A2	447
0x044	CANIF1DB1	R/W	0x0000.0000	CAN IF1 Data B1	447
0x048	CANIF1DB2	R/W	0x0000.0000	CAN IF1 Data B2	447
0x080	CANIF2CRQ	R/W	0x0000.0001	CAN IF2 Command Request	437
0x084	CANIF2CMSK	R/W	0x0000.0000	CAN IF2 Command Mask	438
0x088	CANIF2MSK1	R/W	0x0000.FFFF	CAN IF2 Mask 1	441
0x08C	CANIF2MSK2	R/W	0x0000.FFFF	CAN IF2 Mask 2	442
0x090	CANIF2ARB1	R/W	0x0000.0000	CAN IF2 Arbitration 1	443
0x094	CANIF2ARB2	R/W	0x0000.0000	CAN IF2 Arbitration 2	444
0x098	CANIF2MCTL	R/W	0x0000.0000	CAN IF2 Message Control	445
0x09C	CANIF2DA1	R/W	0x0000.0000	CAN IF2 Data A1	447
0x0A0	CANIF2DA2	R/W	0x0000.0000	CAN IF2 Data A2	447
0x0A4	CANIF2DB1	R/W	0x0000.0000	CAN IF2 Data B1	447
0x0A8	CANIF2DB2	R/W	0x0000.0000	CAN IF2 Data B2	447
0x100	CANTXRQ1	RO	0x0000.0000	CAN Transmission Request 1	448
0x104	CANTXRQ2	RO	0x0000.0000	CAN Transmission Request 2	448
0x120	CANNWDA1	RO	0x0000.0000	CAN New Data 1	449
0x124	CANNWDA2	RO	0x0000.0000	CAN New Data 2	449
0x140	CANMSG1INT	RO	0x0000.0000	CAN Message 1 Interrupt Pending	450
0x144	CANMSG2INT	RO	0x0000.0000	CAN Message 2 Interrupt Pending	450
0x160	CANMSG1VAL	RO	0x0000.0000	CAN Message 1 Valid	451
0x164	CANMSG2VAL	RO	0x0000.0000	CAN Message 2 Valid	451

## 16.6 Register Descriptions

The remainder of this section lists and describes the CAN registers, in numerical order by address offset. There are two sets of Interface Registers which are used to access the Message Objects in the Message RAM: **CANIF1x** and **CANIF2x**. The function of the two sets are identical and are used to queue transactions.

## Register 1: CAN Control (CANCTL), offset 0x000

This control register initializes the module and enables test mode and interrupts.

The bus-off recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or resetting INIT. If the device goes bus-off, it sets INIT, stopping all bus activities. Once INIT has been cleared by the CPU, the device then waits for 129 occurrences of Bus Idle (129 \* 11 consecutive High bits) before resuming normal operations. At the end of the bus-off recovery sequence, the Error Management Counters are reset.

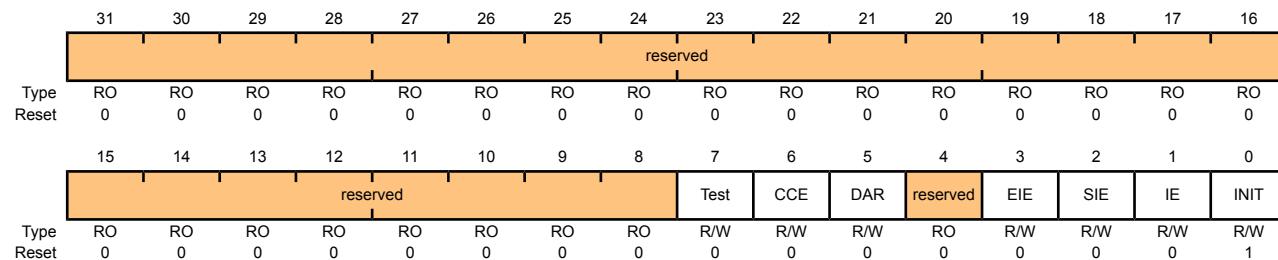
During the waiting time after INIT is reset, each time a sequence of 11 High bits has been monitored, a Bit0Error code is written to the **CANSTS** status register, enabling the CPU to readily check whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the proceeding of the bus-off recovery sequence.

### CAN Control (CANCTL)

CAN0 base: 0x4004.0000

Offset 0x000

Type R/W, reset 0x0000.0001



Bit/Field	Name	Type	Reset	Description
31:8	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.
7	Test	R/W	0	Test Mode Enable 0: Normal Operation 1: Test Mode
6	CCE	R/W	0	Configuration Change Enable 0: Do not allow write access to the <b>CANBIT</b> register. 1: Allow write access to the <b>CANBIT</b> register if the INIT bit is 1.
5	DAR	R/W	0	Disable Automatic Retransmission 0: Auto retransmission of disturbed messages is enabled. 1: Auto retransmission is disabled.
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	EIE	R/W	0	Error Interrupt Enable 0: Disabled. No Error Status interrupt is generated. 1: Enabled. A change in the Boff or EWarn bits in the <b>CANSTS</b> register generates an interrupt.

Bit/Field	Name	Type	Reset	Description
2	SIE	R/W	0	<p>Status Change Interrupt Enable</p> <p>0: Disabled. No Status Change interrupt is generated.</p> <p>1: Enabled. An interrupt is generated when a message has successfully been transmitted or received, or a CAN bus error has been detected. A change in the TxOk or RxOk bits in the <b>CANSTS</b> register generates an interrupt.</p>
1	IE	R/W	0	<p>CAN Interrupt Enable</p> <p>0: Interrupt disabled.</p> <p>1: Interrupt enabled.</p>
0	INIT	R/W	1	<p>Initialization</p> <p>0: Normal operation.</p> <p>1: Initialization started.</p>

## Register 2: CAN Status (CANSTS), offset 0x004

The status register contains information for interrupt servicing such as Bus-Off, error count threshold, and error types.

The **LEC** field holds the code that indicates the type of the last error to occur on the CAN bus. This field is cleared to 0 when a message has been transferred (reception or transmission) without error. The unused error code 7 may be written by the CPU to check for updates.

An Error Interrupt is generated by the **BOff** and **EWarn** bits and a Status Change Interrupt is generated by the **RxOk**, **TxOk**, and **LEC** bits, assuming that the corresponding enable bits in the **CAN Control (CANCTL)** register are set. A change of the **EPass** bit or a write to the **RxOk**, **TxOk**, or **LEC** bits does not generate an interrupt.

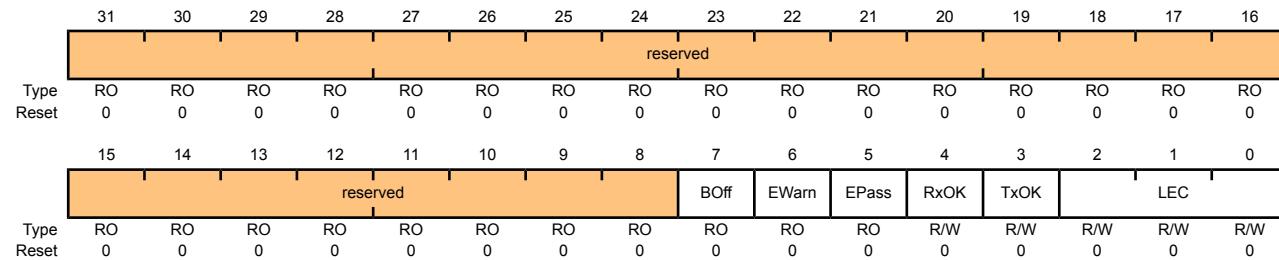
Reading the **CAN Status (CANSTS)** register clears the **CAN Interrupt (CANINT)** register, if it is pending.

### CAN Status (CANSTS)

CAN0 base: 0x4004.0000

Offset 0x004

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	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.
7	BOff	RO	0	Bus-Off Status 0: Module is not in bus-off state. 1: Module is in bus-off state.
6	EWarn	RO	0	Warning Status 0: Both error counters are below the error warning limit of 96. 1: At least one of the error counters has reached the error warning limit of 96.
5	EPass	RO	0	Error Passive 0: The CAN module is in the Error Active state, that is, the receive or transmit error count is less than or equal to 127. 1: The CAN module is in the Error Passive state, that is, the receive or transmit error count is greater than 127.

Bit/Field	Name	Type	Reset	Description
4	RxOK	R/W	0	<p>Received a Message Successfully</p> <p>0: Since this bit was last reset to 0, no message has been successfully received.</p> <p>1: Since this bit was last reset to 0, a message has been successfully received, independent of the result of the acceptance filtering.</p> <p>This bit is never reset by the CAN module.</p>
3	TxOK	R/W	0	<p>Transmitted a Message Successfully</p> <p>0: Since this bit was last reset to 0, no message has been successfully transmitted.</p> <p>1: Since this bit was last reset to 0, a message has been successfully transmitted error-free and acknowledged by at least one other node.</p> <p>This bit is never reset by the CAN module.</p>

Bit/Field	Name	Type	Reset	Description
2:0	LEC	R/W	0x0	Last Error Code  This is the type of the last error to occur on the CAN bus.
				Value Definition
			0x0	No Error
			0x1	Stuff Error  More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
			0x2	Form Error  A fixed format part of the received frame has the wrong format.
			0x3	ACK Error  The message transmitted was not acknowledged by another node.
			0x4	Bit 1 Error  When a message is transmitted, the CAN controller monitors the data lines to detect any conflicts. When the arbitration field is transmitted, data conflicts are a part of the arbitration protocol. When other frame fields are transmitted, data conflicts are considered errors.  A Bit 1 Error indicates that the device wanted to send a High level (logical 1) but the monitored bus value was Low (logical 0).
			0x5	Bit 0 Error  A Bit 0 Error indicates that the device wanted to send a Low level (logical 0) but the monitored bus value was High (logical 1).  During bus-off recovery, this status is set each time a sequence of 11 High bits has been monitored. This enables the CPU to monitor the proceeding of the bus-off recovery sequence without any disturbances to the bus.
			0x6	CRC Error  The CRC checksum was incorrect in the received message, indicating that the calculated value received did not match the calculated CRC of the data.
			0x7	Unused  When the LEC bit shows this value, no CAN bus event was detected since the CPU wrote this value to LEC.

## Register 3: CAN Error Counter (CANERR), offset 0x008

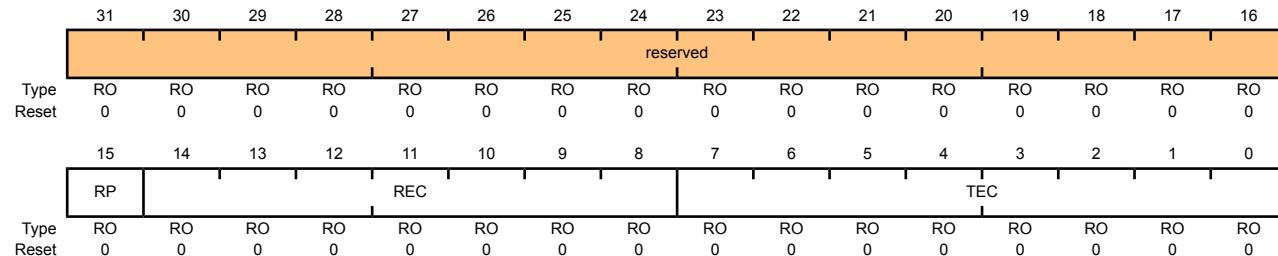
This register contains the error counter values, which can be used to analyze the cause of an error.

### CAN Error Counter (CANERR)

CAN0 base: 0x4004.0000

Offset 0x008

Type RO, 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	RP	RO	0	Received Error Passive 0: The Receive Error counter is below the Error Passive level (127 or less). 1: The Receive Error counter has reached the Error Passive level (128 or greater).
14:8	REC	RO	0x0	Receive Error Counter State of the receiver error counter (0 to 127).
7:0	TEC	RO	0x0	Transmit Error Counter State of the transmit error counter (0 to 255).

## Register 4: CAN Bit Timing (CANBIT), offset 0x00C

This register is used to program the bit width and bit quantum. Values are to be programmed to the system clock frequency. This register is write-enabled by the **CCE** and **INIT** bits in the **CANCTL** register.

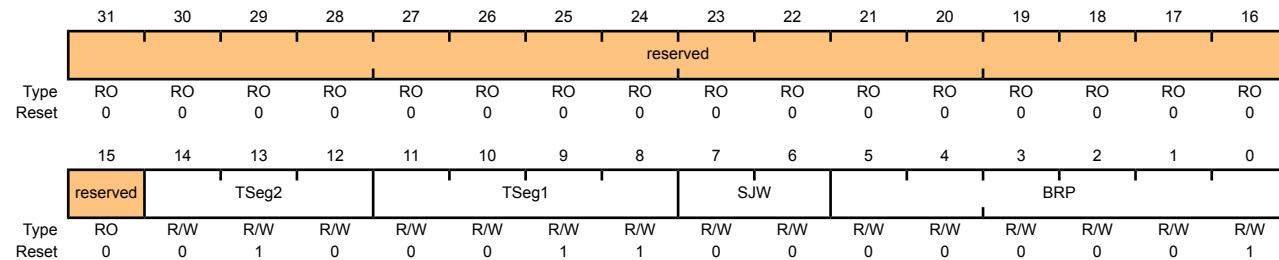
With a CAN module clock (CAN\_CLK) of 8 MHz, the register reset value of 0x230 configures the CAN for a bit rate of 500 Kbps.

### CAN Bit Timing (CANBIT)

CAN0 base: 0x4004.0000

Offset 0x00C

Type R/W, reset 0x0000.2301



Bit/Field	Name	Type	Reset	Description
31:15	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.
14:12	TSeg2	R/W	0x2	<p>Time Segment after Sample Point</p> <p>0x00-0x07: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.</p> <p>So, for example, a reset value of 0x2 defines that there is 3(2+1) bit time quanta defined for Phase_Seg2 (see Figure 16-2 on page 419). The bit time quanta is defined by BRP.</p>
11:8	TSeg1	R/W	0x3	<p>Time Segment Before Sample Point</p> <p>0x00-0x0F: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.</p> <p>So, for example, the reset value of 0x3 defines that there is 4(3+1) bit time quanta defined for Phase_Seg1 (see Figure 16-2 on page 419). The bit time quanta is define by BRP.</p>
7:6	SJW	R/W	0x0	<p>(Re)Synchronization Jump Width</p> <p>0x00-0x03: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.</p> <p>During the start of frame (SOF), if the CAN controller detects a phase error (misalignment), it can adjust the length of TSeg2 or TSeg1 by the value in SJW. So the reset value of 0 adjusts the length by 1 bit time quanta.</p>

Bit/Field	Name	Type	Reset	Description
5:0	BRP	R/W	0x1	<p>Baud Rate Prescalar</p> <p>0x00-0x03F: The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quantum. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.</p> <p>BRP defines the number of CAN clock periods that make up 1 bit time quanta, so the reset value is 2 bit time quanta (1+1).</p> <p>The <b>BRPRE</b> register can be used to further divide the bit time.</p>

## Register 5: CAN Interrupt (CANINT), offset 0x010

This register indicates the source of the interrupt.

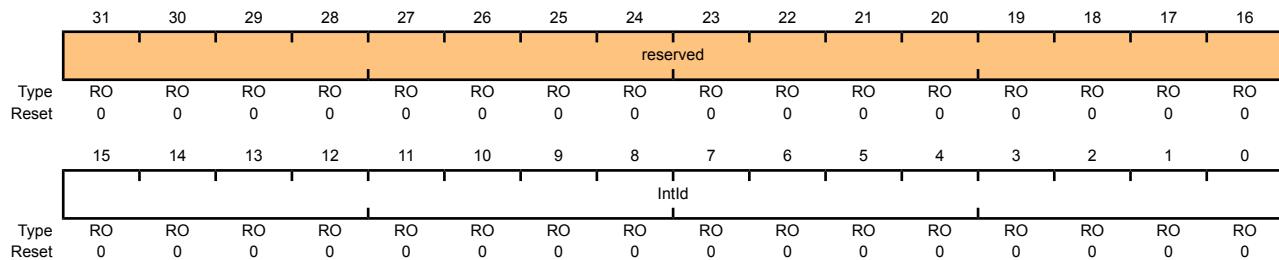
If several interrupts are pending, the **CAN Interrupt (CANINT)** register points to the pending interrupt with the highest priority, disregarding their chronological order. An interrupt remains pending until the CPU has cleared it. If the **IntId** bit is not 0x0000 (the default) and the **IE** bit in the **CANCTL** register is set, the interrupt is active. The interrupt line remains active until the **IntId** bit is set back to 0x0000 when the cause of all interrupts are reset or until **IE** is reset.

### CAN Interrupt (CANINT)

CANO base: 0x4004.0000

Offset 0x010

Type RO, 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	IntId	RO	0x0000	Interrupt Identifier The number in this field indicates the source of the interrupt.
		Value	Definition	
		0x0000	No interrupt pending	
		0x0001-0x0020	Number of the message object that caused the interrupt	
		0x0021-0x7FFF	Unused	
		0x8000	Status Interrupt	
		0x8001-0xFFFF	Unused	

## Register 6: CAN Test (CANTST), offset 0x014

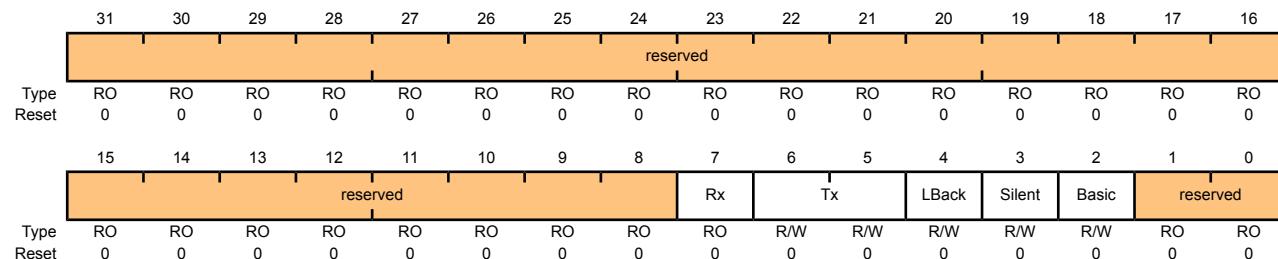
This is the test mode register for self-test and external pin access. It is write-enabled by the Test bit in the **CANCTL** register. Different test functions may be combined but when the Tx bit is not equal to 0x0, it disturbs message transmits.

### CAN Test (CANTST)

CAN0 base: 0x4004.0000

Offset 0x014

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	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.
7	Rx	RO	0	Receive Observation Displays the value on the CANnRx pin.
6:5	Tx	R/W	0x0	Transmit Control Overrides control of the CANnTx pin.
4	LBack	R/W	0	Loopback Mode 0: Disabled. 1: Enabled.
3	Silent	R/W	0	Silent Mode Do not transmit data; monitor the bus. Also known as Bus Monitor mode. 0: Disabled. 1: Enabled.
2	Basic	R/W	0	Basic Mode 0: Disabled. 1: Use CANIF1 registers as transmit buffer, and use CANIF2 registers as receive buffer.
1:0				reserved

---

Bit/Field	Name	Type	Reset	Description
1:0	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 7: CAN Baud Rate Prescalar Extension (CANBRPE), offset 0x018

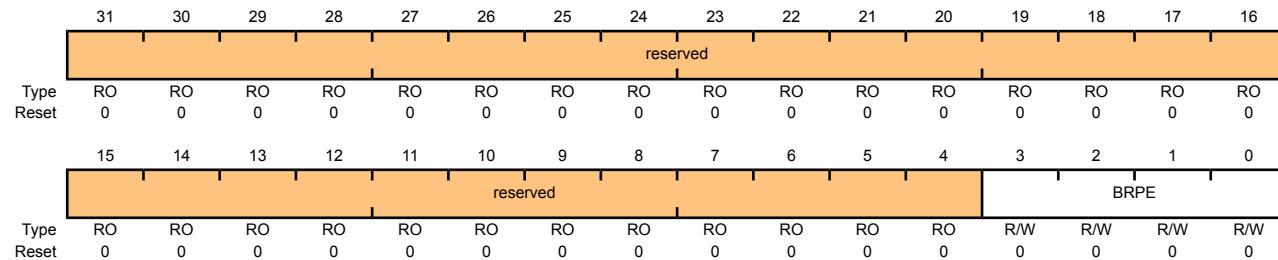
This register is used to further divide the bit time set with the **BRP** bit in the **CANBIT** register. It is write-enabled with the **CCE** bit in the **CANCTL** register.

### CAN Baud Rate Prescalar Extension (CANBRPE)

CAN0 base: 0x4004.0000

Offset 0x018

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	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.
3:0	BRPE	R/W	0x0	Baud Rate Prescalar Extension. 0x00-0x0F: Extend the <b>BRP</b> bit to values up to 1023. The actual interpretation by the hardware is one more than the value programmed by <b>BRPE</b> (MSBs) and <b>BRP</b> (LSBs) are used.

## Register 8: CAN IF1 Command Request (CANIF1CRQ), offset 0x020

## Register 9: CAN IF2 Command Request (CANIF2CRQ), offset 0x080

This register is used to start a transfer when its **MNUM** bit field is updated. Its **Busy** bit indicates that the information is transferring from the CAN Interface Registers to the internal message RAM.

A message transfer is started as soon as there is a write of the message object number with the **MNUM** bit. With this write operation, the **Busy** bit is automatically set to 1 to indicate that a transfer is in progress. After a wait time of 3 to 6 CAN\_CLK periods, the transfer between the interface register and the message RAM completes, which then sets the **Busy** bit back to 0.

### CAN IF1 Command Request (CANIF1CRQ)

CAN0 base: 0x4004.0000

Offset 0x020

Type RO, reset 0x0000.0001

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															MNUM
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	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	Busy	RO	0x0	Busy Flag 0: Reset when read/write action has finished. 1: Set when a write occurs to the message number in this register.
14:6	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	MNUM	R/W	0x01	Message Number Selects one of the 32 message objects in the message RAM for data transfer. The message objects are numbered from 1 to 32.
		Value	Description	
		0x00	0 is not a valid message number; it is interpreted as 0x20, or object 32.	
		0x01-0x20	Indicates specified message object 1 to 32.	
		0x21-0x3F	Not a valid message number; values are shifted and it is interpreted as 0x01-0x1F.	

**Register 10: CAN IF1 Command Mask (CANIF1CMSK), offset 0x024****Register 11: CAN IF2 Command Mask (CANIF2CMSK), offset 0x084**

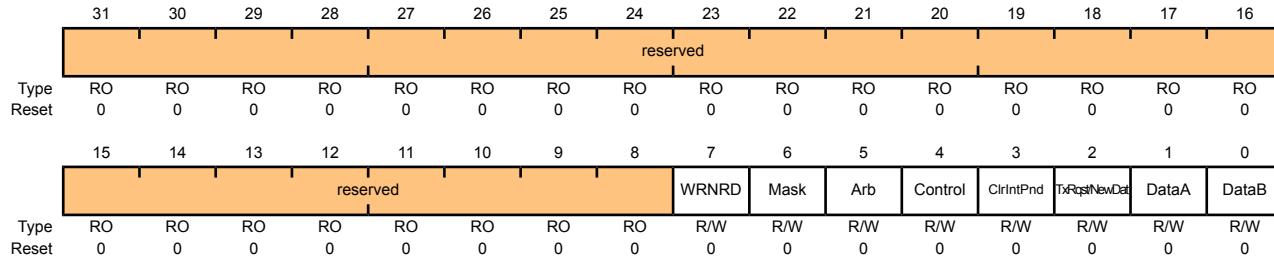
The Command Mask registers specify the transfer direction and select which buffer registers are the source or target of the data transfer.

## CAN IF1 Command Mask (CANIF1CMSK)

CAN0 base: 0x4004.0000

Offset 0x024

Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	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.
7	WRNRD	R/W	0	<p>Write, Not Read</p> <p>0: Read. Transfer the message object address specified by the <b>CAN Command Request (CANIFnCRQ)</b> register to the CAN message buffer registers (<b>CANIFnMSK1</b>, <b>CANIFnMSK2</b>, <b>CANIFnARB1</b>, <b>CANIFnARB2</b>, <b>CANIFnCTL</b>, <b>CANIFnDA1</b>, <b>CANIFnDA2</b>, <b>CANIFnDB1</b>, and <b>CANIFnDB2</b>).</p> <p>1: Write. Transfer data from the message buffer registers to the message object address specified by the <b>CANIFnCRQ</b> register.</p>
6	Mask	R/W	0x0	<p>Access Mask Bits</p> <p>When WRNRD=1 (writes):</p> <p>0: Mask bits unchanged.</p> <p>1: Transfer IDMask + Dir + MXtd to message object.</p> <p>When WRNRD=0 (reads):</p> <p>0: Mask bits unchanged.</p> <p>1: Transfer IDMask + Dir + MXtd of the message object into the Interface Registers.</p>
5	Arb	R/W	0x0	<p>Access Arbitration Bits</p> <p>When WRNRD=1 (writes):</p> <p>0: Arbitration bits unchanged.</p> <p>1: Transfer ID + Dir + Xtd + MsgVal to message object.</p> <p>When WRNRD=0 (reads):</p> <p>0: Arbitration bits unchanged.</p> <p>1: Transfer ID + Dir + Xtd + MsgVal to Message Buffer Register.</p>

Bit/Field	Name	Type	Reset	Description
4	Control	R/W	0x0	<p>Access Control Bits</p> <p>When WRNRD=1 (writes):</p> <p>0: Control bits unchanged.</p> <p>1: Transfer control bits to message object.</p> <p>When WRNRD=0 (reads):</p> <p>0: Control bits unchanged.</p> <p>1: Transfer control bits to Message Buffer Register.</p>
3	ClrlntPnd	R/W	0x0	<p>Clear Interrupt Pending Bit</p> <p><b>Note:</b> This bit is not used when in write (WRNRD=1).</p> <p>0: IntPnd bit in <b>CANIFnMCTL</b> register remains unchanged.</p> <p>1: Clear IntPnd bit in the <b>CANIFnMCTL</b> register in the message object.</p>
2	TxRqst/NewDat	R/W	0x0	<p>Access Transmission Request or New Data</p> <p>When WRNRD=1 (writes):</p> <p>Access Transmission Request Bit</p> <p>0: TxRqst bit unchanged.</p> <p>1: Set TxRqst bit</p> <p><b>Note:</b> If a transmission is requested by programming this TxRqst bit, the parallel TxRqst in the <b>CANIFnMCTL</b> register is ignored.</p> <p>When WRNRD=0 (reads):</p> <p>Access New Data Bit</p> <p>0: NewDat bit unchanged.</p> <p>1: Clear NewDat bit in the message object.</p> <p><b>Note:</b> A read access to a message object can be combined with the reset of the control bits IntPdn and NewDat. The values of these bits that are transferred to the <b>CANIFnMCTL</b> register always reflect the status before resetting these bits.</p>
1	DataA	R/W	0x0	<p>Access Data Byte 0 to 3</p> <p>When WRNRD=1 (writes):</p> <p>0: Data bytes 0-3 are unchanged.</p> <p>1: Transfer data bytes 0-3 (<b>CANIFnDA1</b> and <b>CANIFnDA2</b>) to message object.</p> <p>When WRNRD=0 (reads):</p> <p>0: Data bytes 0-3 are unchanged.</p> <p>1: Transfer data bytes 0-3 in message object to <b>CANIFnDA1</b> and <b>CANIFnDA2</b>.</p>

Bit/Field	Name	Type	Reset	Description
0	DataB	R/W	0x0	<p>Access Data Byte 4 to 7</p> <p>When WRNRD=1 (writes):</p> <p>0: Data bytes 4-7 unchanged.</p> <p>1: Transfer data bytes 4-7 (<b>CANIFnDB1</b> and <b>CANIFnDB2</b>) to message object.</p> <p>When WRNRD=0 (reads):</p> <p>0: Data bytes 4-7 unchanged.</p> <p>1: Transfer data bytes 4-7 in message object to <b>CANIFnDB1</b> and <b>CANIFnDB2</b>.</p>

**Register 12: CAN IF1 Mask 1 (CANIF1MSK1), offset 0x028****Register 13: CAN IF2 Mask 1 (CANIF2MSK1), offset 0x088**

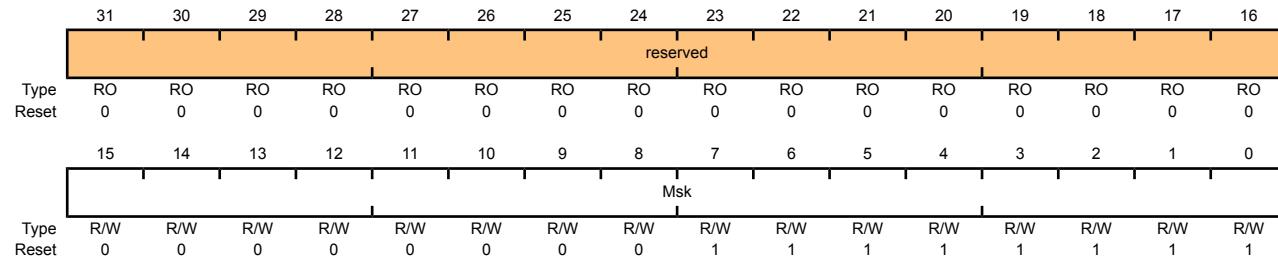
The mask information provided in this register accompanies the data (**CANIFnDAn**), arbitration information (**CANIFnARBn**), and control information (**CANIFnMCTL**) to the message object in the message RAM. The mask is used with the **ID** bit in the **CANIFnARBn** register for acceptance filtering. Additional mask information is contained in the **CANIFnMSK2** register.

**CAN IF1 Mask 1 (CANIF1MSK1)**

CAN0 base: 0x4004.0000

Offset 0x028

Type RO, reset 0x0000.FFFF



**Register 14: CAN IF1 Mask 2 (CANIF1MSK2), offset 0x02C****Register 15: CAN IF2 Mask 2 (CANIF2MSK2), offset 0x08C**

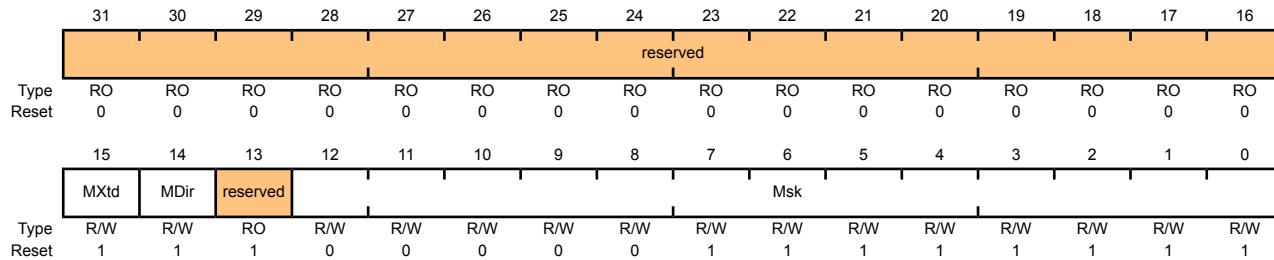
This register holds extended mask information that accompanies the **CANIFnMSK1** register.

**CAN IF1 Mask 2 (CANIF1MSK2)**

CAN0 base: 0x4004.0000

Offset 0x02C

Type RO, reset 0x0000.FFFF



**Register 16: CAN IF1 Arbitration 1 (CANIF1ARB1), offset 0x030****Register 17: CAN IF2 Arbitration 1 (CANIF2ARB1), offset 0x090**

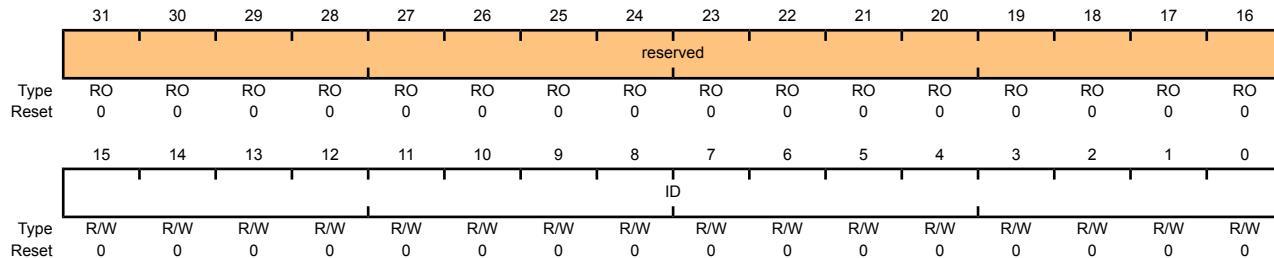
These registers hold the identifiers for acceptance filtering.

**CAN IF1 Arbitration 1 (CANIF1ARB1)**

CAN0 base: 0x4004.0000

Offset 0x030

Type RO, 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	ID	R/W	0x00	<p>Message Identifier</p> <p>This bit field is used with the <code>ID</code> field in the <b>CANIFnARB2</b> register to create the message identifier. <code>ID[28:0]</code> is the Extended Frame and <code>ID[28:18]</code> is the Standard Frame.</p>

**Register 18: CAN IF1 Arbitration 2 (CANIF1ARB2), offset 0x034****Register 19: CAN IF2 Arbitration 2 (CANIF2ARB2), offset 0x094**

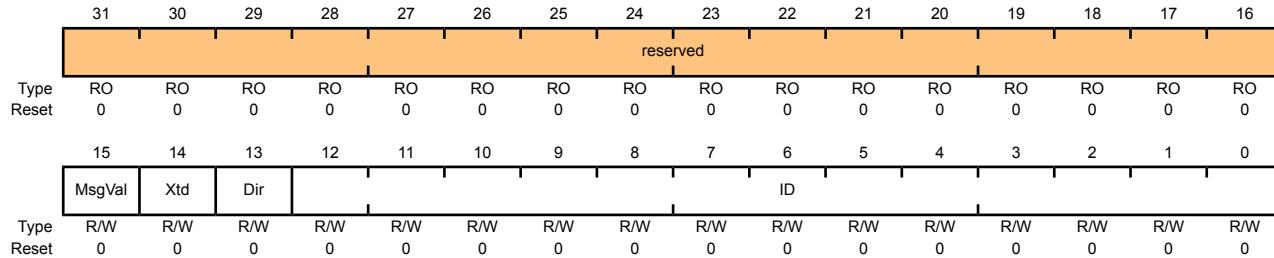
These registers hold information for acceptance filtering.

**CAN IF1 Arbitration 2 (CANIF1ARB2)**

CAN0 base: 0x4004.0000

Offset 0x034

Type RO, 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	MsgVal	R/W	0x0	<p><b>Message Valid</b></p> <p>0: The message object is ignored by the message handler.</p> <p>1: The message object is configured and will be considered by the message handler within the CAN controller.</p> <p>All unused message objects should have this bit cleared during initialization and before clearing the <b>Init</b> bit in the <b>CANCTL</b> register. The <b>MsgVal</b> bit must also be cleared before any of the following bits are modified or if the message object is no longer required: the <b>ID</b> bit fields in the <b>CANIFnARBn</b> registers, the <b>Xtd</b> and <b>Dir</b> bits in the <b>CANIFnARB2</b> register, or the <b>DLC</b> bits in the <b>CANIFnMCTL</b> register.</p>
14	Xtd	R/W	0x0	<p><b>Extended Identifier</b></p> <p>0: The 11-bit Standard Identifier will be used for this message object.</p> <p>1: The 29-bit Extended Identifier will be used for this message object.</p>
13	Dir	R/W	0x0	<p><b>Message Direction</b></p> <p>0: Receive. On <b>TxRqst</b>, a Remote Frame with the identifier of this message object is transmitted. On reception of a Data Frame with matching identifier, that message is stored in this message object.</p> <p>1: Transmit. On <b>TxRqst</b>, the respective message object is transmitted as a Data Frame. On reception of a Remote Frame with matching identifier, <b>TxRqst</b> bit of this message object is set (if <b>RmtEn=1</b>).</p>
12:0	ID	R/W	0x0	<p><b>Message Identifier</b></p> <p>Used with the <b>ID</b> bit in the <b>CANIFnARB1</b> register to create the message identifier. ID[28:0] is the Extended Frame and ID[28:18] is the Standard Frame.</p>

**Register 20: CAN IF1 Message Control (CANIF1MCTL), offset 0x038****Register 21: CAN IF2 Message Control (CANIF2MCTL), offset 0x098**

This register holds the control information associated with the message object to be sent to the Message RAM.

**CAN IF1 Message Control (CANIF1MCTL)**

CAN0 base: 0x4004.0000

Offset 0x038

Type RO, reset 0x0000.0000

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	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NewDat	MsgLst	IntPnd	UMask	Umsk	TxEIE	RxIE	RmtEn	TxRqst	EoB	reserved	reserved	DLC			
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	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	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	NewDat	R/W	0x0	<p>New Data</p> <p>0: No new data has been written into the data portion of this message object by the message handler since the last time this flag was cleared by the CPU.</p> <p>1: The message handler or the CPU has written new data into the data portion of this message object.</p>
14	MsgLst	R/W	0x0	<p>Message Lost</p> <p>0 : No message was lost since the last time this bit was reset by the CPU.</p> <p>1: The message handler stored a new message into this object when NewDat was set; the CPU has lost a message.</p> <p>This bit is only valid for message objects with the Dir bit in the <b>CANIFnARB2</b> register set to 0 (receive).</p>
13	IntPnd	R/W	0x0	<p>Interrupt Pending</p> <p>0: This message object is not the source of an interrupt.</p> <p>1: This message object is the source of an interrupt. The interrupt identifier in the <b>CAN Interrupt (CANINT)</b> register will point to this message object if there is not another interrupt source with a higher priority.</p>
12	UMask	R/W	0x0	<p>Use Acceptance Mask</p> <p>0: Mask ignored.</p> <p>1: Use mask (Msk, MXtd, and MDix) for acceptance filtering.</p>

Bit/Field	Name	Type	Reset	Description						
11	TxIE	R/W	0x0	<p>Transmit Interrupt Enable</p> <p>0: The <code>IntPnd</code> bit in the <b>CANIFnMCTL</b> register is unchanged after a successful transmission of a frame.</p> <p>1: The <code>IntPnd</code> bit in the <b>CANIFnMCTL</b> register is set after a successful transmission of a frame.</p>						
10	RxIE	R/W	0x0	<p>Receive Interrupt Enable</p> <p>0: The <code>IntPnd</code> bit in the <b>CANIFnMCTL</b> register is unchanged after a successful reception of a frame.</p> <p>1: The <code>IntPnd</code> bit in the <b>CANIFnMCTL</b> register is set after a successful reception of a frame.</p>						
9	RmtEn	R/W	0x0	<p>Remote Enable</p> <p>0: At the reception of a Remote Frame, the <code>TxRqst</code> bit in the <b>CANIFnMCTL</b> register is left unchanged.</p> <p>1: At the reception of a Remote Frame, the <code>TxRqst</code> bit in the <b>CANIFnMCTL</b> register is set.</p>						
8	TxRqst	R/W	0x0	<p>Transmit Request</p> <p>0: This message object is not waiting for transmission.</p> <p>1: The transmission of this message object is requested and is not yet done.</p>						
7	EoB	R/W	0x0	<p>End of Buffer</p> <p>0: Message object belongs to a FIFO Buffer and is not the last message object of that FIFO Buffer.</p> <p>1: Single message object or last message object of a FIFO Buffer.</p> <p>This bit is used to concatenate two or more message objects (up to 32) to build a FIFO buffer. For a single message object (thus not belonging to a FIFO buffer), this bit must be set to 1.</p>						
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.						
3:0	DLC	R/W	0x0	<p>Data Length Code</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0-0x8</td> <td>Specifies the number of bytes in the Data Frame.</td> </tr> <tr> <td>0x9-0xF</td> <td>Defaults to a Data Frame with 8 bytes.</td> </tr> </tbody> </table> <p>The <code>DLC</code> bit in the <b>CANIFnMCTL</b> register of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it writes <code>DLC</code> to the value given by the received message.</p>	Value	Description	0x0-0x8	Specifies the number of bytes in the Data Frame.	0x9-0xF	Defaults to a Data Frame with 8 bytes.
Value	Description									
0x0-0x8	Specifies the number of bytes in the Data Frame.									
0x9-0xF	Defaults to a Data Frame with 8 bytes.									

**Register 22: CAN IF1 Data A1 (CANIF1DA1), offset 0x03C****Register 23: CAN IF1 Data A2 (CANIF1DA2), offset 0x040****Register 24: CAN IF1 Data B1 (CANIF1DB1), offset 0x044****Register 25: CAN IF1 Data B2 (CANIF1DB2), offset 0x048****Register 26: CAN IF2 Data A1 (CANIF2DA1), offset 0x09C****Register 27: CAN IF2 Data A2 (CANIF2DA2), offset 0x0A0****Register 28: CAN IF2 Data B1 (CANIF2DB1), offset 0x0A4****Register 29: CAN IF2 Data B2 (CANIF2DB2), offset 0x0A8**

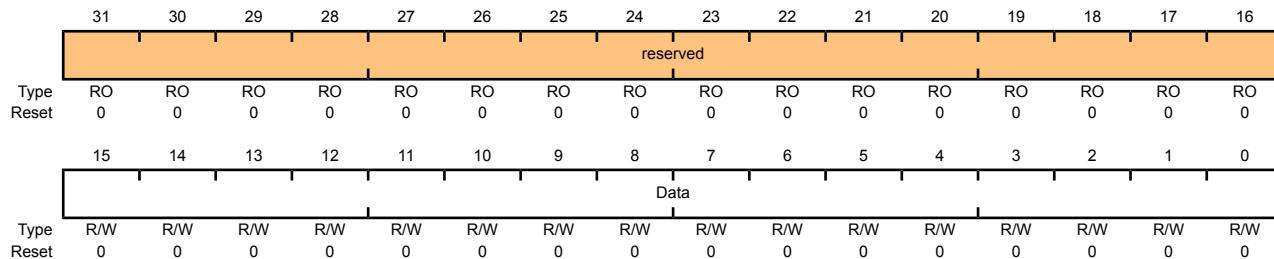
These registers contain the data to be sent or that has been received. In a CAN Data Frame, data byte 0 is the first byte to be transmitted or received and data byte 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte is transmitted first.

## CAN IF1 Data A1 (CANIF1DA1)

CAN0 base: 0x4004.0000

Offset 0x03C

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	0x00	The <b>CANIFnDA1</b> registers contain data bytes 1 and 0; <b>CANIFnDA2</b> data bytes 3 and 2; <b>CANIFnDB1</b> data bytes 5 and 4; and <b>CANIFnDB2</b> data bytes 7 and 6.

**Register 30: CAN Transmission Request 1 (CANTXRQ1), offset 0x100****Register 31: CAN Transmission Request 2 (CANTXRQ2), offset 0x104**

The **CANTXRQ1** and **CANTXRQ2** registers hold the TxRqst bits of the 32 message objects. By reading out these bits, the CPU can check which message object has a transmission request pending. The TxRqst bit of a specific message object can be changed by three sources: (1) the CPU via the **CAN IFn Message Control (CANIFnMCTL)** register, (2) the message handler state machine after the reception of a Remote Frame, or (3) the message handler state machine after a successful transmission.

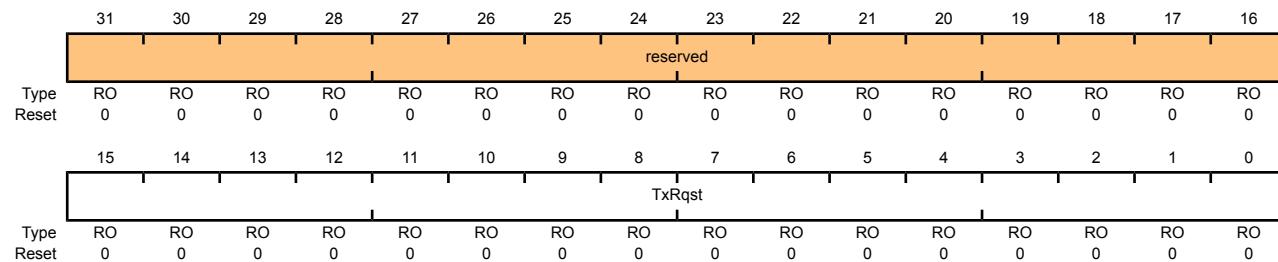
The **CANTXRQ1** register contains the TxRqst bit of the first 16 message objects in the message RAM; the **CANTXRQ2** register contains the TxRqst bit of the second 16 message objects.

## CAN Transmission Request 1 (CANTXRQ1)

CAN0 base: 0x4004.0000

Offset 0x100

Type RO, 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	TxRqst	RO	0x00	Transmission Request Bits (of all message objects) 0: The message object is not waiting for transmission. 1: The transmission of the message object is requested and is not yet done.

## Register 32: CAN New Data 1 (CANNWDA1), offset 0x120

## Register 33: CAN New Data 2 (CANNWDA2), offset 0x124

The **CANNWDA1** and **CANNWDA2** registers hold the **NewDat** bits of the 32 message objects. By reading these bits, the CPU can check which message object has its data portion updated. The **NewDat** bit of a specific message object can be changed by three sources: (1) the CPU via the **CAN IFn Message Control (CANIFnMCTL)** register, (2) the message handler state machine after the reception of a Data Frame, or (3) the message handler state machine after a successful transmission.

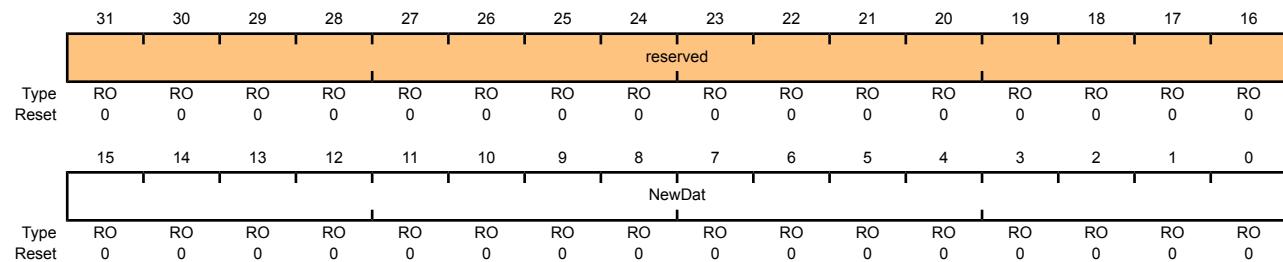
The **CANNWDA1** register contains the **NewDat** bit of the first 16 message objects in the message RAM; the **CANNWDA2** register contains the **NewDat** bit of the second 16 message objects.

### CAN New Data 1 (CANNWDA1)

CAN0 base: 0x4004.0000

Offset 0x120

Type RO, 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	NewDat	RO	0x00	<p>New Data Bits (of all message objects)</p> <p>0: No new data has been written into the data portion of this message object by the message handler since the last time this flag was cleared by the CPU.</p> <p>1: The message handler or the CPU has written new data into the data portion of this message object.</p>

**Register 34: CAN Message 1 Interrupt Pending (CANMSG1INT), offset 0x140****Register 35: CAN Message 2 Interrupt Pending (CANMSG2INT), offset 0x144**

The **CANMSG1INT** and **CANMSG2INT** registers hold the `IntPnd` bits of the 32 message objects. By reading these bits, the CPU can check which message object has an interrupt pending. The `IntPnd` bit of a specific message object can be changed through two sources: (1) the CPU via the **CAN IFn Message Control (CANIFnMCTL)** register, or (2) the message handler state machine after the reception or transmission of a frame.

This field is also encoded in the **CAN Interrupt (CANINT)** register.

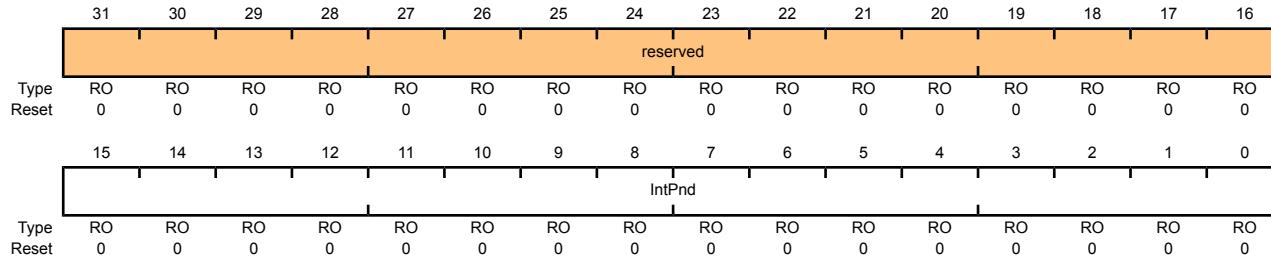
The **CANMSG1INT** register contains the `IntPnd` bit of the first 16 message objects in the message RAM; the **CANMSG2INT** register contains the `IntPnd` bit of the second 16 message objects.

## CAN Message 1 Interrupt Pending (CANMSG1INT)

CAN0 base: 0x4004.0000

Offset 0x140

Type RO, 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	IntPnd	RO	0x00	Interrupt Pending Bits (of all message objects) 0: This message object is not the source of an interrupt. 1: This message object is the source of an interrupt.

## Register 36: CAN Message 1 Valid (CANMSG1VAL), offset 0x160

## Register 37: CAN Message 2 Valid (CANMSG2VAL), offset 0x164

The **CANMSG1VAL** and **CANMSG2VAL** registers hold the `MsgVal` bits of the 32 message objects. By reading these bits, the CPU can check which message object is valid. The message value of a specific message object can be changed with the **CAN IFn Message Control (CANIFnMCTL)** register.

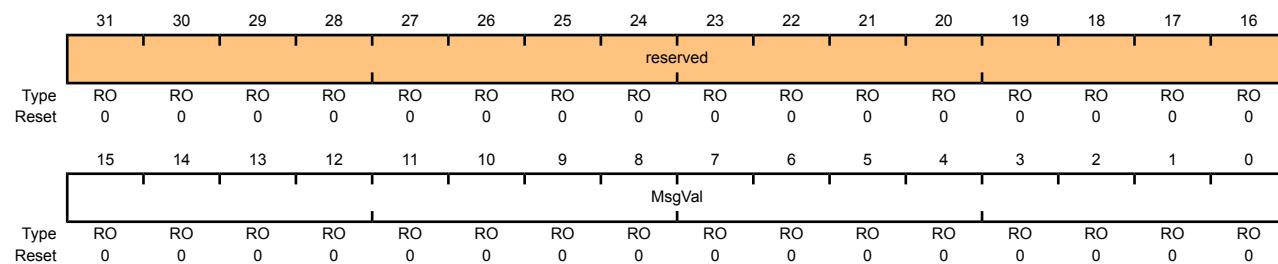
The **CANMSG1VAL** register contains the `MsgVal` bit of the first 16 message objects in the message RAM; the **CANMSG2VAL** register contains the `MsgVal` bit of the second 16 message objects in the message RAM.

### CAN Message 1 Valid (CANMSG1VAL)

CANO base: 0x4004.0000

Offset 0x160

Type RO, 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	MsgVal	RO	0x00	Message Valid Bits (of all message objects) 0: This message object is not configured and is ignored by the message handler. 1: This message object is configured and should be considered by the message handler.

## 17 Ethernet Controller

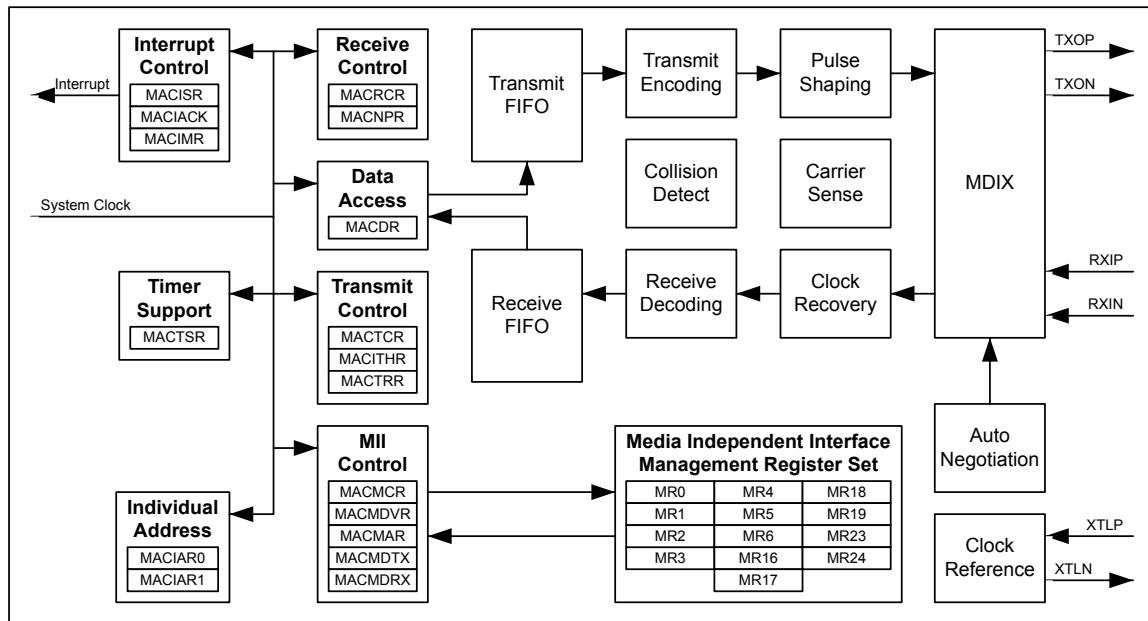
The Stellaris® Ethernet Controller consists of a fully integrated media access controller (MAC) and network physical (PHY) interface device. The Ethernet Controller conforms to *IEEE 802.3* specifications and fully supports 10BASE-T and 100BASE-TX standards.

The Ethernet Controller module has the following features:

- Conforms to the *IEEE 802.3-2002 specification*
  - 10BASE-T/100BASE-TX IEEE-802.3 compliant. Requires only a dual 1:1 isolation transformer interface to the line
  - 10BASE-T/100BASE-TX ENDEC, 100BASE-TX scrambler/descrambler
  - Full-featured auto-negotiation
- Multiple operational modes
  - Full- and half-duplex 100 Mbps
  - Full- and half-duplex 10 Mbps
  - Power-saving and power-down modes
- Highly configurable
  - Programmable MAC address
  - LED activity selection
  - Promiscuous mode support
  - CRC error-rejection control
  - User-configurable interrupts
- Physical media manipulation
  - Automatic MDI/MDI-X cross-over correction
  - Register-programmable transmit amplitude
  - Automatic polarity correction and 10BASE-T signal reception

## 17.1 Block Diagram

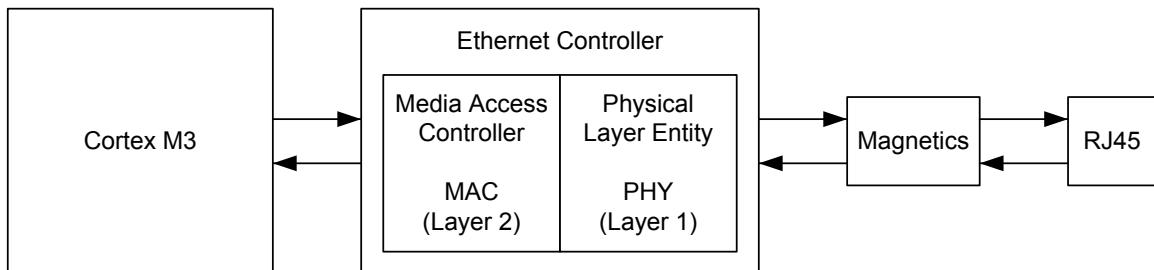
Figure 17-1. Ethernet Controller Block Diagram



## 17.2 Functional Description

As shown in Figure 17-2 on page 453, the Ethernet Controller is functionally divided into two layers or modules: the Media Access Controller (MAC) layer and the Network Physical (PHY) layer. These correspond to the OSI model layers 2 and 1. The primary interface to the Ethernet Controller is a simple bus interface to the MAC layer. The MAC layer provides transmit and receive processing for Ethernet frames. The MAC layer also provides the interface to the PHY module via an internal Media Independent Interface (MII).

Figure 17-2. Ethernet Controller



### 17.2.1 Internal MII Operation

For the MII management interface to function properly, the MDIO signal must be connected through a  $10\text{ k}\Omega$  pull-up resistor to the +3.3 V supply. Failure to connect this pull-up resistor will prevent management transactions on this internal MII to function. Note that it is possible for data transmission across the MII to still function since the PHY layer will auto-negotiate the link parameters by default.

For the MII management interface to function properly, the internal clock must be divided down from the system clock to a frequency no greater than 2.5 MHz. The **MACMDV** register contains the divider used for scaling down the system clock. See page 473 for more details about the use of this register.

## 17.2.2 PHY Configuration/Operation

The Physical Layer (PHY) in the Ethernet Controller includes integrated ENDECs, scrambler/descrambler, dual-speed clock recovery, and full-featured auto-negotiation functions. The transmitter includes an on-chip pulse shaper and a low-power line driver. The receiver has an adaptive equalizer and a baseline restoration circuit required for accurate clock and data recovery. The transceiver interfaces to Category-5 unshielded twisted pair (Cat-5 UTP) cabling for 100BASE-TX applications, and Category-3 unshielded twisted pair (Cat-3 UTP) for 10BASE-T applications. The Ethernet Controller is connected to the line media via dual 1:1 isolation transformers. No external filter is required.

### 17.2.2.1 Clock Selection

The PHY has an on-chip crystal oscillator which can also be driven by an external oscillator. In this mode of operation, a 25-MHz crystal should be connected between the **XTALPPHY** and **XTALNPHY** pins. Alternatively, an external 25-MHz clock input can be connected to the **XTALPPHY** pin. In this mode of operation, a crystal is not required and the **XTALNPHY** pin must be tied to ground.

### 17.2.2.2 Auto-Negotiation

The PHY supports the auto-negotiation functions of Clause 28 of the *IEEE 802.3* standard for 10/100 Mbps operation over copper wiring. This function can be enabled via register settings. The auto-negotiation function defaults to On and the **ANEGEN** bit in the **MR0** register is High after reset. Software can disable the auto-negotiation function by writing to the **ANEGEN** bit. The contents of the **MR4** register are sent to the PHY's link partner during auto-negotiation via fast-link pulse coding.

Once auto-negotiation is complete, the **DPLX** and **RATE** bits in the **MR18** register reflect the actual speed and duplex that was chosen. If auto-negotiation fails to establish a link for any reason, the **ANEGF** bit in the **MR18** register reflects this and auto-negotiation restarts from the beginning. Writing a 1 to the **RANEG** bit in the **MR0** register also causes auto-negotiation to restart.

### 17.2.2.3 Polarity Correction

The PHY is capable of either automatic or manual polarity reversal for 10BASE-T and auto-negotiation functions. Bits 4 and 5 (**RVSPOL** and **APOL**) in the **MR16** register control this feature. The default is automatic mode, where **APOL** is Low and **RVSPOL** indicates if the detection circuitry has inverted the input signal. To enter manual mode, **APOL** should be set High and **RVSPOL** then controls the signal polarity.

### 17.2.2.4 MDI/MDI-X Configuration

The PHY supports the automatic MDI/MDI-X configuration as defined in *IEEE 802.3-2002 specification*. This eliminates the need for cross-over cables when connecting to another device, such as a hub. The algorithm is controlled via settings in the **MR24** register. Refer to page 496 for additional details about these settings.

### 17.2.2.5 LED Indicators

The PHY supports two LED signals that can be used to indicate various states of operation of the Ethernet Controller. These signals are mapped to the **LED0** and **LED1** pins. By default, these pins are configured as GPIO signals (**PF3** and **PF2**). For the PHY layer to drive these signals, they must be reconfigured to their hardware function. See "General-Purpose Input/Outputs (GPIOs)" on page

165 for additional details. The function of these pins is programmable via the PHY layer **MR23** register. Refer to page 495 for additional details on how to program these LED functions.

### 17.2.3 MAC Configuration/Operation

#### 17.2.3.1 Ethernet Frame Format

Ethernet data is carried by Ethernet frames. The basic frame format is shown in Figure 17-3 on page 455.

**Figure 17-3. Ethernet Frame**

Preamble	SFD	Destination Address	Source Address	Length/ Type	Data	FCS
7 Bytes	1 Byte	6 Bytes	6 Bytes	2 Bytes	46 - 1500 Bytes	4 Bytes

The seven fields of the frame are transmitted from left to right. The bits within the frame are transmitted from least to most significant bit.

- Preamble

The Preamble field is used by the physical layer signaling circuitry to synchronize with the received frame's timing. The preamble is 7 octets long.

- Start Frame Delimiter (SFD)

The SFD field follows the preamble pattern and indicates the start of the frame. Its value is 1010.1011.

- Destination Address (DA)

This field specifies destination addresses for which the frame is intended. The LSB of the DA determines whether the address is an individual (0), or group/multicast (1) address.

- Source Address (SA)

The source address field identifies the station from which the frame was initiated.

- Length/Type Field

The meaning of this field depends on its numeric value. The first of two octets is most significant. This field can be interpreted as length or type code. The maximum length of the data field is 1500 octets. If the value of the Length/Type field is less than or equal to 1500 decimal, it indicates the number of MAC client data octets. If the value of this field is greater than or equal to 1536 decimal, then it is type interpretation. The meaning of the Length/Type field when the value is between 1500 and 1536 decimal is unspecified by the standard. The MAC module assumes type interpretation if the value of the Length/Type field is greater than 1500 decimal.

- Data

The data field is a sequence of 0 to 1500 octets. Full data transparency is provided so any values can appear in this field. A minimum frame size is required to properly meet the IEEE standard. If necessary, the data field is extended by appending extra bits (a pad). The pad field can have a size of 0 to 46 octets. The sum of the data and pad lengths must be a minimum of 46 octets. The MAC module automatically inserts pads if required, though it can be disabled by a register

write. For the MAC module core, data sent/received can be larger than 1500 bytes, and no Frame Too Long error is reported. Instead, a FIFO Overrun error is reported when the frame received is too large to fit into the Ethernet Controller's RAM.

- **Frame Check Sequence (FCS)**

The frame check sequence carries the cyclic redundancy check (CRC) value. The value of this field is computed over destination address, source address, length/type, data, and pad fields using the CRC-32 algorithm. The MAC module computes the FCS value one nibble at a time. For transmitted frames, this field is automatically inserted by the MAC layer, unless disabled by the **CRC** bit in the **MACTCTL** register. For received frames, this field is automatically checked. If the FCS does not pass, the frame will not be placed in the RX FIFO, unless the FCS check is disabled by the **BADCRC** bit in the **MACRCTL** register.

### 17.2.3.2 MAC Layer FIFOs

For Ethernet frame transmission, a 2 KB TX FIFO is provided that can be used to store a single frame. While the *IEEE 802.3 specification* limits the size of an Ethernet frame's payload section to 1500 Bytes, the Ethernet Controller places no such limit. The full buffer can be used, for a payload of up to 2032 bytes.

For Ethernet frame reception, a 2-KB RX FIFO is provided that can be used to store multiple frames, up to a maximum of 31 frames. If a frame is received and there is insufficient space in the RX FIFO, an overflow error will be indicated.

For details regarding the TX and RX FIFO layout, refer to Table 17-1 on page 456. Please note the following difference between TX and RX FIFO layout. For the TX FIFO, the Data Length field in the first FIFO word refers to the Ethernet frame data payload, as shown in the 5th to nth FIFO positions. For the RX FIFO, the Frame Length field is the total length of the received Ethernet frame, including the FCS and Frame Length bytes. Also note that if FCS generation is disabled with the **CRC** bit in the **MACTCTL** register, the last word in the FIFO must be the FCS bytes for the frame that has been written to the FIFO.

Also note that if the length of the data payload section is not a multiple of 4, the FCS field will overlap words in the FIFO. However, for the RX FIFO, the beginning of the next frame will always be on a word boundary.

**Table 17-1. TX & RX FIFO Organization**

FIFO Word Read/Write Sequence	Word Bit Fields	TX FIFO (Write)	RX FIFO (Read)
1st	7:0	Data Length LSB	Frame Length LSB
	15:8	Data Length MSB	Frame Length MSB
	23:16		DA oct 1
	31:24		DA oct 2
2nd	7:0		DA oct 3
	15:8		DA oct 4
	23:16		DA oct 5
	31:24		DA oct 6
3rd	7:0		SA oct 1
	15:8		SA oct 2
	23:16		SA oct 3
	31:24		SA oct 4

FIFO Word Read/Write Sequence	Word Bit Fields	TX FIFO (Write)	RX FIFO (Read)
4th	7:0	SA oct 5	
	15:8	SA oct 6	
	23:16	Len/Type MSB	
	31:24	Len/Type LSB	
5th to nth	7:0	data oct n	
	15:8	data oct n+1	
	23:16	data oct n+2	
	31:24	data oct n+3	
last	7:0	FCS 1 (if the CRC bit in <b>MACCTL</b> is 0)	FCS 1
	15:8	FCS 2 (if the CRC bit in <b>MACCTL</b> is 0)	FCS 2
	23:16	FCS 3 (if the CRC bit in <b>MACCTL</b> is 0)	FCS 3
	31:24	FCS 4 (if the CRC bit in <b>MACCTL</b> is 0)	FCS 4

### 17.2.3.3 Ethernet Transmission Options

The Ethernet Controller can automatically generate and insert the Frame Check Sequence (FCS) at the end of the transmit frame. This is controlled by the **CRC** bit in the **MACTCTL** register. For test purposes, in order to generate a frame with an invalid CRC, this feature can be disabled.

The *IEEE 802.3 specification* requires that the Ethernet frame payload section be a minimum of 46 bytes. The Ethernet Controller can be configured to automatically pad the data section if the payload data section loaded into the FIFO is less than the minimum 46 bytes. This feature is controlled by the **PADEN** bit in the **MACTCTL** register.

At the MAC layer, the transmitter can be configured for both full-duplex and half-duplex operation by using the **DUPLEX** bit in the **MACTCTL** register.

### 17.2.3.4 Ethernet Reception Options

Using the **BADCRC** bit in the **MACRCTL** register, the Ethernet Controller can be configured to reject incoming Ethernet frames with an invalid FCS field.

The Ethernet receiver can also be configured for Promiscuous and Multicast modes using the **PRMS** and **AMUL** fields in the **MACRCTL** register. If these modes are not enabled, only Ethernet frames with a broadcast address, or frames matching the MAC address programmed into the **MACIA0** and **MACIA1** register will be placed into the RX FIFO.

### 17.2.3.5 Packet Timestamps

Using the **TSEN** bit in the **MACTS** register, the MAC transmit and receive interrupts can be used to trigger edge capture events on General-Purpose Timer 3. The transmit interrupt is routed to the CCP (even) input of General-Purpose Timer 3, while the receive interrupt is routed to the CCP (odd) input of General-Purpose Timer 3. This timer can then be configured in 16-bit edge capture mode and be used with a third 16-bit free-running timer to capture a more accurate timestamp for the transmit or receive packet. This feature can be used with a protocol such as IEEE-1588 to provide more accurate timestamps of the synchronization packets, improving the overall accuracy of the protocol.

#### 17.2.4 Interrupts

The Ethernet Controller can generate an interrupt for one or more of the following conditions:

- A frame has been received into an empty RX FIFO
- A frame transmission error has occurred
- A frame has been transmitted successfully
- A frame has been received with no room in the RX FIFO (overrun)
- A frame has been received with one or more error conditions (for example, FCS failed)
- An MII management transaction between the MAC and PHY layers has completed
- One or more of the following PHY layer conditions occurs:
  - Auto-Negotiate Complete
  - Remote Fault
  - Link Status Change
  - Link Partner Acknowledge
  - Parallel Detect Fault
  - Page Received
  - Receive Error
  - Jabber Event Detected

### 17.3 Initialization and Configuration

To use the Ethernet Controller, the peripheral must be enabled by setting the `EPHY0` and `EMAC0` bits in the **RCGC2** register. The following steps can then be used to configure the Ethernet Controller for basic operation.

1. Program the **MACDIV** register to obtain a 2.5 MHz clock (or less) on the internal MII. Assuming a 20-MHz system clock, the **MACDIV** value would be 4.
2. Program the **MACIA0** and **MACIA1** register for address filtering.
3. Program the **MACTCTL** register for Auto CRC generation, padding, and full-duplex operation using a value of 0x16.
4. Program the **MACRCTL** register to reject frames with bad FCS using a value of 0x08.
5. Enable both the Transmitter and Receive by setting the LSB in both the **MACTCTL** and **MACRCTL** registers.
6. To transmit a frame, write the frame into the TX FIFO using the **MACDATA** register. Then set the `NEWTX` bit in the **MACTR** register to initiate the transmit process. When the `NEWTX` bit has been cleared, the TX FIFO will be available for the next transmit frame.

7. To receive a frame, wait for the **NPR** field in the **MACNP** register to be non-zero. Then begin reading the frame from the RX FIFO by using the **MACDATA** register. When the frame (including the FCS field) has been read, the **NPR** field should decrement by one. When there are no more frames in the RX FIFO, the **NPR** field will read 0.

## 17.4 Ethernet Register Map

Table 17-2 on page 459 lists the Ethernet MAC registers. All addresses given are relative to the Ethernet MAC base address of 0x4004.8000.

The *IEEE 802.3* standard specifies a register set for controlling and gathering status from the PHY. The registers are collectively known as the MII Management registers and are detailed in Section 22.2.4 of the *IEEE 802.3 specification*. Table 17-2 on page 459 also lists these MII Management registers. *All addresses given are absolute and are written directly to the REGADR field of the MACMCTL register*. The format of registers 0 to 15 are defined by the IEEE specification and are common to all PHY implementations. The only variance allowed is for features that may or may not be supported by a specific PHY. Registers 16 to 31 are vendor-specific registers, used to support features that are specific to a vendors PHY implementation. Vendor-specific registers not listed are reserved.

**Table 17-2. Ethernet Register Map**

Offset	Name	Type	Reset	Description	See page
<b>Ethernet MAC</b>					
0x000	MACRIS	RO	0x0000.0000	Ethernet MAC Raw Interrupt Status	461
0x000	MACIACK	W1C	0x0000.0000	Ethernet MAC Interrupt Acknowledge	463
0x004	MACIM	R/W	0x0000.007F	Ethernet MAC Interrupt Mask	464
0x008	MACRCTL	R/W	0x0000.0008	Ethernet MAC Receive Control	465
0x00C	MACTCTL	R/W	0x0000.0000	Ethernet MAC Transmit Control	466
0x010	MACDATA	R/W	0x0000.0000	Ethernet MAC Data	467
0x014	MACIA0	R/W	0x0000.0000	Ethernet MAC Individual Address 0	469
0x018	MACIA1	R/W	0x0000.0000	Ethernet MAC Individual Address 1	470
0x01C	MACTHR	R/W	0x0000.003F	Ethernet MAC Threshold	471
0x020	MACMCTL	R/W	0x0000.0000	Ethernet MAC Management Control	472
0x024	MACMDV	R/W	0x0000.0080	Ethernet MAC Management Divider	473
0x02C	MACMTXD	R/W	0x0000.0000	Ethernet MAC Management Transmit Data	474
0x030	MACMRXD	R/W	0x0000.0000	Ethernet MAC Management Receive Data	475
0x034	MACNP	RO	0x0000.0000	Ethernet MAC Number of Packets	476
0x038	MACTR	R/W	0x0000.0000	Ethernet MAC Transmission Request	477
0x03C	MACTS	R/W	0x0000.0000	Ethernet MAC Timer Support	478
<b>MII Management</b>					
-	MR0	R/W	0x3100	Ethernet PHY Management Register 0 – Control	479

Offset	Name	Type	Reset	Description	See page
-	MR1	RO	0x7849	Ethernet PHY Management Register 1 – Status	481
-	MR2	RO	0x000E	Ethernet PHY Management Register 2 – PHY Identifier 1	483
-	MR3	RO	0x7237	Ethernet PHY Management Register 3 – PHY Identifier 2	484
-	MR4	R/W	0x01E1	Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement	485
-	MR5	RO	0x0000	Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability	487
-	MR6	RO	0x0000	Ethernet PHY Management Register 6 – Auto-Negotiation Expansion	488
-	MR16	R/W	0x0140	Ethernet PHY Management Register 16 – Vendor-Specific	489
-	MR17	R/W	0x0000	Ethernet PHY Management Register 17 – Interrupt Control/Status	491
-	MR18	RO	0x0000	Ethernet PHY Management Register 18 – Diagnostic	493
-	MR19	R/W	0x4000	Ethernet PHY Management Register 19 – Transceiver Control	494
-	MR23	R/W	0x0010	Ethernet PHY Management Register 23 – LED Configuration	495
-	MR24	R/W	0x00C0	Ethernet PHY Management Register 24 –MDI/MDIX Control	496

## 17.5 Ethernet MAC Register Descriptions

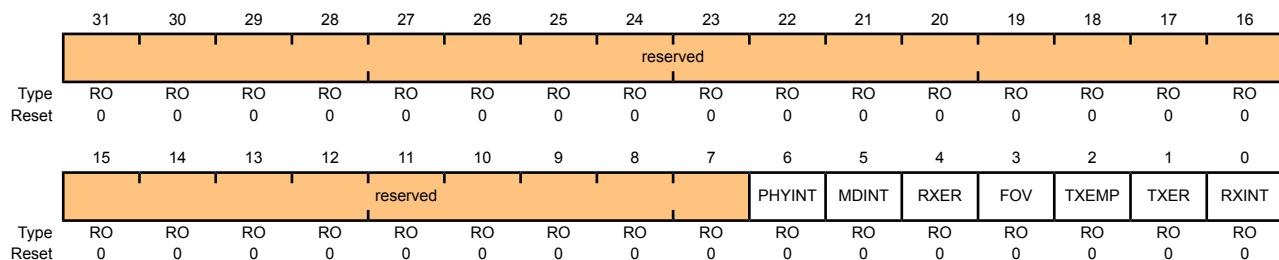
The remainder of this section lists and describes the Ethernet MAC registers, in numerical order by address offset. Also see “MII Management Register Descriptions” on page 478.

## Register 1: Ethernet MAC Raw Interrupt Status (MACRIS), offset 0x000

The **MACRIS** register is the interrupt status register. On a read, this register gives the current status value of the corresponding interrupt prior to masking.

### Ethernet MAC Raw Interrupt Status (MACRIS)

Base 0x4004.8000  
Offset 0x000  
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PHYINT	RO	0x0	PHY Interrupt  When set, indicates that an enabled interrupt in the PHY layer has occurred. <b>MR17</b> in the PHY must be read to determine the specific PHY event that triggered this interrupt.
5	MDINT	RO	0x0	MII Transaction Complete  When set, indicates that a transaction (read or write) on the MII interface has completed successfully.
4	RXER	RO	0x0	Receive Error  This bit indicates that an error was encountered on the receiver. The possible errors that can cause this interrupt bit to be set are: <ul style="list-style-type: none"> <li>■ A receive error occurs during the reception of a frame (100 Mb/s only).</li> <li>■ The frame is not an integer number of bytes (dribble bits) due to an alignment error.</li> <li>■ The CRC of the frame does not pass the FCS check.</li> <li>■ The length/type field is inconsistent with the frame data size when interpreted as a length field.</li> </ul>
3	FOV	RO	0x0	FIFO Overrun  When set, indicates that an overrun was encountered on the receive FIFO.
2	TXEMP	RO	0x0	Transmit FIFO Empty  When set, indicates that the packet was transmitted and that the TX FIFO is empty.

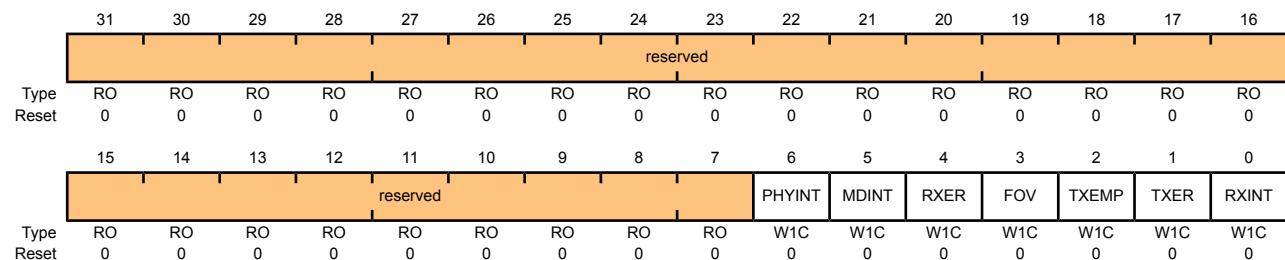
Bit/Field	Name	Type	Reset	Description
1	TXER	RO	0x0	<p>Transmit Error</p> <p>When set, indicates that an error was encountered on the transmitter. The possible errors that can cause this interrupt bit to be set are:</p> <ul style="list-style-type: none"><li>■ The data length field stored in the TX FIFO exceeds 2032. The frame is not sent when this error occurs.</li><li>■ The retransmission attempts during the backoff process have exceeded the maximum limit of 16.</li></ul>
0	RXINT	RO	0x0	<p>Packet Received</p> <p>When set, indicates that at least one packet has been received and is stored in the receiver FIFO.</p>

## Register 2: Ethernet MAC Interrupt Acknowledge (MACIACK), offset 0x000

A write of a 1 to any bit position of this register clears the corresponding interrupt bit in the **Ethernet MAC Raw Interrupt Status (MACRIS)** register.

### Ethernet MAC Interrupt Acknowledge (MACIACK)

Base 0x4004.8000  
Offset 0x000  
Type W1C, reset 0x0000.0000



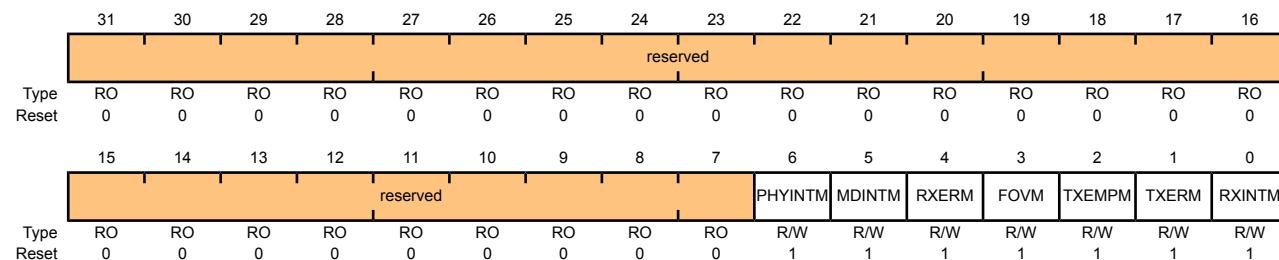
Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PHYINT	W1C	0x0	Clear PHY Interrupt  A write of a 1 clears the PHYINT interrupt read from the <b>MACRIS</b> register.
5	MDINT	W1C	0x0	Clear MII Transaction Complete  A write of a 1 clears the MDINT interrupt read from the <b>MACRIS</b> register.
4	RXER	W1C	0x0	Clear Receive Error  A write of a 1 clears the RXER interrupt read from the <b>MACRIS</b> register.
3	FOV	W1C	0x0	Clear FIFO Overrun  A write of a 1 clears the FOV interrupt read from the <b>MACRIS</b> register.
2	TXEMP	W1C	0x0	Clear Transmit FIFO Empty  A write of a 1 clears the TXEMP interrupt read from the <b>MACRIS</b> register.
1	TXER	W1C	0x0	Clear Transmit Error  A write of a 1 clears the TXER interrupt read from the <b>MACRIS</b> register and resets the TX FIFO write pointer.
0	RXINT	W1C	0x0	Clear Packet Received  A write of a 1 clears the RXINT interrupt read from the <b>MACRIS</b> register.

### Register 3: Ethernet MAC Interrupt Mask (MACIM), offset 0x004

This register allows software to enable/disable Ethernet MAC interrupts. Writing a 0 disables the interrupt, while writing a 1 enables it.

#### Ethernet MAC Interrupt Mask (MACIM)

Base 0x4004.8000  
Offset 0x004  
Type R/W, reset 0x0000.007F



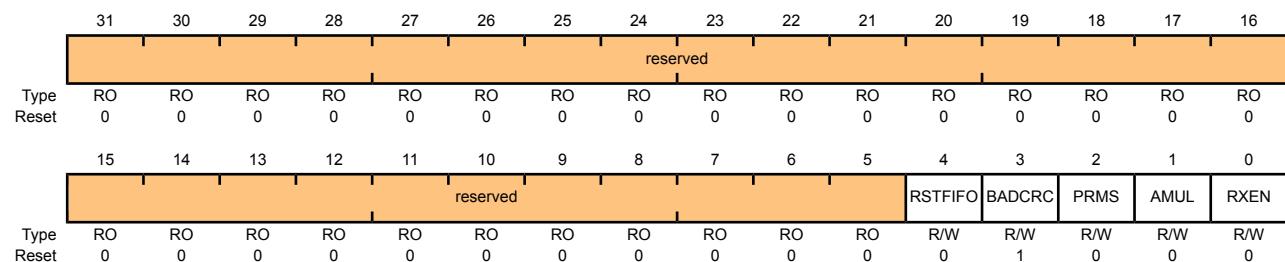
Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PHYINTM	R/W	1	Mask PHY Interrupt  This bit masks the <b>PHYINT</b> bit in the <b>MACRIS</b> register from being asserted.
5	MDINTM	R/W	1	Mask MII Transaction Complete  This bit masks the <b>MDINT</b> bit in the <b>MACRIS</b> register from being asserted.
4	RXERM	R/W	1	Mask Receive Error  This bit masks the <b>RXER</b> bit in the <b>MACRIS</b> register from being asserted.
3	FOVM	R/W	1	Mask FIFO Overrun  This bit masks the <b>FOV</b> bit in the <b>MACRIS</b> register from being asserted.
2	TXEMPM	R/W	1	Mask Transmit FIFO Empty  This bit masks the <b>TXEMP</b> bit in the <b>MACRIS</b> register from being asserted.
1	TXERM	R/W	1	Mask Transmit Error  This bit masks the <b>TXER</b> bit in the <b>MACRIS</b> register from being asserted.
0	RXINTM	R/W	1	Mask Packet Received  This bit masks the <b>RXINT</b> bit in the <b>MACRIS</b> register from being asserted.

## Register 4: Ethernet MAC Receive Control (MACRCTL), offset 0x008

This register enables software to configure the receive module and control the types of frames that are received from the physical medium. It is important to note that when the receive module is enabled, all valid frames with a broadcast address of FF-FF-FF-FF-FF-FF in the Destination Address field will be received and stored in the RX FIFO, even if the **AMUL** bit is not set.

### Ethernet MAC Receive Control (MACRCTL)

Base 0x4004.8000  
Offset 0x008  
Type R/W, reset 0x0000.0008



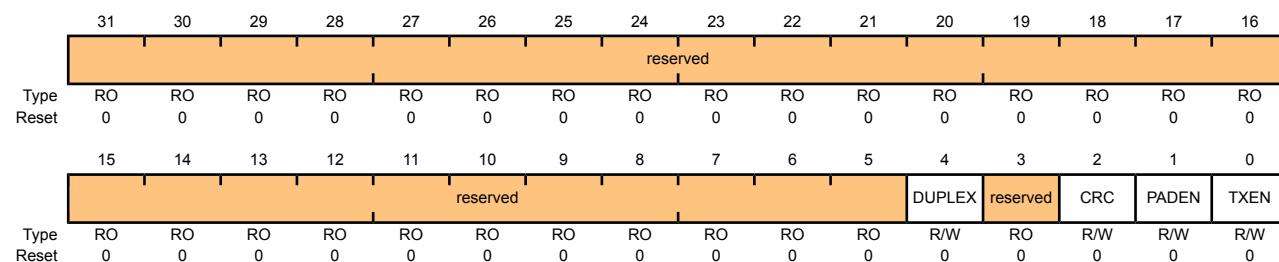
Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	RSTFIFO	R/W	0x0	Clear Receive FIFO  When set, clears the receive FIFO. This should be done when software initialization is performed.  It is recommended that the receiver be disabled ( <b>RXEN</b> = 0), and then the reset initiated ( <b>RSTFIFO</b> = 1). This sequence will flush and reset the RX FIFO.
3	BADCRC	R/W	0x1	Enable Reject Bad CRC  The <b>BADCRC</b> bit enables the rejection of frames with an incorrectly calculated CRC.
2	PRMS	R/W	0x0	Enable Promiscuous Mode  The <b>PRMS</b> bit enables Promiscuous mode, which accepts all valid frames, regardless of the Destination Address.
1	AMUL	R/W	0x0	Enable Multicast Frames  The <b>AMUL</b> bit enables the reception of multicast frames from the physical medium.
0	RXEN	R/W	0x0	Enable Receiver  The <b>RXEN</b> bit enables the Ethernet receiver. When this bit is Low, the receiver is disabled and all frames on the physical medium are ignored.

## Register 5: Ethernet MAC Transmit Control (MACTCTL), offset 0x00C

This register enables software to configure the transmit module, and control frames are placed onto the physical medium.

### Ethernet MAC Transmit Control (MACTCTL)

Base 0x4004.8000  
Offset 0x00C  
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	DUPLEX	R/W	0x0	Enable Duplex Mode When set, enables Duplex mode, allowing simultaneous transmission and reception.
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	CRC	R/W	0x0	Enable CRC Generation When set, enables the automatic generation of the CRC and the placement at the end of the packet. If this bit is not set, the frames placed in the TX FIFO will be sent exactly as they are written into the FIFO.
1	PADEN	R/W	0x0	Enable Packet Padding When set, enables the automatic padding of packets that do not meet the minimum frame size.
0	TXEN	R/W	0x0	Enable Transmitter When set, enables the transmitter. When this bit is 0, the transmitter is disabled.

## Register 6: Ethernet MAC Data (MACDATA), offset 0x010

This register enables software to access the TX and RX FIFOs.

Reads from this register return the data stored in the RX FIFO from the location indicated by the read pointer.

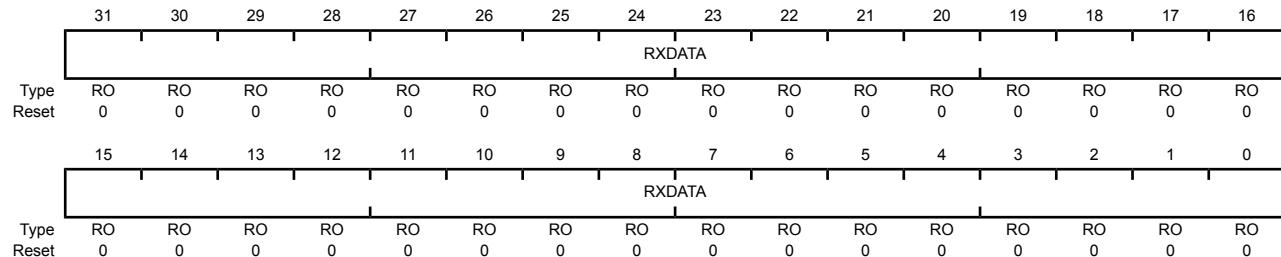
Writes to this register store the data in the TX FIFO at the location indicated by the write pointer. The write pointer is then auto-incremented to the next TX FIFO location.

There is no mechanism for randomly accessing bytes in either the RX or TX FIFOs. Data must be read from the RX FIFO sequentially and stored in a buffer for further processing. Once a read has been performed, the data in the FIFO cannot be re-read. Data must be written to the TX FIFO sequentially. If an error is made in placing the frame into the TX FIFO, the write pointer can be reset to the start of the TX FIFO by writing the TXER bit of the **MACIACK** register and then the data re-written.

### Read-Only Register

#### Ethernet MAC Data (MACDATA)

Base 0x4004.8000  
Offset 0x010  
Type RO, reset 0x0000.0000

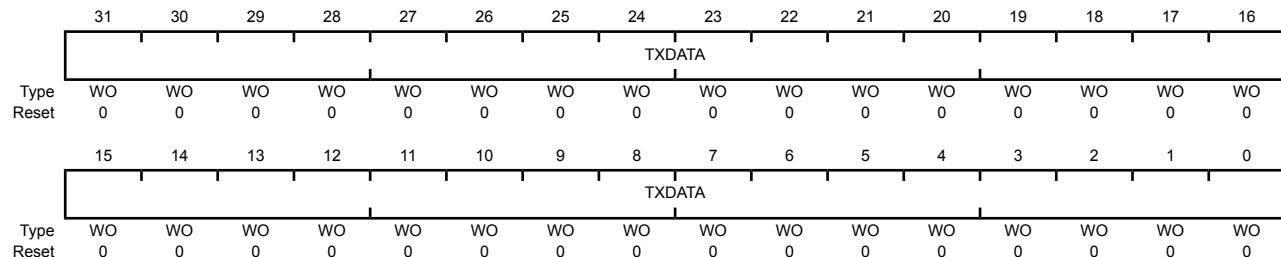


Bit/Field	Name	Type	Reset	Description
31:0	RXDATA	RO	0x0	Receive FIFO Data
				The RXDATA bits represent the next four bytes of data stored in the RX FIFO.

### Write-Only Register

#### Ethernet MAC Data (MACDATA)

Base 0x4004.8000  
Offset 0x010  
Type WO, reset 0x0000.0000



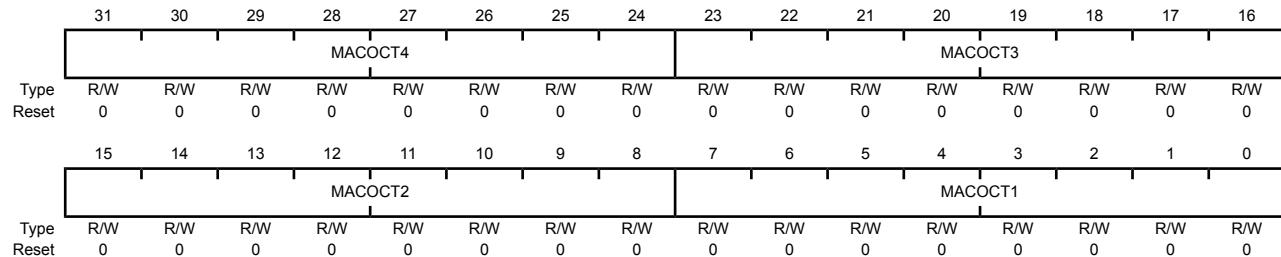
Bit/Field	Name	Type	Reset	Description
31:0	TXDATA	WO	0x0	<p>Transmit FIFO Data</p> <p>The TXDATA bits represent the next four bytes of data to place in the TX FIFO for transmission.</p>

## Register 7: Ethernet MAC Individual Address 0 (MACIA0), offset 0x014

This register enables software to program the first four bytes of the hardware MAC address of the Network Interface Card (NIC). (The last two bytes are in **MACIA1**). The 6-byte IAR is compared against the incoming Destination Address fields to determine whether the frame should be received.

### Ethernet MAC Individual Address 0 (MACIA0)

Base 0x4004.8000  
Offset 0x014  
Type R/W, reset 0x0000.0000



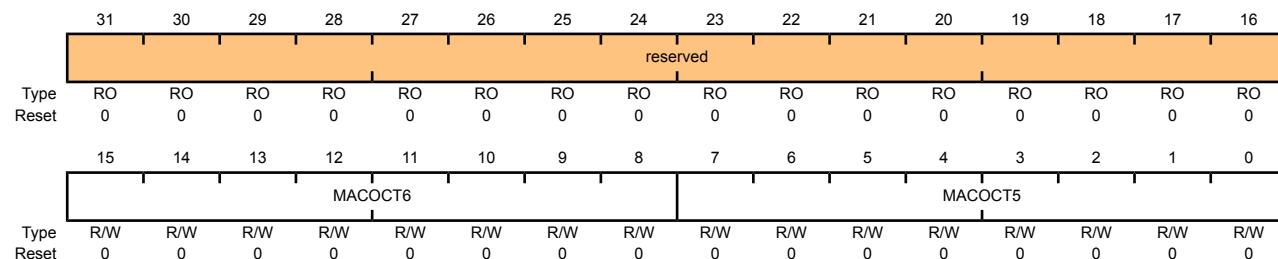
Bit/Field	Name	Type	Reset	Description
31:24	MACOCT4	R/W	0x0	MAC Address Octet 4  The MACOCT4 bits represent the fourth octet of the MAC address used to uniquely identify each Ethernet Controller.
23:16	MACOCT3	R/W	0x0	MAC Address Octet 3  The MACOCT3 bits represent the third octet of the MAC address used to uniquely identify each Ethernet Controller.
15:8	MACOCT2	R/W	0x0	MAC Address Octet 2  The MACOCT2 bits represent the second octet of the MAC address used to uniquely identify each Ethernet Controller.
7:0	MACOCT1	R/W	0x0	MAC Address Octet 1  The MACOCT1 bits represent the first octet of the MAC address used to uniquely identify each Ethernet Controller.

## Register 8: Ethernet MAC Individual Address 1 (MACIA1), offset 0x018

This register enables software to program the last two bytes of the hardware MAC address of the Network Interface Card (NIC). (The first four bytes are in **MACIA0**). The 6-byte IAR is compared against the incoming Destination Address fields to determine whether the frame should be received.

### Ethernet MAC Individual Address 1 (MACIA1)

Base 0x4004.8000  
Offset 0x018  
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.
15:8	MACOCT6	R/W	0x0	MAC Address Octet 6  The <b>MACOCT6</b> bits represent the sixth octet of the MAC address used to uniquely identify each Ethernet Controller.
7:0	MACOCT5	R/W	0x0	MAC Address Octet 5  The <b>MACOCT5</b> bits represent the fifth octet of the MAC address used to uniquely identify each Ethernet Controller.

## Register 9: Ethernet MAC Threshold (MACTHR), offset 0x01C

This register enables software to set the threshold level at which the transmission of the frame begins. If the THRESH bits are set to 0x3F, which is the reset value, transmission does not start until the NEWTX bit is set in the **MACTR** register. This effectively disables the early transmission feature.

Writing the THRESH bits to any value besides all 1s enables the early transmission feature. Once the byte count of data in the TX FIFO reaches this level, transmission of the frame begins. When THRESH is set to all 0s, transmission of the frame begins after 4 bytes (a single write) are stored in the TX FIFO. Each increment of the THRESH bit field waits for an additional 32 bytes of data (eight writes) to be stored in the TX FIFO. Therefore, a value of 0x01 would wait for 36 bytes of data to be written while a value of 0x02 would wait for 68 bytes to be written. In general, early transmission starts when:

$$\text{Number of Bytes} \geq 4 (\text{THRESH} \times 8 + 1)$$

Reaching the threshold level has the same effect as setting the NEWTX bit in the **MACTR** register. Transmission of the frame begins and then the number of bytes indicated by the Data Length field is sent out on the physical medium. Because under-run checking is not performed, it is possible that the tail pointer may reach and pass the write pointer in the TX FIFO. This causes indeterminate values to be written to the physical medium rather than the end of the frame. Therefore, sufficient bus bandwidth for writing to the TX FIFO must be guaranteed by the software.

If a frame smaller than the threshold level needs to be sent, the NEWTX bit in the **MACTR** register must be set with an explicit write. This initiates the transmission of the frame even though the threshold limit has not been reached.

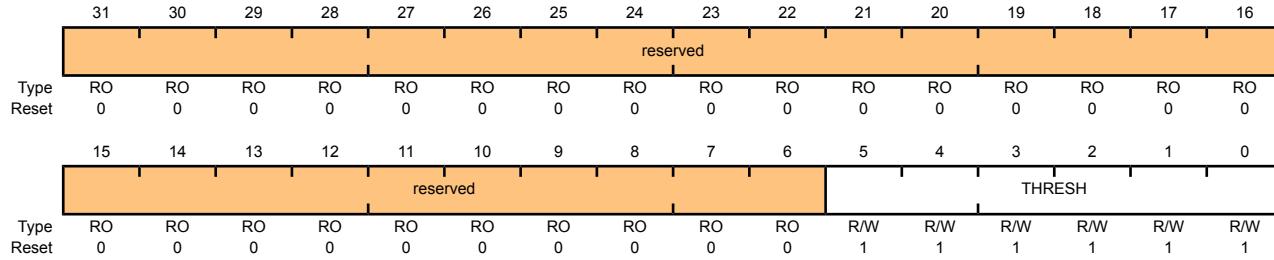
If the threshold level is set too small, it is possible for the transmitter to underrun. If this occurs, the transmit frame is aborted, and a transmit error occurs.

### Ethernet MAC Threshold (MACTHR)

Base 0x4004.8000

Offset 0x01C

Type R/W, reset 0x0000.003F



Bit/Field	Name	Type	Reset	Description
31:6	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	THRESH	R/W	0x3F	The THRESH bits represent the early transmit threshold. Once the amount of data in the TX FIFO exceeds this value, transmission of the packet begins.

## Register 10: Ethernet MAC Management Control (MACMCTL), offset 0x020

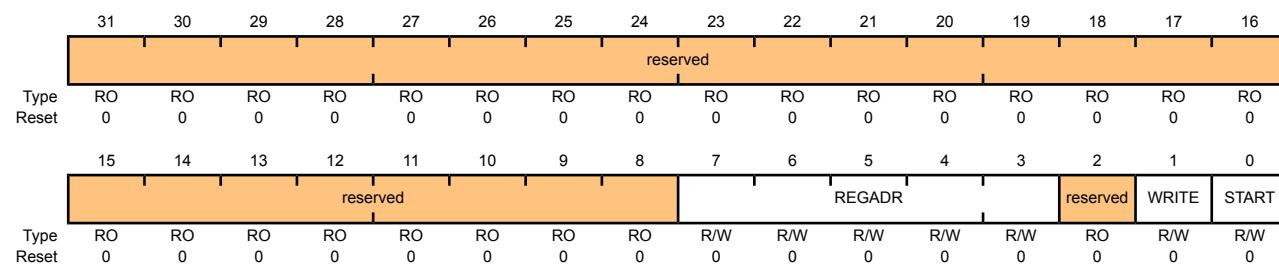
This register enables software to control the transfer of data to and from the MII Management registers in the Ethernet PHY. The address, name, type, reset configuration, and functional description of each of these registers can be found in Table 17-2 on page 459 and in “MII Management Register Descriptions” on page 478.

In order to initiate a *read* transaction from the MII Management registers, the **WRITE** bit must be written with a 0 during the same cycle that the **START** bit is written with a 1.

In order to initiate a *write* transaction to the MII Management registers, the **WRITE** bit must be written with a 1 during the same cycle that the **START** bit is written with a 1.

### Ethernet MAC Management Control (MACMCTL)

Base 0x4004.8000  
Offset 0x020  
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:3	REGADR	R/W	0x0	MII Register Address  The REGADR bit field represents the MII Management register address for the next MII management interface transaction.
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	WRITE	R/W	0x0	MII Register Transaction Type  The WRITE bit represents the operation of the next MII management interface transaction. If WRITE is set, the next operation will be a write; otherwise, it will be a read.
0	START	R/W	0x0	MII Register Transaction Enable  The START bit represents the initiation of the next MII management interface transaction. When a 1 is written to this bit, the MII register located at REGADR will be read (WRITE=0) or written (WRITE=1).

## Register 11: Ethernet MAC Management Divider (MACMDV), offset 0x024

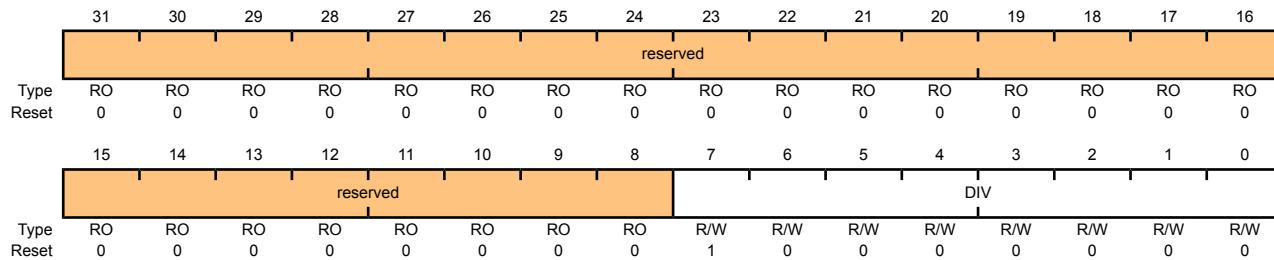
This register enables software to set the clock divider for the Management Data Clock (MDC). This clock is used to synchronize read and write transactions between the system and the MII Management registers. The frequency of the MDC clock can be calculated from the following formula:

$$F_{mdc} = F_{ipclk} / (2 * (MACMDVR + 1))$$

The clock divider must be written with a value that ensures that the MDC clock will not exceed a frequency of 2.5 MHz.

### Ethernet MAC Management Divider (MACMDV)

Base 0x4004.8000  
Offset 0x024  
Type R/W, reset 0x0000.0080



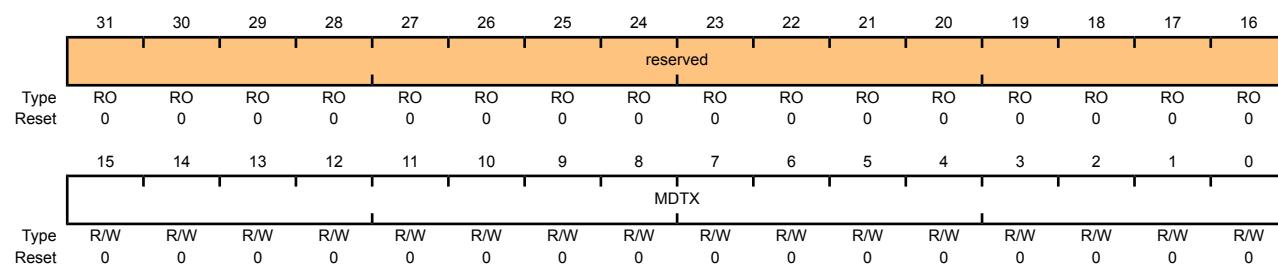
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DIV	R/W	0x80	Clock Divider  The DIV bits are used to set the clock divider for the MDC clock used to transmit data between the MAC and PHY over the serial MII interface.

## Register 12: Ethernet MAC Management Transmit Data (MACMTXD), offset 0x02C

This register holds the next value to be written to the MII Management registers.

### Ethernet MAC Management Transmit Data (MACMTXD)

Base 0x4004.8000  
Offset 0x02C  
Type R/W, reset 0x0000.0000



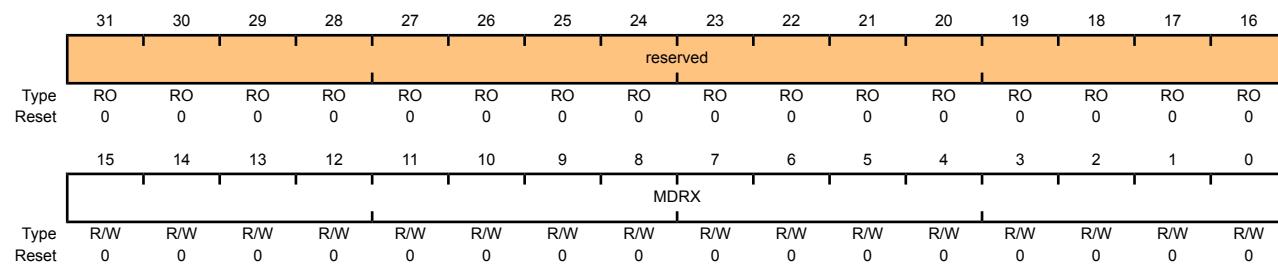
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	MDTX	R/W	0x0	MII Register Transmit Data  The MDTX bits represent the data that will be written in the next MII management transaction.

## Register 13: Ethernet MAC Management Receive Data (MACMRXD), offset 0x030

This register holds the last value read from the MII Management registers.

### Ethernet MAC Management Receive Data (MACMRXD)

Base 0x4004.8000  
Offset 0x030  
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	MDRX	R/W	0x0	MII Register Receive Data The MDRX bits represent the data that was read in the previous MII management transaction.											

## Register 14: Ethernet MAC Number of Packets (MACNP), offset 0x034

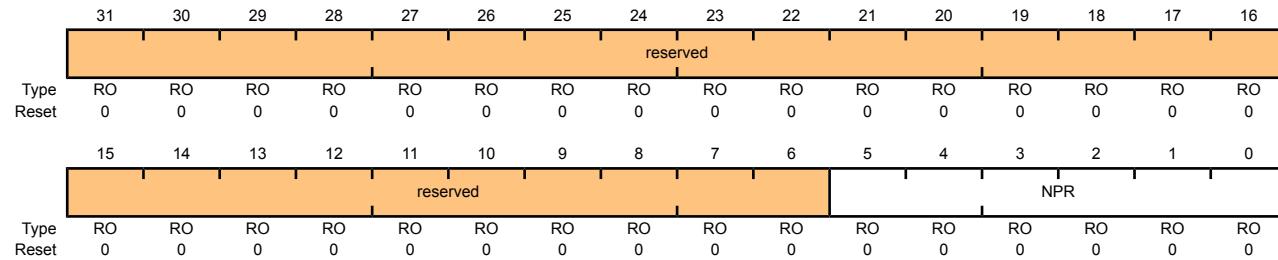
This register holds the number of frames that are currently in the RX FIFO. When **NPR** is 0, there are no frames in the RX FIFO and the **RXINT** bit is not set. When **NPR** is any other value, there is at least one frame in the RX FIFO and the **RXINT** bit in the **MACRIS** register is set.

### Ethernet MAC Number of Packets (MACNP)

Base 0x4004.8000

Offset 0x034

Type RO, reset 0x0000.0000

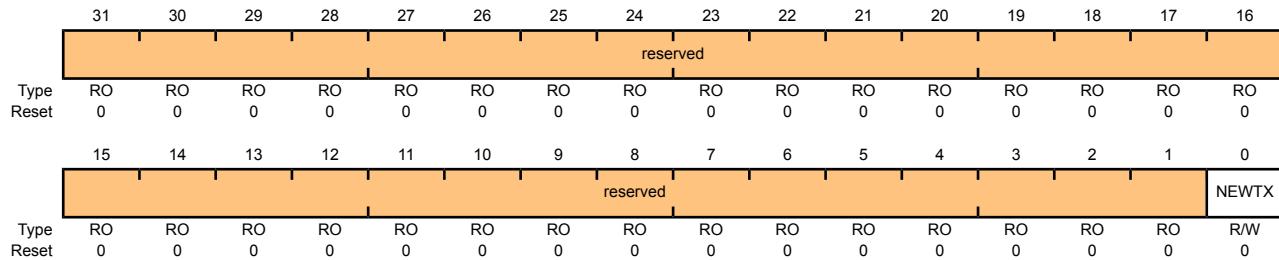


## Register 15: Ethernet MAC Transmission Request (MACTR), offset 0x038

This register enables software to initiate the transmission of the frame currently located in the TX FIFO to the physical medium. Once the frame has been transmitted to the medium from the TX FIFO or a transmission error has been encountered, the NEWTX bit is auto-cleared by the hardware.

### Ethernet MAC Transmission Request (MACTR)

Base 0x4004.8000  
Offset 0x038  
Type R/W, reset 0x0000.0000



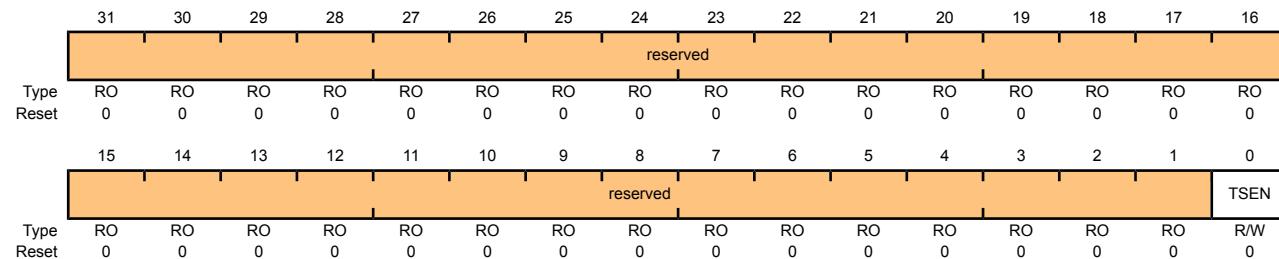
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	NEWTX	R/W	0x0	New Transmission  When set, the NEWTX bit initiates an Ethernet transmission once the packet has been placed in the TX FIFO. This bit is cleared once the transmission has been completed. If early transmission is being used (see the MACTHR register), this bit does not need to be set.

## Register 16: Ethernet MAC Timer Support (MACTS), offset 0x03C

This register enables software to enable timer support on the transmission and reception of frames. This register is only applicable for devices that have 1588 hardware support; for all others, a read returns 0s.

### Ethernet MAC Timer Support (MACTS)

Base 0x4004.8000  
Offset 0x03C  
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	TSEN	R/W	0x0	Time Stamp Enable When set, the TSEN bit multiplexes the TX and RX interrupts to the CCP inputs of General-Purpose Timer 3.

## 17.6 MII Management Register Descriptions

The *IEEE 802.3 standard* specifies a register set for controlling and gathering status from the PHY. The registers are collectively known as the MII Management registers. All addresses given are absolute. Addresses not listed are reserved. Also see “Ethernet MAC Register Descriptions” on page 460.

## Register 17: Ethernet PHY Management Register 0 – Control (MR0), address 0x00

This register enables software to configure the operation of the PHY. The default settings of these registers are designed to initialize the PHY to a normal operational mode without configuration.

Ethernet PHY Management Register 0 – Control (MR0)

Base 0x4004.8000  
Address 0x00  
Type R/W, reset 0x3100

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	R/W															
Reset	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
15	RESET	R/W	0	Reset Registers  When set, resets the registers to their default state and reinitializes internal state machines. Once the reset operation has completed, this bit is cleared by hardware.
14	LOOPBK	R/W	0	Loopback Mode  When set, enables the Loopback mode of operation. The receive circuitry is isolated from the physical medium and transmissions are sent back through the receive circuitry instead of the medium.
13	SPEEDSL	R/W	1	Speed Select  1: Enables the 100 Mb/s mode of operation (100BASE-TX). 0: Enables the 10 Mb/s mode of operation (10BASE-T).
12	ANEGEN	R/W	1	Auto-Negotiation Enable  When set, enables the Auto-Negotiation process.
11	PWRDN	R/W	0	Power Down  When set, places the PHY into a low-power consuming state.
10	ISO	R/W	0	Isolate  When set, isolates transmit and receive data paths and ignores all signaling on these buses.
9	RANEG	R/W	0	Restart Auto-Negotiation  When set, restarts the Auto-Negotiation process. Once the restart has initiated, this bit is cleared by hardware.
8	DUPLEX	R/W	1	Set Duplex Mode  1: Enables the Full-Duplex mode of operation. This bit can be set by software in a manual configuration process or by the Auto-Negotiation process. 0: Enables the Half-Duplex mode of operation.

Bit/Field	Name	Type	Reset	Description
7	COLT	R/W	0	Collision Test When set, enables the Collision Test mode of operation. The COLT bit asserts after the initiation of a transmission and de-asserts once the transmission is halted.
6:0	reserved	R/W	0x00	Write as 0, ignore on read.

## Register 18: Ethernet PHY Management Register 1 – Status (MR1), address 0x01

This register enables software to determine the capabilities of the PHY and perform its initialization and operation appropriately.

Ethernet PHY Management Register 1 – Status (MR1)

Base 0x4004.8000  
Address 0x01  
Type RO, reset 0x7849

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	reserved	100X_F	100X_H	10T_F	10T_H	reserved			MFPS	ANEGC	RFAULT	ANEGA	LINK	JAB	EXTD	
Reset	RO 0	RO 1	RO 1	RO 1	RO 1	RO 0	RO 0	RO 0	RO 1	RO 0	RC 0	RO 1	RO 0	RC 0	RO 1	

Bit/Field	Name	Type	Reset	Description
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	100X_F	RO	1	100BASE-TX Full-Duplex Mode  When set, indicates that the PHY is capable of supporting 100BASE-TX Full-Duplex mode.
13	100X_H	RO	1	100BASE-TX Half-Duplex Mode  When set, indicates that the PHY is capable of supporting 100BASE-TX Half-Duplex mode.
12	10T_F	RO	1	10BASE-T Full-Duplex Mode  When set, indicates that the PHY is capable of 10BASE-T Full-Duplex mode.
11	10T_H	RO	1	10BASE-T Half-Duplex Mode  When set, indicates that the PHY is capable of supporting 10BASE-T Half-Duplex mode.
10:7	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	MFPS	RO	1	Management Frames with Preamble Suppressed  When set, indicates that the Management Interface is capable of receiving management frames with the preamble suppressed.
5	ANEGC	RO	0	Auto-Negotiation Complete  When set, indicates that the Auto-Negotiation process has been completed and that the extended registers defined by the Auto-Negotiation protocol are valid.
4	RFAULT	RC	0	Remote Fault  When set, indicates that a remote fault condition has been detected. This bit remains set until it is read, even if the condition no longer exists.

Bit/Field	Name	Type	Reset	Description
3	ANEGA	RO	1	Auto-Negotiation When set, indicates that the PHY has the ability to perform Auto-Negotiation.
2	LINK	RO	0	Link Made When set, indicates that a valid link has been established by the PHY.
1	JAB	RC	0	Jabber Condition When set, indicates that a jabber condition has been detected by the PHY. This bit remains set until it is read, even if the jabber condition no longer exists.
0	EXTD	RO	1	Extended Capabilities When set, indicates that the PHY provides an extended set of capabilities that can be accessed through the extended register set.

## Register 19: Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2), address 0x02

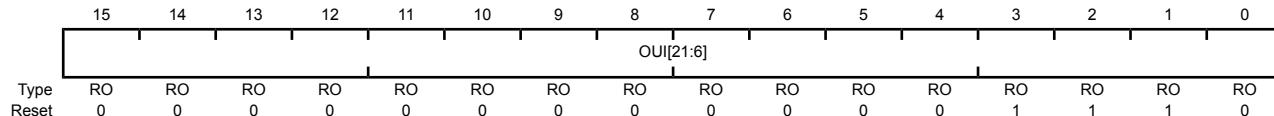
This register, along with **MR3**, provides a 32-bit value indicating the manufacturer, model, and revision information.

### Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2)

Base 0x4004.8000

Address 0x02

Type RO, reset 0x000E



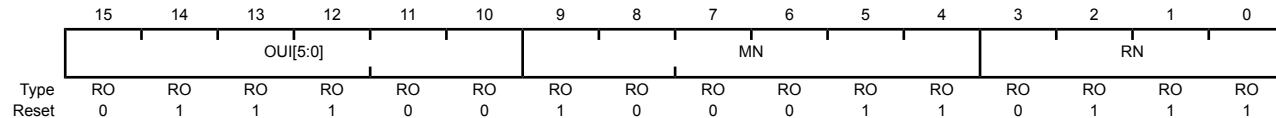
Bit/Field	Name	Type	Reset	Description
15:0	OUI[21:6]	RO	0x000E	Organizationally Unique Identifier[21:6] This field, along with the OUI[5:0] field in <b>MR3</b> , makes up the Organizationally Unique Identifier indicating the PHY manufacturer.

## Register 20: Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3), address 0x03

This register, along with **MR2**, provides a 32-bit value indicating the manufacturer, model, and revision information.

Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3)

Base 0x4004.8000  
Address 0x03  
Type RO, reset 0x7237



Bit/Field	Name	Type	Reset	Description
15:10	OUI[5:0]	RO	0x1C	Organizationally Unique Identifier[5:0]  This field, along with the OUI[21:6] field in <b>MR2</b> , makes up the Organizationally Unique Identifier indicating the PHY manufacturer.
9:4	MN	RO	0x23	Model Number  The MN field represents the Model Number of the PHY.
3:0	RN	RO	0x7	Revision Number  The RN field represents the Revision Number of the PHY.

## Register 21: Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4), address 0x04

This register provides the advertised abilities of the PHY used during Auto-Negotiation. Bits 8:5 represent the Technology Ability Field bits. This field can be overwritten by software to Auto-Negotiate to an alternate common technology. Writing to this register has no effect until Auto-Negotiation is re-initiated.

Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4)

Base 0x4004.8000  
Address 0x04  
Type R/W, reset 0x01E1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	NP	reserved	RF	reserved	reserved	reserved	A3	A2	A1	A0			S[4:0]			
Reset	RO 0	RO 0	R/W 0	RO 0	RO 0	RO 0	R/W 1	R/W 1	R/W 1	R/W 1	RO 0	RO 0	RO 0	RO 0	RO 1	

Bit/Field	Name	Type	Reset	Description
15	NP	RO	0	Next Page  When set, indicates the PHY is capable of Next Page exchanges to provide more detailed information on the PHY's capabilities.
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	RF	R/W	0	Remote Fault  When set, indicates to the link partner that a Remote Fault condition has been encountered.
12:9	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	A3	R/W	1	Technology Ability Field[3]  When set, indicates that the PHY supports the 100Base-TX full-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be written to 0 and Auto-Negotiation re-initiated with the RANEG bit in the <b>MR0</b> register.
7	A2	R/W	1	Technology Ability Field[2]  When set, indicates that the PHY supports the 100Base-T half-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be written to 0 and Auto-Negotiation re-initiated.
6	A1	R/W	1	Technology Ability Field[1]  When set, indicates that the PHY supports the 10Base-T full-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be written to 0 and Auto-Negotiation re-initiated.
5	A0	R/W	1	Technology Ability Field[0]  When set, indicates that the PHY supports the 10Base-T half-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be written to 0 and Auto-Negotiation re-initiated.

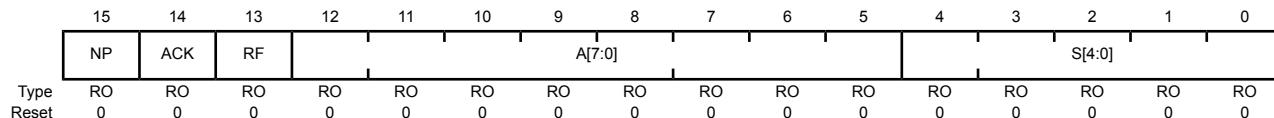
Bit/Field	Name	Type	Reset	Description
4:0	S[4:0]	RO	0x01	<p>Selector Field</p> <p>The S[4:0] field encodes 32 possible messages for communicating between PHYs. This field is hard-coded to 0x01, indicating that the Stellaris® PHY is <i>IEEE 802.3</i> compliant.</p>

## Register 22: Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5), address 0x05

This register provides the advertised abilities of the link partner's PHY that are received and stored during Auto-Negotiation.

Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5)

Base 0x4004.8000  
Address 0x05  
Type RO, reset 0x0000



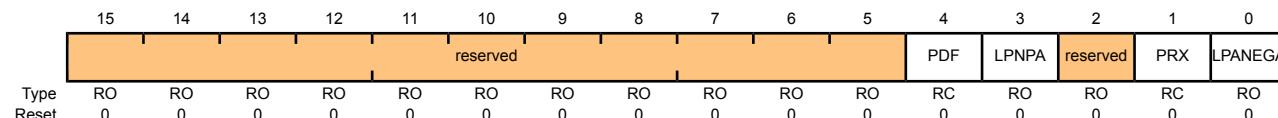
Bit/Field	Name	Type	Reset	Description														
15	NP	RO	0	Next Page When set, indicates that the link partner's PHY is capable of Next page exchanges to provide more detailed information on the PHY's capabilities.														
14	ACK	RO	0	Acknowledge When set, indicates that the device has successfully received the link partner's advertised abilities during Auto-Negotiation.														
13	RF	RO	0	Remote Fault Used as a standard transport mechanism for transmitting simple fault information.														
12:5	A[7:0]	RO	0x00	Technology Ability Field The A[7:0] field encodes individual technologies that are supported by the PHY. See the <b>MR4</b> register.														
4:0	S[4:0]	RO	0x00	Selector Field The S[4:0] field encodes possible messages for communicating between PHYs.														
				<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x00</td> <td>Reserved</td> </tr> <tr> <td>0x01</td> <td>IEEE Std 802.3</td> </tr> <tr> <td>0x02</td> <td>IEEE Std 802.9 ISLAN-16T</td> </tr> <tr> <td>0x03</td> <td>IEEE Std 802.5</td> </tr> <tr> <td>0x04</td> <td>IEEE Std 1394</td> </tr> <tr> <td>0x05–0x1F</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Description	0x00	Reserved	0x01	IEEE Std 802.3	0x02	IEEE Std 802.9 ISLAN-16T	0x03	IEEE Std 802.5	0x04	IEEE Std 1394	0x05–0x1F	Reserved
Value	Description																	
0x00	Reserved																	
0x01	IEEE Std 802.3																	
0x02	IEEE Std 802.9 ISLAN-16T																	
0x03	IEEE Std 802.5																	
0x04	IEEE Std 1394																	
0x05–0x1F	Reserved																	

## Register 23: Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6), address 0x06

This register enables software to determine the Auto-Negotiation and Next Page capabilities of the PHY and the link partner after Auto-Negotiation.

### Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6)

Base 0x4004.8000  
Address 0x06  
Type RO, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:5	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.
4	PDF	RC	0	Parallel Detection Fault  When set, indicates that more than one technology has been detected at link up. This bit is cleared when read.
3	LPNPA	RO	0	Link Partner is Next Page Able  When set, indicates that the link partner is Next Page Able.
2	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.
1	PRX	RC	0	New Page Received  When set, indicates that a New Page has been received from the link partner and stored in the appropriate location. This bit remains set until the register is read.
0	LPANEGA	RO	0	Link Partner is Auto-Negotiation Able  When set, indicates that the Link partner is Auto-Negotiation Able.

## Register 24: Ethernet PHY Management Register 16 – Vendor-Specific (MR16), address 0x10

This register enables software to configure the operation of vendor-specific modes of the PHY.

### Ethernet PHY Management Register 16 – Vendor-Specific (MR16)

Base 0x4004.8000  
Address 0x10  
Type R/W, reset 0x0140

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RPTR	INPOL	reserved	TXHIM	SQEI	NL10		reserved		APOL	RVSPOL		reserved	PCSBP	RXCC	
Reset	R/W 0	R/W 0	RO 0	R/W 0	R/W 0	R/W 0	RO 0	RO 1	RO 0	RO 1	R/W 0	R/W 0	RO 0	RO 0	R/W 0	R/W 0

Bit/Field	Name	Type	Reset	Description
15	RPTR	R/W	0	Repeater Mode  When set, enables the repeater mode of operation. In this mode, full-duplex is not allowed and the Carrier Sense signal only responds to receive activity. If the PHY is configured to 10Base-T mode, the SQE test function is disabled.
14	INPOL	R/W	0	Interrupt Polarity  1: Sets the polarity of the PHY interrupt to be active High. 0: Sets the polarity of the PHY interrupt to active Low.
<b>Important:</b> Because the Media Access Controller expects active Low interrupts from the PHY, this bit must always be written with a 0 to ensure proper operation.				
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	TXHIM	R/W	0	Transmit High Impedance Mode  When set, enables the transmitter High Impedance mode. In this mode, the TXOP and TXON transmitter pins are put into a high impedance state. The RXIP and RXIN pins remain fully functional.
11	SQEI	R/W	0	SQE Inhibit Testing  When set, prohibits 10Base-T SQE testing.  When 0, the SQE testing is performed by generating a Collision pulse following the completion of the transmission of a frame.
10	NL10	R/W	0	Natural Loopback Mode  When set, enables the 10Base-T Natural Loopback mode. This causes the transmission data received by the PHY to be looped back onto the receive data path when 10Base-T mode is enabled.
9:6	reserved	RO	0x05	Software should not rely on the value of 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	APOL	R/W	0	<p>Auto-Polarity Disable</p> <p>When set, disables the PHY's auto-polarity function.</p> <p>If this bit is 0, the PHY automatically inverts the received signal due to a wrong polarity connection during Auto-Negotiation if the PHY is in 10Base-T mode.</p>
4	RVSPOL	R/W	0	<p>Receive Data Polarity</p> <p>This bit indicates whether the receive data pulses are being inverted.</p> <p>If the APOL bit is 0, then the RVSPOL bit is read-only and indicates whether the auto-polarity circuitry is reversing the polarity. In this case, a 1 in the RVSPOL bit indicates that the receive data is inverted while a 0 indicates that the receive data is not inverted.</p> <p>If the APOL bit is 1, then the RVSPOL bit is writable and software can force the receive data to be inverted. Setting RVSPOL to 1 forces the receive data to be inverted while a 0 does not invert the receive data.</p>
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	PCSBP	R/W	0	<p>PCS Bypass</p> <p>When set, enables the bypass of the PCS and scrambling/descrambling functions in 100Base-TX mode. This mode is only valid when Auto-Negotiation is disabled and 100Base-T mode is enabled.</p>
0	RXCC	R/W	0	<p>Receive Clock Control</p> <p>When set, enables the Receive Clock Control power saving mode if the PHY is configured in 100Base-TX mode. This mode shuts down the receive clock when no data is being received from the physical medium to save power. This mode should not be used when PCSBP is enabled and is automatically disabled when the LOOPBK bit in the MR0 register is set.</p>

## Register 25: Ethernet PHY Management Register 17 – Interrupt Control/Status (MR17), address 0x11

This register provides the means for controlling and observing the events, which trigger a PHY interrupt in the **MACRIS** register. This register can also be used in a polling mode via the MII Serial Interface as a means to observe key events within the PHY via one register address. Bits 0 through 7 are status bits, which are each set to logic 1 based on an event. These bits are cleared after the register is read. Bits 8 through 15 of this register, when set to logic 1, enable their corresponding bit in the lower byte to signal a PHY interrupt in the **MACRIS** register.

Ethernet PHY Management Register 17 – Interrupt Control/Status (MR17)

Base 0x4004.8000

Address 0x11

Type R/W, reset 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	JABBER_IE	RXER_IE	PRX_IE	PDF_IE	LPACK_IE	LSCHG_IE	RFAULT_IE	ANECOMP_IE	JABBER_INT	RXER_INT	PRX_INT	PDF_INT	LPACK_INT	LSCHG_INT	RFAULT_INT	ANECOMP_INT
Reset	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	RC 0	RC 0	RC 0	RC 0	RC 0	RC 0	RC 0	RC 0

Bit/Field	Name	Type	Reset	Description
15	JABBER_IE	R/W	0	Jabber Interrupt Enable  When set, enables system interrupts when a Jabber condition is detected by the PHY.
14	RXER_IE	R/W	0	Receive Error Interrupt Enable  When set, enables system interrupts when a receive error is detected by the PHY.
13	PRX_IE	R/W	0	Page Received Interrupt Enable  When set, enables system interrupts when a new page is received by the PHY.
12	PDF_IE	R/W	0	Parallel Detection Fault Interrupt Enable  When set, enables system interrupts when a Parallel Detection Fault is detected by the PHY.
11	LPACK_IE	R/W	0	LP Acknowledge Interrupt Enable  When set, enables system interrupts when FLP bursts are received with the Acknowledge bit during Auto-Negotiation.
10	LSCHG_IE	R/W	0	Link Status Change Interrupt Enable  When set, enables system interrupts when the Link Status changes from OK to FAIL.
9	RFAULT_IE	R/W	0	Remote Fault Interrupt Enable  When set, enables system interrupts when a Remote Fault condition is signaled by the link partner.
8	ANECOMP_IE	R/W	0	Auto-Negotiation Complete Interrupt Enable  When set, enables system interrupts when the Auto-Negotiation sequence has completed successfully.

Bit/Field	Name	Type	Reset	Description
7	JABBER_INT	RC	0	Jabber Event Interrupt When set, indicates that a Jabber event has been detected by the 10Base-T circuitry.
6	RXER_INT	RC	0	Receive Error Interrupt When set, indicates that a receive error has been detected by the PHY.
5	PRX_INT	RC	0	Page Receive Interrupt When set, indicates that a new page has been received from the link partner during Auto-Negotiation.
4	PDF_INT	RC	0	Parallel Detection Fault Interrupt When set, indicates that a Parallel Detection Fault has been detected by the PHY during the Auto-Negotiation process.
3	LPACK_INT	RC	0	LP Acknowledge Interrupt When set, indicates that an FLP burst has been received with the Acknowledge bit set during Auto-Negotiation.
2	LSCHG_INT	RC	0	Link Status Change Interrupt When set, indicates that the link status has changed from OK to FAIL.
1	RFAULT_INT	RC	0	Remote Fault Interrupt When set, indicates that a Remote Fault condition has been signaled by the link partner.
0	ANECCOMP_INT	RC	0	Auto-Negotiation Complete Interrupt When set, indicates that the Auto-Negotiation sequence has completed successfully.

## Register 26: Ethernet PHY Management Register 18 – Diagnostic (MR18), address 0x12

This register enables software to diagnose the results of the previous Auto-Negotiation.

### Ethernet PHY Management Register 18 – Diagnostic (MR18)

Base 0x4004.8000  
Address 0x12  
Type RO, reset 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RO	RO	RO	RC	RO											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
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	ANEKF	RC	0	Auto-Negotiation Failure  When set, indicates that no common technology was found during Auto-Negotiation and has failed. This bit remains set until read.
11	DPLX	RO	0	Duplex Mode  When set, indicates that Full-Duplex was the highest common denominator found during the Auto-Negotiation process. Otherwise, Half-Duplex was the highest common denominator found.
10	RATE	RO	0	Rate  When set, indicates that 100Base-TX was the highest common denominator found during the Auto-Negotiation process. Otherwise, 10Base-T was the highest common denominator found.
9	RXSD	RO	0	Receive Detection  When set, indicates that receive signal detection has occurred (in 100Base-TX mode) or that Manchester-encoded data has been detected (in 10Base-T mode).
8	RX_LOCK	RO	0	Receive PLL Lock  When set, indicates that the Receive PLL has locked onto the receive signal for the selected speed of operation (10Base-T or 100Base-TX).
7:0	reserved	RO	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.

## Register 27: Ethernet PHY Management Register 19 – Transceiver Control (MR19), address 0x13

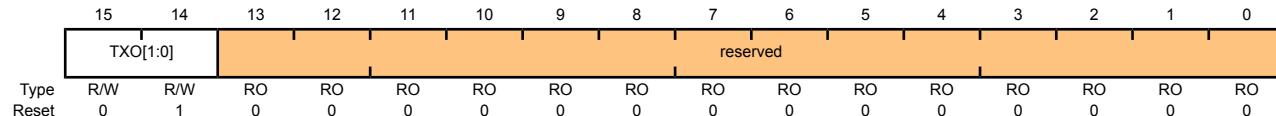
This register enables software to set the gain of the transmit output to compensate for transformer loss.

### Ethernet PHY Management Register 19 – Transceiver Control (MR19)

Base 0x4004.8000

Address 0x13

Type R/W, reset 0x4000



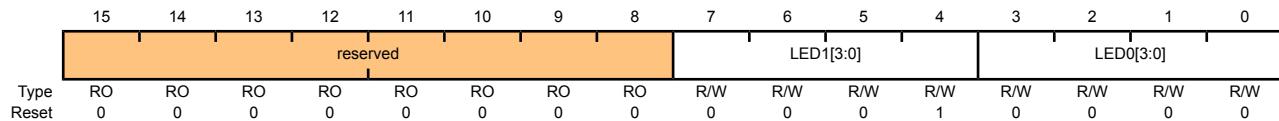
Bit/Field	Name	Type	Reset	Description
15:14	TXO[1:0]	R/W	1	Transmit Amplitude Selection  The TXO[1:0] field sets the transmit output amplitude to account for transmit transformer insertion loss.  Value Description 0x0 Gain set for 0.0dB of insertion loss 0x1 Gain set for 0.4dB of insertion loss 0x2 Gain set for 0.8dB of insertion loss 0x3 Gain set for 1.2dB of insertion loss
13:0	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 28: Ethernet PHY Management Register 23 – LED Configuration (MR23), address 0x17

This register enables software to select the source that will cause the LEDs to toggle.

### Ethernet PHY Management Register 23 – LED Configuration (MR23)

Base 0x4004.8000  
Address 0x17  
Type R/W, reset 0x0010



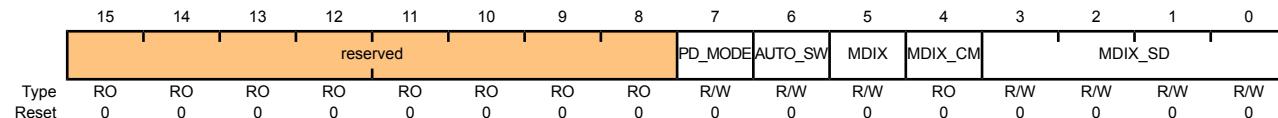
Bit/Field	Name	Type	Reset	Description																				
15:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.																				
7:4	LED1[3:0]	R/W	1	<p>LED1 Source</p> <p>The LED1 field selects the source that will toggle the LED1 signal.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr><td>0x0</td><td>Link OK</td></tr> <tr><td>0x1</td><td>RX or TX Activity (Default LED1)</td></tr> <tr><td>0x2</td><td>TX Activity</td></tr> <tr><td>0x3</td><td>RX Activity</td></tr> <tr><td>0x4</td><td>Collision</td></tr> <tr><td>0x5</td><td>100BASE-TX mode</td></tr> <tr><td>0x6</td><td>10BASE-T mode</td></tr> <tr><td>0x7</td><td>Full-Duplex</td></tr> <tr><td>0x8</td><td>Link OK &amp; Blink=RX or TX Activity</td></tr> </tbody> </table>	Value	Description	0x0	Link OK	0x1	RX or TX Activity (Default LED1)	0x2	TX Activity	0x3	RX Activity	0x4	Collision	0x5	100BASE-TX mode	0x6	10BASE-T mode	0x7	Full-Duplex	0x8	Link OK & Blink=RX or TX Activity
Value	Description																							
0x0	Link OK																							
0x1	RX or TX Activity (Default LED1)																							
0x2	TX Activity																							
0x3	RX Activity																							
0x4	Collision																							
0x5	100BASE-TX mode																							
0x6	10BASE-T mode																							
0x7	Full-Duplex																							
0x8	Link OK & Blink=RX or TX Activity																							
3:0	LED0[3:0]	R/W	0	<p>LED0 Source</p> <p>The LED0 field selects the source that will toggle the LED0 signal.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr><td>0x0</td><td>Link OK (Default LED0)</td></tr> <tr><td>0x1</td><td>RX or TX Activity</td></tr> <tr><td>0x2</td><td>TX Activity</td></tr> <tr><td>0x3</td><td>RX Activity</td></tr> <tr><td>0x4</td><td>Collision</td></tr> <tr><td>0x5</td><td>100BASE-TX mode</td></tr> <tr><td>0x6</td><td>10BASE-T mode</td></tr> <tr><td>0x7</td><td>Full-Duplex</td></tr> <tr><td>0x8</td><td>Link OK &amp; Blink=RX or TX Activity</td></tr> </tbody> </table>	Value	Description	0x0	Link OK (Default LED0)	0x1	RX or TX Activity	0x2	TX Activity	0x3	RX Activity	0x4	Collision	0x5	100BASE-TX mode	0x6	10BASE-T mode	0x7	Full-Duplex	0x8	Link OK & Blink=RX or TX Activity
Value	Description																							
0x0	Link OK (Default LED0)																							
0x1	RX or TX Activity																							
0x2	TX Activity																							
0x3	RX Activity																							
0x4	Collision																							
0x5	100BASE-TX mode																							
0x6	10BASE-T mode																							
0x7	Full-Duplex																							
0x8	Link OK & Blink=RX or TX Activity																							

## Register 29: Ethernet PHY Management Register 24 –MDI/MDIX Control (MR24), address 0x18

This register enables software to control the behavior of the MDI/MDIX mux and its switching capabilities.

### Ethernet PHY Management Register 24 –MDI/MDIX Control (MR24)

Base 0x4004.8000  
Address 0x18  
Type R/W, reset 0x00C0



Bit/Field	Name	Type	Reset	Description
15:8	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	PD_MODE	R/W	0	Parallel Detection Mode  When set, enables the Parallel Detection mode and allows auto-switching to work when Auto-Negotiation is not enabled.
6	AUTO_SW	R/W	0	Auto-Switching Enable  When set, enables Auto-Switching of the MDI/MDIX mux.
5	MDIX	R/W	0	Auto-Switching Configuration  When set, indicates that the MDI/MDIX mux is in the crossover (MDIX) configuration.  When 0, it indicates that the mux is in the pass-through (MDI) configuration.  When the AUTO_SW bit is 1, the MDIX bit is read-only. When the AUTO_SW bit is 0, the MDIX bit is read/write and can be configured manually.
4	MDIX_CM	RO	0	Auto-Switching Complete  When set, indicates that the auto-switching sequence has completed. If 0, it indicates that the sequence has not completed or that auto-switching is disabled.
3:0	MDIX_SD	R/W	0	Auto-Switching Seed  This field provides the initial seed for the switching algorithm. This seed directly affects the number of attempts [5,4] respectively to write bits [3:0].  A 0 sets the seed to 0x5.

## 18 Analog Comparator

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

The LM3S8962 controller provides one analog comparator that can be configured to drive an output or generate an interrupt or ADC event.

**Note:** Not all comparators have the option to drive an output pin. See the Comparator Operating Mode tables for more information.

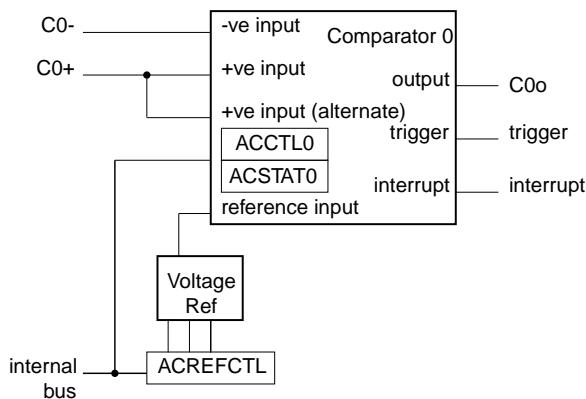
A comparator can compare a test voltage against any one of these voltages:

- 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.

### 18.1 Block Diagram

Figure 18-1. Analog Comparator Module Block Diagram



### 18.2 Functional Description

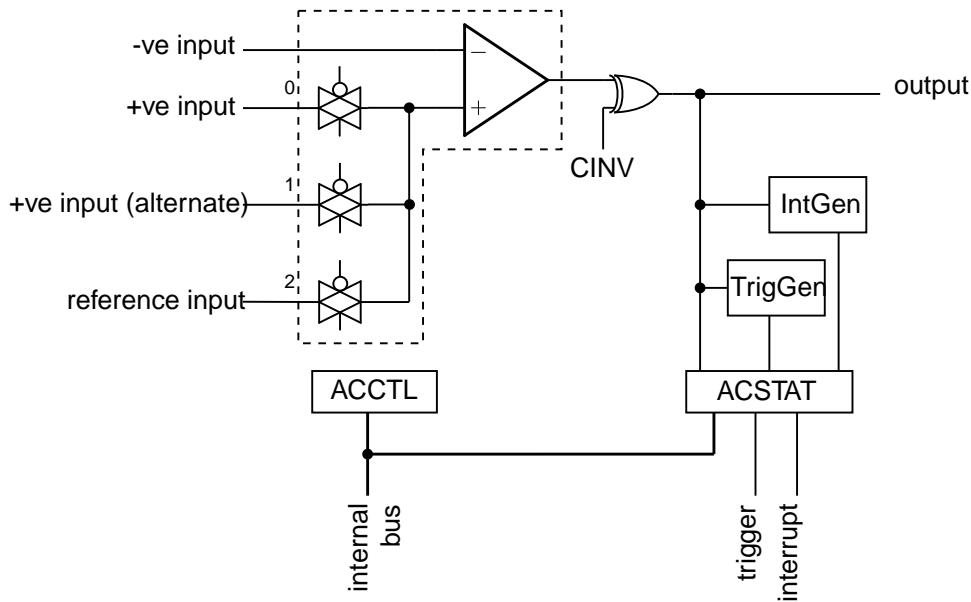
**Important:** It is recommended that the Digital-Input enable (the GPIODEN bit in the GPIO module) for the analog input pin be disabled to prevent excessive current draw from the I/O pads.

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 18-2 on page 498, 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 18-2. 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.

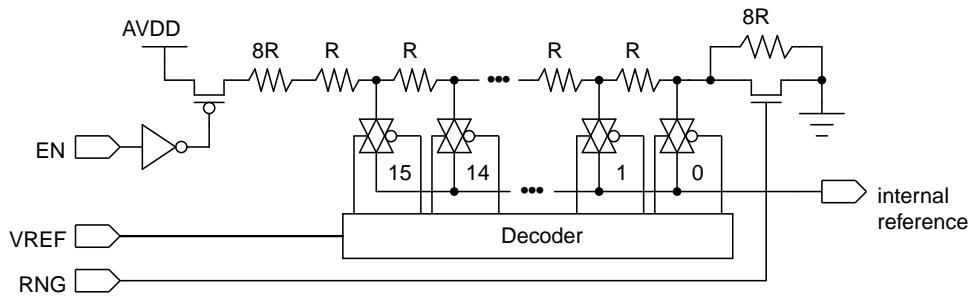
**Important:** Certain register bit values must be set before using the analog comparators. The proper pad configuration for the comparator input and output pins are described in the Comparator Operating Mode tables.

**Table 18-1. Comparator 0 Operating Modes**

ACCNTL0		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

## 18.2.1 Internal Reference Programming

The structure of the internal reference is shown in Figure 18-3 on page 499. This is controlled by a single configuration register (**ACREFCTL**). Table 18-2 on page 499 shows the programming options to develop specific internal reference values, to compare an external voltage against a particular voltage generated internally.

**Figure 18-3. Comparator Internal Reference Structure****Table 18-2. 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.
EN=1	RNG=0	<p>Total resistance in ladder is 32 R.</p> $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{(VREF + 8)}{32}$ $V_{REF} = 0.825 + 0.103 \cdot VREF$ <p>The range of internal reference in this mode is 0.825-2.37 V.</p>
	RNG=1	<p>Total resistance in ladder is 24 R.</p> $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{(VREF)}{24}$ $V_{REF} = 0.1375 \times VREF$ <p>The range of internal reference for this mode is 0.0-2.0625 V.</p>

## 18.3 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 on the C0o pin by writing the **ACCTL0** register with the value of 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.

## 18.4 Register Map

Table 18-3 on page 500 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.

**Table 18-3. Analog Comparators Register Map**

Offset	Name	Type	Reset	Description	See page
0x00	ACMIS	R/W1C	0x0000.0000	Analog Comparator Masked Interrupt Status	501
0x04	ACRIS	RO	0x0000.0000	Analog Comparator Raw Interrupt Status	502
0x08	ACINTEN	R/W	0x0000.0000	Analog Comparator Interrupt Enable	503
0x10	ACREFCTL	R/W	0x0000.0000	Analog Comparator Reference Voltage Control	504
0x20	ACSTAT0	RO	0x0000.0000	Analog Comparator Status 0	505
0x24	ACCTL0	R/W	0x0000.0000	Analog Comparator Control 0	506

## 18.5 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 0x00

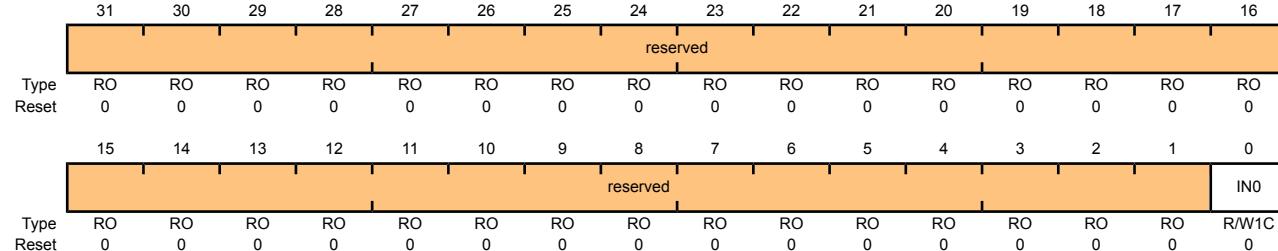
This register provides a summary of the interrupt status (masked) of the comparator.

### Analog Comparator Masked Interrupt Status (ACMIS)

Base 0x4003.C000

Offset 0x00

Type R/W1C, 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.
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 0x04

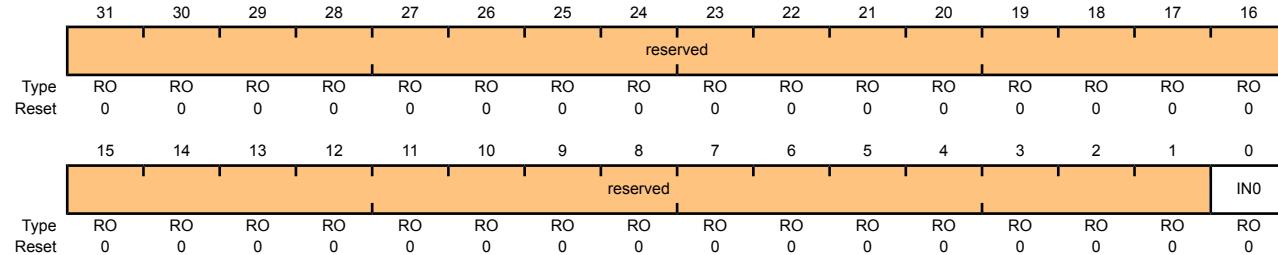
This register provides a summary of the interrupt status (raw) of the comparator.

### Analog Comparator Raw Interrupt Status (ACRIS)

Base 0x4003.C000

Offset 0x04

Type RO, reset 0x0000.0000



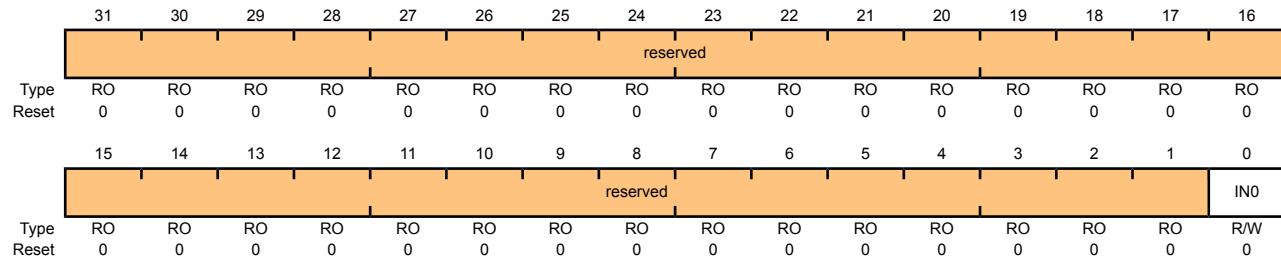
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	IN0	RO	0	Comparator 0 Interrupt Status When set, indicates that an interrupt has been generated by comparator 0.

## Register 3: Analog Comparator Interrupt Enable (ACINTEN), offset 0x08

This register provides the interrupt enable for the comparator.

### Analog Comparator Interrupt Enable (ACINTEN)

Base 0x4003.C000  
Offset 0x08  
Type R/W, reset 0x0000.0000



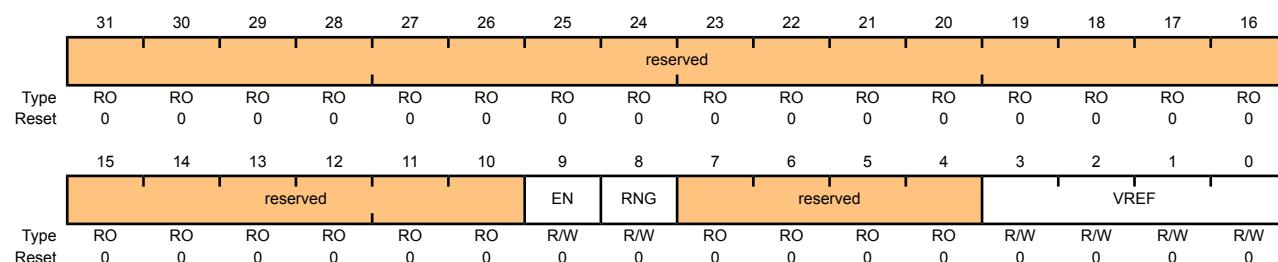
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	IN0	R/W	0	Comparator 0 Interrupt Enable When set, enables the controller interrupt from the comparator 0 output.

## Register 4: Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x10

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 0x10  
Type R/W, reset 0x0000.0000

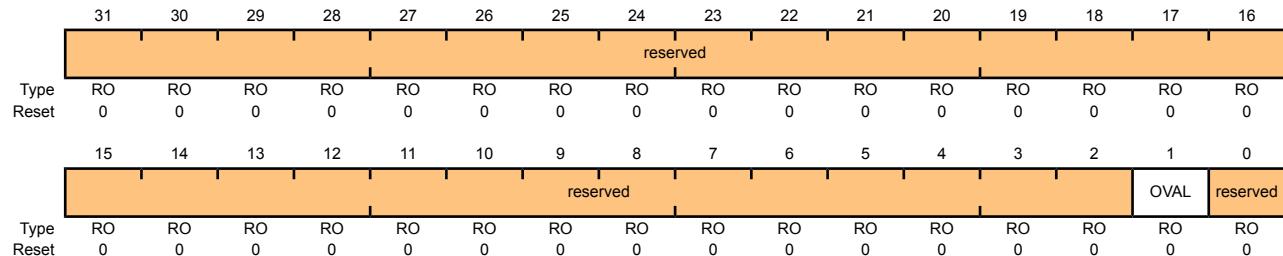


## Register 5: Analog Comparator Status 0 (ACSTAT0), offset 0x20

This register specifies the current output value of the comparator.

### Analog Comparator Status 0 (ACSTAT0)

Base 0x4003.C000  
Offset 0x20  
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 0x24

This register configures the comparator's input and output.

### Analog Comparator Control 0 (ACCTL0)

Base 0x4003.C000  
Offset 0x24  
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	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>Trigger Sense</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 <a href="#">TSLVAL</a></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 <a href="#">TSLVAL</a>	0x1	Falling edge	0x2	Rising edge	0x3	Either edge
Value	Function													
0x0	Level sense, see <a href="#">TSLVAL</a>													
0x1	Falling edge													
0x2	Rising edge													
0x3	Either edge													
4	ISLVAL	R/W	0	<p>Interrupt Sense Level Value</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>Interrupt Sense</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 <a href="#">ISLVAL</a></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 <a href="#">ISLVAL</a>	0x1	Falling edge	0x2	Rising edge	0x3	Either edge
Value	Function													
0x0	Level sense, see <a href="#">ISLVAL</a>													
0x1	Falling edge													
0x2	Rising edge													
0x3	Either edge													
1	CINV	R/W	0	<p>Comparator Output Invert</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.										

# 19 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. Each PWM generator block contains one timer (16-bit down or up/down counter), two PWM 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.

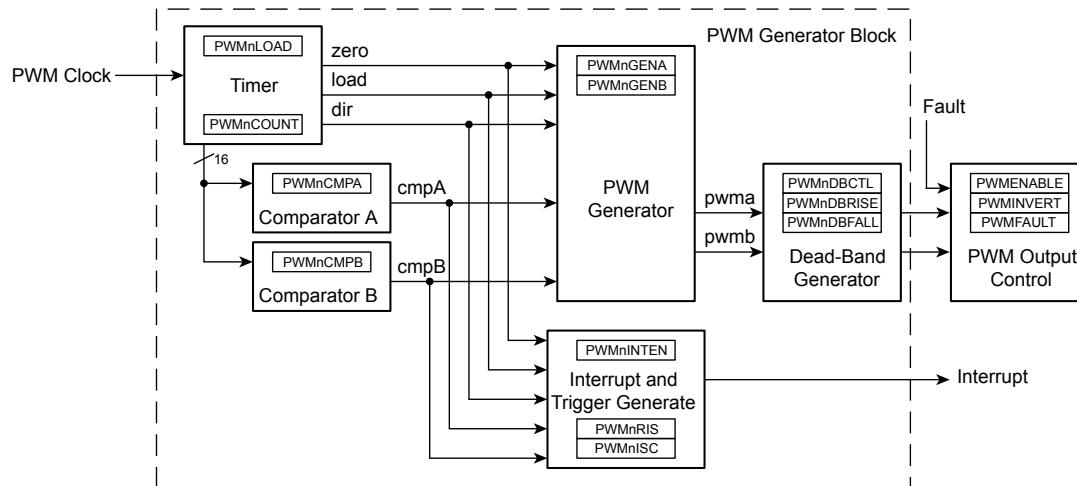
Each PWM generator block produces two PWM signals that can either be independent signals (other than being based on the same timer and therefore having the same frequency) or a single pair of complementary signals with dead-band delays inserted. The output of the PWM generation blocks are managed by 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. The three generator blocks can also generate the full six channels of gate controls required by a 3-phase inverter bridge.

## 19.1 Block Diagram

Figure 19-1 on page 508 provides a block diagram of a Stellaris® PWM module. The LM3S8962 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 19-1. PWM Module Block Diagram**



## 19.2 Functional Description

### 19.2.1 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.

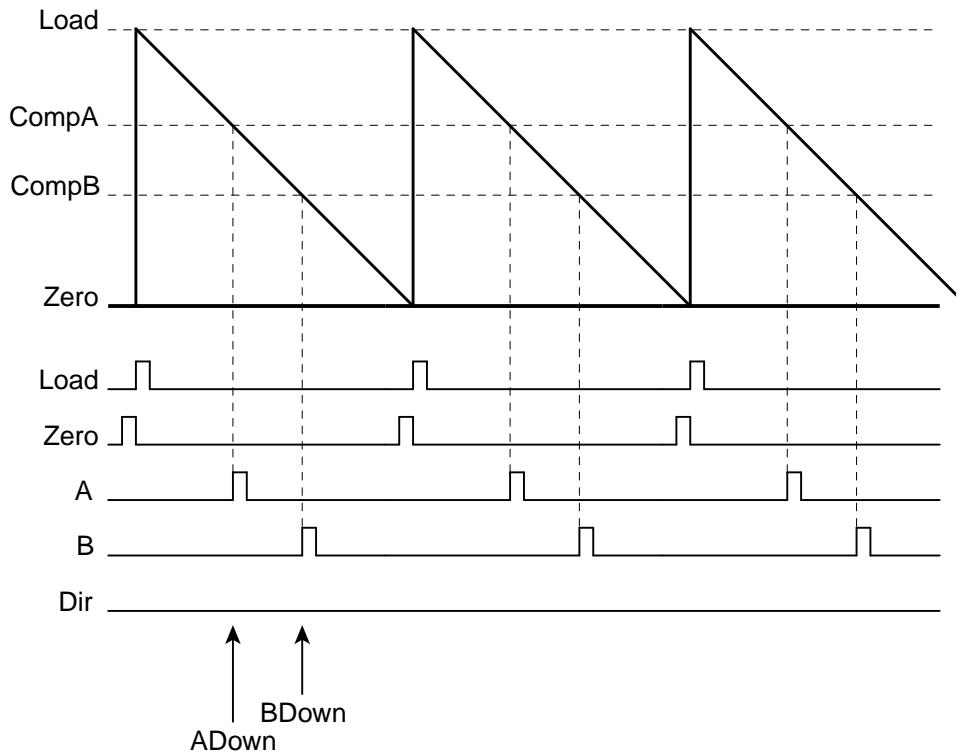
The timers output three signals that are used in the PWM generation process: the direction signal (this is always Low in Count-Down mode, but alternates between Low and High in Count-Up/Down mode), a single-clock-cycle-width High pulse when the counter is zero, and a single-clock-cycle-width High pulse 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.

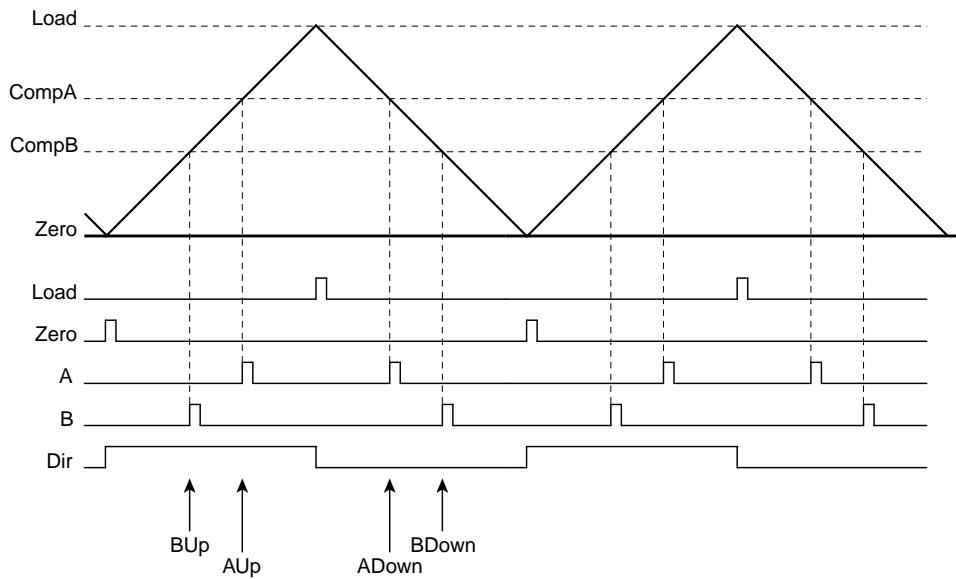
### 19.2.2 PWM Comparators

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 19-2 on page 509 shows the behavior of the counter and the relationship of these pulses when the counter is in Count-Down mode. Figure 19-3 on page 510 shows the behavior of the counter and the relationship of these pulses when the counter is in Count-Up/Down mode.

**Figure 19-2. PWM Count-Down Mode**

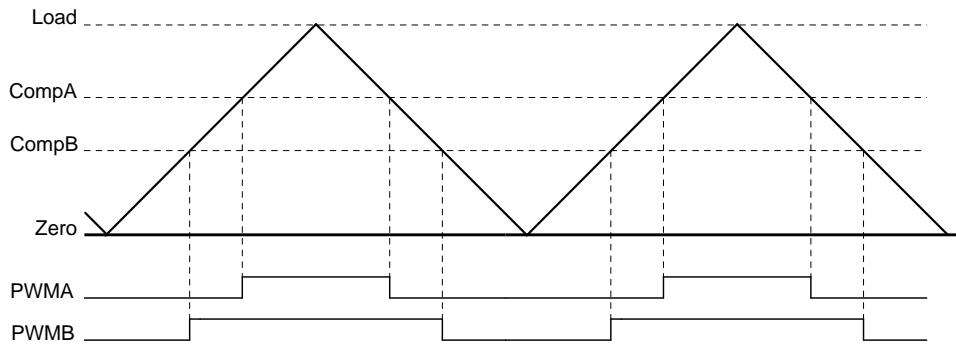


**Figure 19-3. PWM Count-Up/Down Mode**

### 19.2.3 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 19-4 on page 510 shows the use of Count-Up/Down mode to generate a pair of center-aligned, overlapped PWM signals that have different duty cycles.

**Figure 19-4. 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

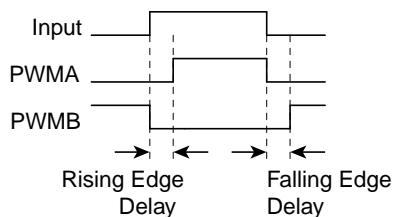
changes the duty cycle of the PWMA signal, and changing the value of comparator B changes the duty cycle of the PWMB signal.

#### 19.2.4 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 19-5 on page 511 shows the effect of the dead-band generator on an input PWM signal.

**Figure 19-5. PWM Dead-Band Generator**



#### 19.2.5 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.

#### 19.2.6 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.

### 19.2.7 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.

### 19.2.8 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.

## 19.3 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. 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.
4. 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).
5. 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.
6. 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.
7. Set the pulse width of the **PWM0** pin for a 25% duty cycle.
- Write the **PWM0CMPA** register with a value of 0x0000.012B.
8. Set the pulse width of the **PWM1** pin for a 75% duty cycle.
- Write the **PWM0CMPB** register with a value of 0x0000.0063.
9. Start the timers in PWM generator 0.
- Write the **PWM0CTL** register with a value of 0x0000.0001.
10. Enable PWM outputs.
- Write the **PWMENABLE** register with a value of 0x0000.0003.

## 19.4 Register Map

Table 19-1 on page 513 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.

**Table 19-1. PWM Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	PWMCTL	R/W	0x0000.0000	PWM Master Control	516
0x004	PWMSYNC	R/W	0x0000.0000	PWM Time Base Sync	517
0x008	PWMENABLE	R/W	0x0000.0000	PWM Output Enable	518
0x00C	PWMINVERT	R/W	0x0000.0000	PWM Output Inversion	519
0x010	PWMFAULT	R/W	0x0000.0000	PWM Output Fault	520
0x014	PWMINTEN	R/W	0x0000.0000	PWM Interrupt Enable	521
0x018	PWMRIS	RO	0x0000.0000	PWM Raw Interrupt Status	522
0x01C	PWMISC	R/W1C	0x0000.0000	PWM Interrupt Status and Clear	523
0x020	PWMSTATUS	RO	0x0000.0000	PWM Status	524
0x040	PWM0CTL	R/W	0x0000.0000	PWM0 Control	525
0x044	PWM0INTEN	R/W	0x0000.0000	PWM0 Interrupt and Trigger Enable	527
0x048	PWM0RIS	RO	0x0000.0000	PWM0 Raw Interrupt Status	529
0x04C	PWM0ISC	R/W1C	0x0000.0000	PWM0 Interrupt Status and Clear	530

Offset	Name	Type	Reset	Description	See page
0x050	PWM0LOAD	R/W	0x0000.0000	PWM0 Load	531
0x054	PWM0COUNT	RO	0x0000.0000	PWM0 Counter	532
0x058	PWM0CMPA	R/W	0x0000.0000	PWM0 Compare A	533
0x05C	PWM0CMPB	R/W	0x0000.0000	PWM0 Compare B	534
0x060	PWM0GENA	R/W	0x0000.0000	PWM0 Generator A Control	535
0x064	PWM0GENB	R/W	0x0000.0000	PWM0 Generator B Control	538
0x068	PWM0DBCTL	R/W	0x0000.0000	PWM0 Dead-Band Control	541
0x06C	PWM0DBRISE	R/W	0x0000.0000	PWM0 Dead-Band Rising-Edge Delay	542
0x070	PWM0DBFALL	R/W	0x0000.0000	PWM0 Dead-Band Falling-Edge-Delay	543
0x080	PWM1CTL	R/W	0x0000.0000	PWM1 Control	525
0x084	PWM1INTEN	R/W	0x0000.0000	PWM1 Interrupt and Trigger Enable	527
0x088	PWM1RIS	RO	0x0000.0000	PWM1 Raw Interrupt Status	529
0x08C	PWM1ISC	R/W1C	0x0000.0000	PWM1 Interrupt Status and Clear	530
0x090	PWM1LOAD	R/W	0x0000.0000	PWM1 Load	531
0x094	PWM1COUNT	RO	0x0000.0000	PWM1 Counter	532
0x098	PWM1CMPA	R/W	0x0000.0000	PWM1 Compare A	533
0x09C	PWM1CMPB	R/W	0x0000.0000	PWM1 Compare B	534
0x0A0	PWM1GENA	R/W	0x0000.0000	PWM1 Generator A Control	535
0x0A4	PWM1GENB	R/W	0x0000.0000	PWM1 Generator B Control	538
0x0A8	PWM1DBCTL	R/W	0x0000.0000	PWM1 Dead-Band Control	541
0x0AC	PWM1DBRISE	R/W	0x0000.0000	PWM1 Dead-Band Rising-Edge Delay	542
0x0B0	PWM1DBFALL	R/W	0x0000.0000	PWM1 Dead-Band Falling-Edge-Delay	543
0x0C0	PWM2CTL	R/W	0x0000.0000	PWM2 Control	525
0x0C4	PWM2INTEN	R/W	0x0000.0000	PWM2 Interrupt and Trigger Enable	527
0x0C8	PWM2RIS	RO	0x0000.0000	PWM2 Raw Interrupt Status	529
0x0CC	PWM2ISC	R/W1C	0x0000.0000	PWM2 Interrupt Status and Clear	530
0x0D0	PWM2LOAD	R/W	0x0000.0000	PWM2 Load	531
0x0D4	PWM2COUNT	RO	0x0000.0000	PWM2 Counter	532
0x0D8	PWM2CMPA	R/W	0x0000.0000	PWM2 Compare A	533
0x0DC	PWM2CMPB	R/W	0x0000.0000	PWM2 Compare B	534
0x0E0	PWM2GENA	R/W	0x0000.0000	PWM2 Generator A Control	535
0x0E4	PWM2GENB	R/W	0x0000.0000	PWM2 Generator B Control	538
0x0E8	PWM2DBCTL	R/W	0x0000.0000	PWM2 Dead-Band Control	541

Offset	Name	Type	Reset	Description	See page
0x0EC	PWM2DBRISE	R/W	0x0000.0000	PWM2 Dead-Band Rising-Edge Delay	542
0x0F0	PWM2DBFALL	R/W	0x0000.0000	PWM2 Dead-Band Falling-Edge-Delay	543

## 19.5 Register Descriptions

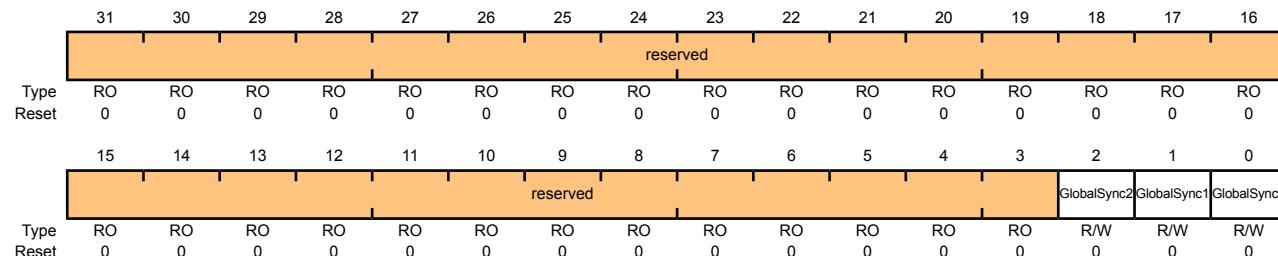
The remainder of this section lists and describes the PWM registers, in numerical order by address offset.

## 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



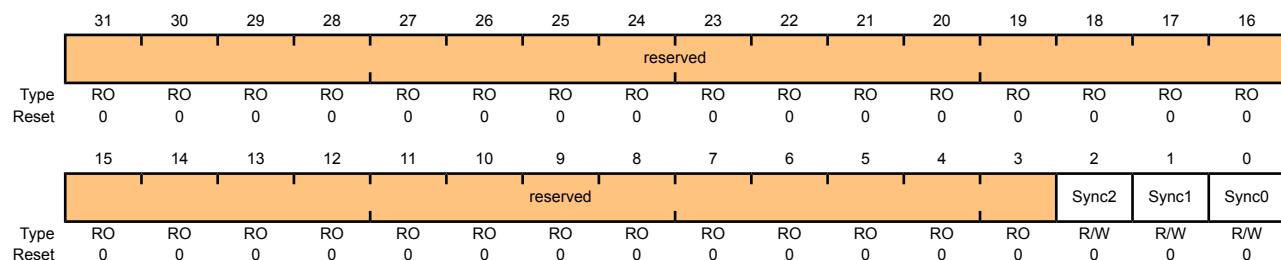
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	GlobalSync2	R/W	0	Update PWM Generator 2 Same as 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



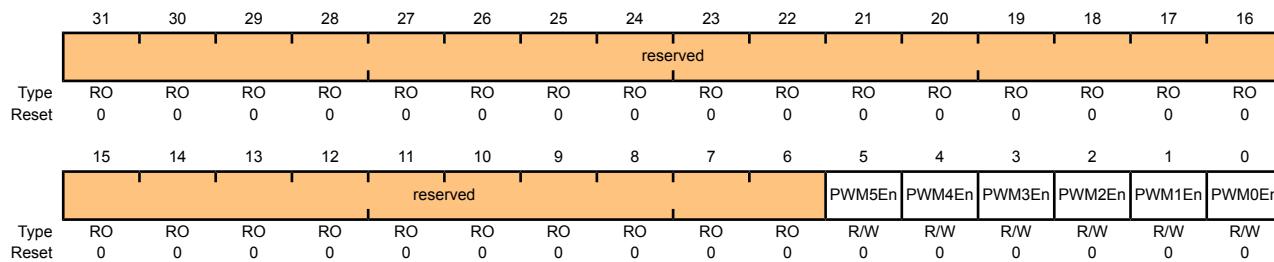
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	Sync2	R/W	0	Reset Generator 2 Counter Performs a 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 (PWMMENABLE), 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 (PWMMENABLE)

Base 0x4002.8000  
Offset 0x008  
Type R/W, reset 0x0000.0000



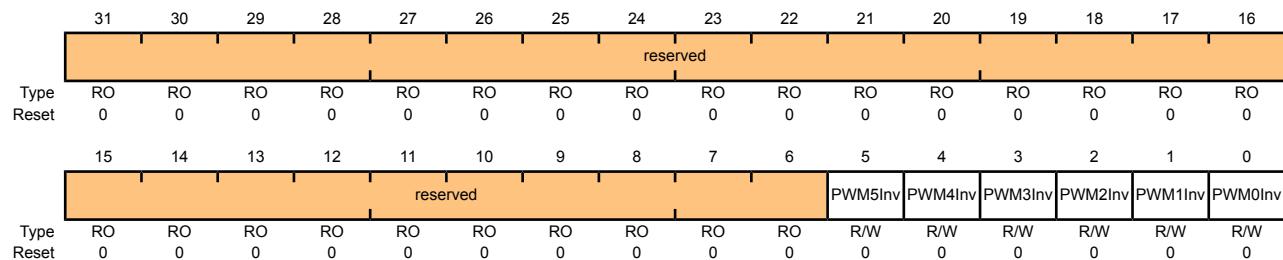
Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	PWM5En	R/W	0	PWM5 Output Enable  When set, allows 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



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 input and debug events are considered fault conditions. On a fault condition, each PWM signal can either 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 happens 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
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	Fault5	Fault4	Fault3	Fault2	Fault1	Fault0									
Reset	0	0	0	0	0	0	0	0	0	0	R/W	R/W	R/W	R/W	R/W	R/W

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 Driven Low on Fault  When set, the PWM5 output signal is driven Low on a fault condition.
4	Fault4	R/W	0	PWM4 Driven Low on Fault  When set, the PWM4 output signal is driven Low on a fault condition.
3	Fault3	R/W	0	PWM3 Driven Low on Fault  When set, the PWM3 output signal is driven Low on a fault condition.
2	Fault2	R/W	0	PWM2 Driven Low on Fault  When set, the PWM2 output signal is driven Low on a fault condition.
1	Fault1	R/W	0	PWM1 Driven Low on Fault  When set, the PWM1 output signal is driven Low on a fault condition.
0	Fault0	R/W	0	PWM0 Driven Low on 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
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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 1, 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 1, an interrupt occurs when the PWM generator 2 block asserts an interrupt.
1	IntPWM1	R/W	0	PWM1 Interrupt Enable  When 1, an interrupt occurs when the PWM generator 1 block asserts an interrupt.
0	IntPWM0	R/W	0	PWM0 Interrupt Enable  When 1, an interrupt occurs when the PWM generator 0 block asserts an interrupt.

## Register 7: PWM Raw Interrupt Status (PWMRIS), offset 0x018

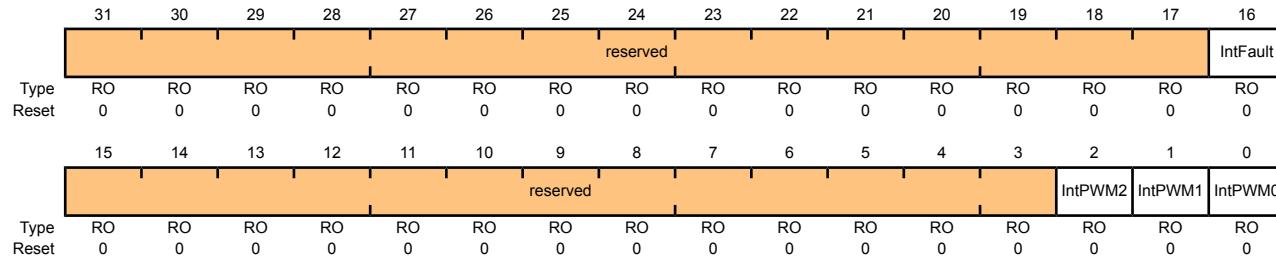
This register provides the current set of interrupt sources that are asserted, regardless of whether they cause an interrupt to be asserted to the controller. The fault interrupt is latched on detection; it must be cleared through the **PWM Interrupt Status and Clear (PWMISC)** register (see page 523). 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; a zero bit indicates that the event in question is not active.

### PWM Raw Interrupt Status (PWMRIS)

Base 0x4002.8000

Offset 0x018

Type RO, reset 0x0000.0000



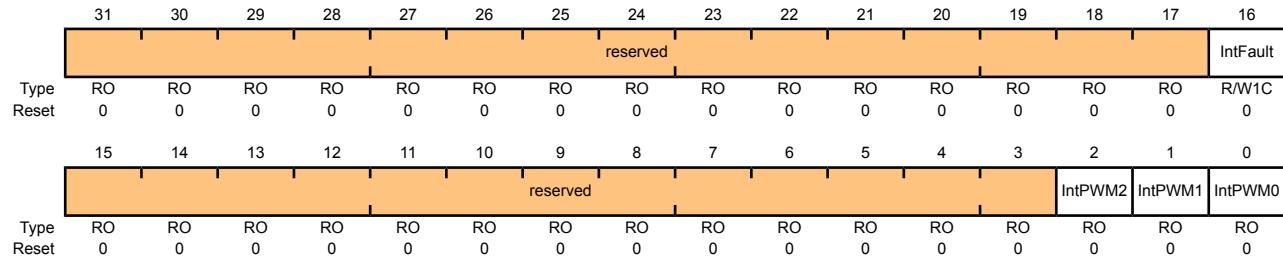
Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	IntFault	RO	0	Fault Interrupt Asserted Indicates that the fault input has been 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	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



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 if 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**

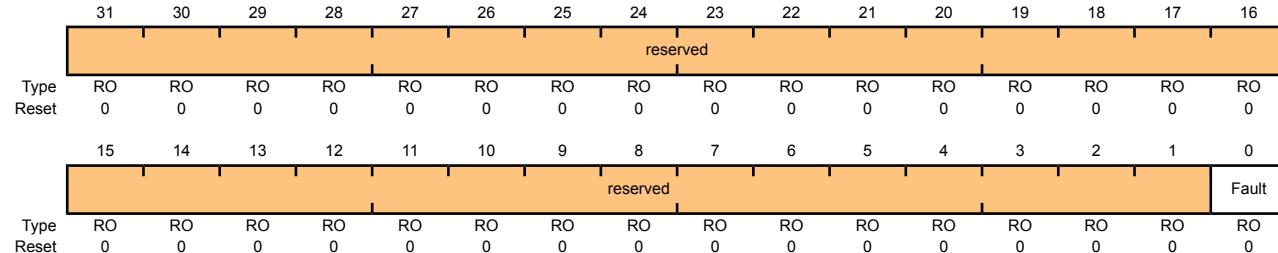
This register provides the status of the Fault input signal.

**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 to 1, indicates the fault input is asserted.

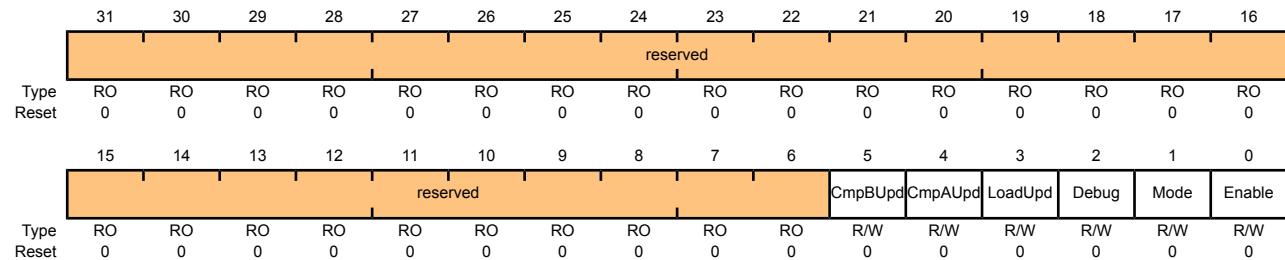
**Register 10: PWM0 Control (PWM0CTL), offset 0x040****Register 11: PWM1 Control (PWM1CTL), offset 0x080****Register 12: PWM2 Control (PWM2CTL), offset 0x0C0**

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. If 0, updates to the register are reflected to the comparator the next time the counter is 0. If 1, 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 516).
3	LoadUpd	R/W	0	Load Register Update Mode The Update mode for the load register. If 0, updates to the register are reflected to the counter the next time the counter is 0. If 1, 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. If 0, the counter stops running when it next reaches 0, and continues running again when no longer in Debug mode. If 1, the counter always runs.

Bit/Field	Name	Type	Reset	Description
1	Mode	R/W	0	<b>Counter Mode</b>  The mode for the counter. If 0, the counter counts down from the load value to 0 and then wraps back to the load value (Count-Down mode). If 1, 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. If 0, the entire block is disabled and not clocked. If 1, 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

																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	reserved	TrCmpBD	TrCmpBU	TrCmpAD	TrCmpAU	TrCntrLoad	TrCntrZero	8	7	6	5	4	3	2	1	0	reserved	IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntrLoad	IntCntrZero	R/W								
Type	RO	RO	R/W	R/W	R/W	R/W	R/W	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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	0x00	Software should not rely on the value of 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 When 1, a trigger pulse is output when the counter matches the comparator B value and the counter is counting down.
12	TrCmpBU	R/W	0	Trigger for Counter=Comparator B Up When 1, a trigger pulse is output when the counter matches the comparator B value and the counter is counting up.
11	TrCmpAD	R/W	0	Trigger for Counter=Comparator A Down When 1, a trigger pulse is output when the counter matches the comparator A value and the counter is counting down.

Bit/Field	Name	Type	Reset	Description
10	TrCmpAU	R/W	0	Trigger for Counter=Comparator A Up When 1, a trigger pulse is output when the counter matches the comparator A value and the counter is counting up.
9	TrCntLoad	R/W	0	Trigger for Counter=Load When 1, a trigger pulse is output when the counter matches the <b>PWMnLOAD</b> register.
8	TrCntZero	R/W	0	Trigger for Counter=0 When 1, a trigger pulse is output when the counter is 0.
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 When 1, an interrupt occurs when the counter matches the comparator B value and the counter is counting down.
4	IntCmpBU	R/W	0	Interrupt for Counter=Comparator B Up When 1, an interrupt occurs when the counter matches the comparator B value and the counter is counting up.
3	IntCmpAD	R/W	0	Interrupt for Counter=Comparator A Down When 1, an interrupt occurs when the counter matches the comparator A value and the counter is counting down.
2	IntCmpAU	R/W	0	Interrupt for Counter=Comparator A Up When 1, an interrupt occurs when the counter matches the comparator A value and the counter is counting up.
1	IntCntLoad	R/W	0	Interrupt for Counter=Load When 1, an interrupt occurs when the counter matches the <b>PWMnLOAD</b> register.
0	IntCntZero	R/W	0	Interrupt for Counter=0 When 1, an interrupt occurs when the counter is 0.

**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**

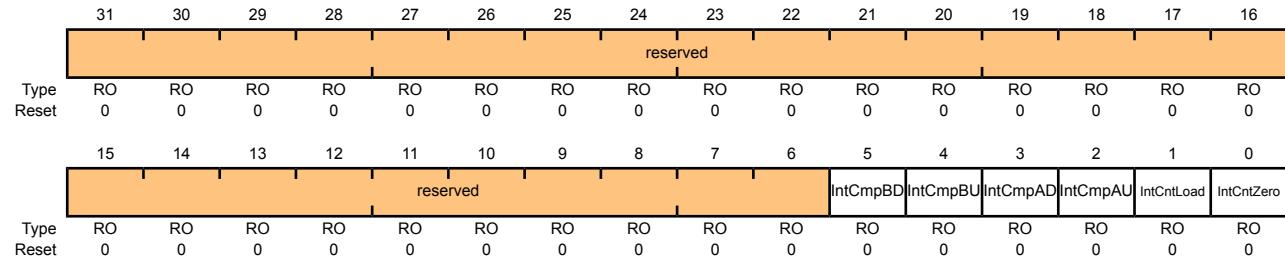
These registers provide the current set of interrupt sources that are asserted, regardless of whether they cause an interrupt to be asserted to the controller (**PWM0RIS** controls the PWM generator 0 block, and so on). Bits set to 1 indicate the latched events that have occurred; a 0 bit indicates that the event in question has not occurred.

**PWM0 Raw Interrupt Status (PWM0RIS)**

Base 0x4002.8000

Offset 0x048

Type RO, reset 0x0000.0000



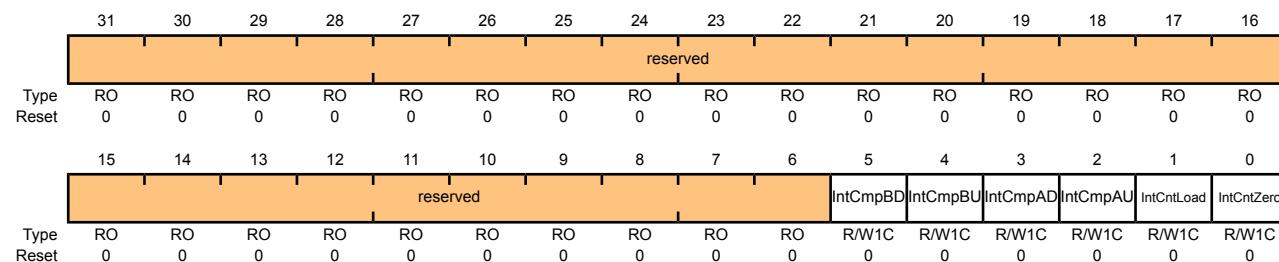
Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	IntCmpBD	RO	0	Comparator B Down Interrupt Status 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; a 0 bit indicates 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



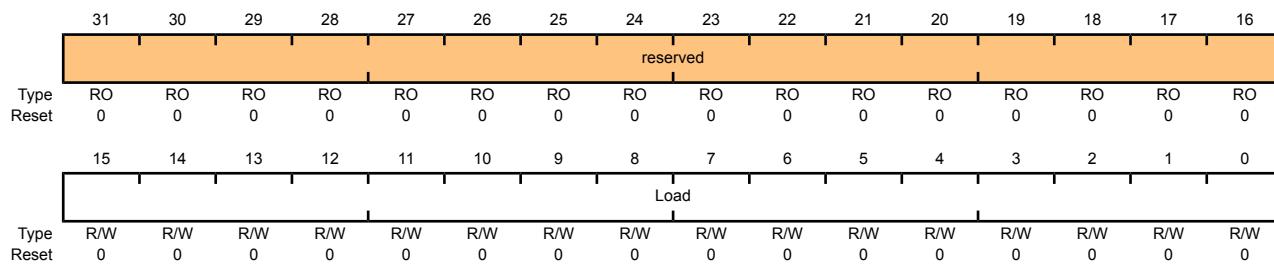
Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	IntCmpBD	R/W1C	0	Comparator B Down Interrupt Indicates 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 0xD0**

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 516). 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



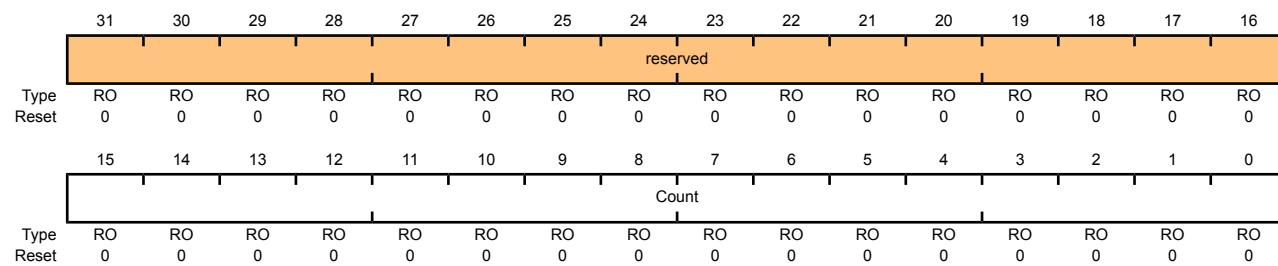
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	Load	R/W	0	Counter Load Value  The counter load 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 535 and page 538) or drive an interrupt or ADC trigger (via the **PWMnINTEN** register, see page 527). 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 531), then no pulse is ever output.

If the comparator A update mode is immediate (based on the CmpAUpd bit in the **PWMnCTL** register), then 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 516). 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, then no pulse is ever output.

If the comparator B update mode is immediate (based on the CmpBUpd bit in the **PWMnCTL** register), then 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 516). 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 0xE0**

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

reserved															
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
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved				ActCmpBD		ActCmpBU		ActCmpAD		ActCmpAU		ActLoad		ActZero	
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.
11:10	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 525) 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

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.													

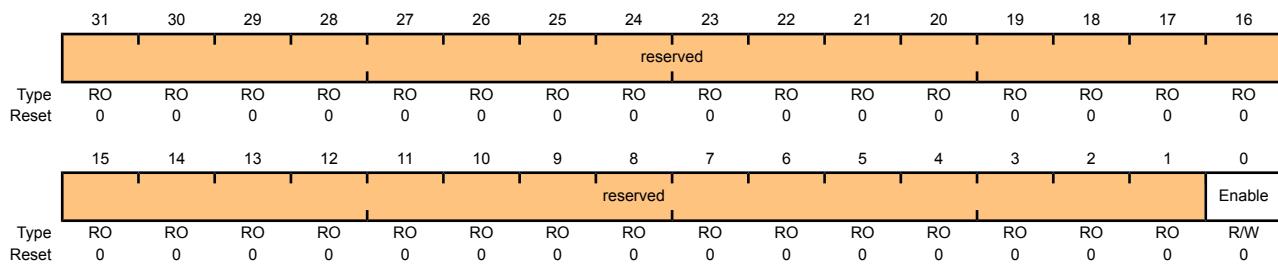
Bit/Field	Name	Type	Reset	Description
1:0	ActZero	R/W	0x0	Action for Counter=0 The action to be taken when the counter is 0. The table below defines the effect of the event on the output signal.
Value Description				
0x0 Do nothing.				
0x1 Invert the output signal.				
0x2 Set the output signal to 0.				
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 542), 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 543). 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



## 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

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

	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					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	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	FallDelay	R/W	0x00	Dead-Band Fall Delay The number of clock ticks to delay the falling edge.

## 20 Quadrature Encoder Interface (QEI)

A quadrature encoder, also known as a 2-channel incremental encoder, converts linear displacement into a pulse signal. By monitoring both the number of pulses and the relative phase of the two signals, you can track the position, direction of rotation, and speed. In addition, a third channel, or index signal, can be used to reset the position counter.

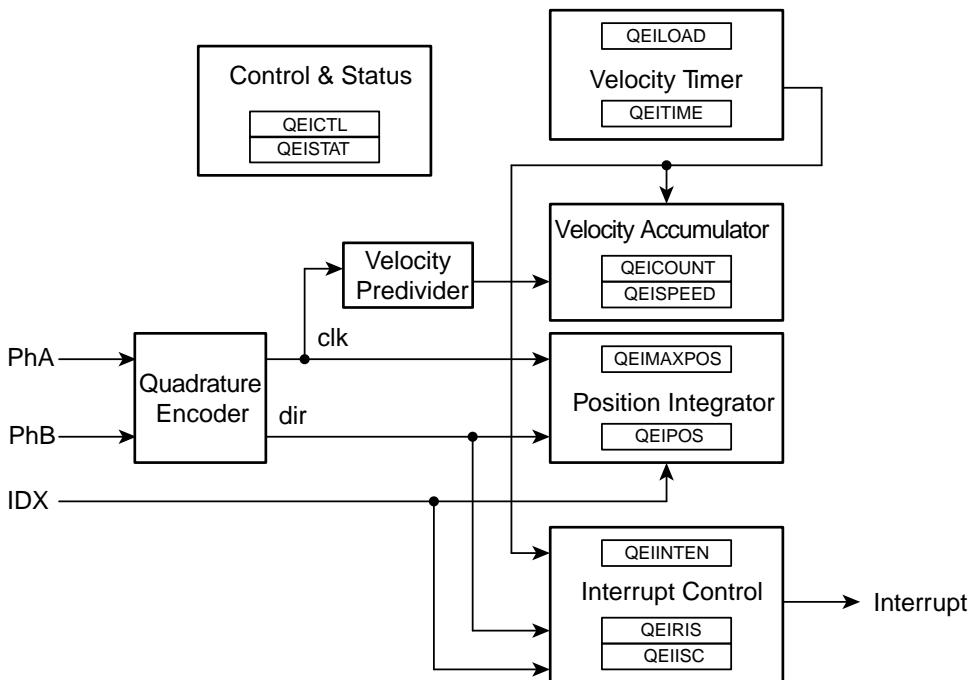
The LM3S8962 microcontroller includes two quadrature encoder interface (QEI) modules. Each QEI module interprets the code produced by a quadrature encoder wheel to integrate position over time and determine direction of rotation. In addition, it can capture a running estimate of the velocity of the encoder wheel.

Each Stellaris® quadrature encoder has the following features:

- Position integrator that tracks the encoder position
- Velocity capture using built-in timer
- Interrupt generation on:
  - Index pulse
  - Velocity-timer expiration
  - Direction change
  - Quadrature error detection

### 20.1 Block Diagram

Figure 20-1 on page 545 provides a block diagram of a Stellaris® QEI module.

**Figure 20-1. QEI Block Diagram**

## 20.2 Functional Description

The QEI module interprets the two-bit gray code produced by a quadrature encoder wheel to integrate position over time and determine direction of rotation. In addition, it can capture a running estimate of the velocity of the encoder wheel.

The position integrator and velocity capture can be independently enabled, though the position integrator must be enabled before the velocity capture can be enabled. The two phase signals, **PhA** and **PhB**, can be swapped before being interpreted by the QEI module to change the meaning of forward and backward, and to correct for miswiring of the system. Alternatively, the phase signals can be interpreted as a clock and direction signal as output by some encoders.

The QEI module supports two modes of signal operation: quadrature phase mode and clock/direction mode. In quadrature phase mode, the encoder produces two clocks that are 90 degrees out of phase; the edge relationship is used to determine the direction of rotation. In clock/direction mode, the encoder produces a clock signal to indicate steps and a direction signal to indicate the direction of rotation. This mode is determined by the **SigMode** bit of the **QEI Control (QEICTL)** register (see page 549).

When the QEI module is set to use the quadrature phase mode (**SigMode** bit equals zero), the capture mode for the position integrator can be set to update the position counter on every edge of the **PhA** signal or to update on every edge of both **PhA** and **PhB**. Updating the position counter on every **PhA** and **PhB** provides more positional resolution at the cost of less range in the positional counter.

When edges on **PhA** lead edges on **PhB**, the position counter is incremented. When edges on **PhB** lead edges on **PhA**, the position counter is decremented. When a rising and falling edge pair is seen on one of the phases without any edges on the other, the direction of rotation has changed.

The positional counter is automatically reset on one of two conditions: sensing the index pulse or reaching the maximum position value. Which mode is determined by the **ResMode** bit of the **QEI Control (QEICTL)** register.

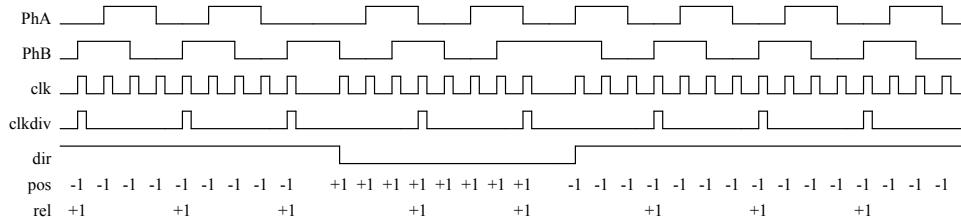
When **ResMode** is 0, the positional counter is reset when the index pulse is sensed. This limits the positional counter to the values [0:N-1], where N is the number of phase edges in a full revolution of the encoder wheel. The **QEIMAXPOS** register must be programmed with N-1 so that the reverse direction from position 0 can move the position counter to N-1. In this mode, the position register contains the absolute position of the encoder relative to the index (or home) position once an index pulse has been seen.

When **ResMode** is 1, the positional counter is constrained to the range [0:M], where M is the programmable maximum value. The index pulse is ignored by the positional counter in this mode.

The velocity capture has a configurable timer and a count register. It counts the number of phase edges (using the same configuration as for the position integrator) in a given time period. The edge count from the previous time period is available to the controller via the **QEISPEED** register, while the edge count for the current time period is being accumulated in the **QEICOUNT** register. As soon as the current time period is complete, the total number of edges counted in that time period is made available in the **QEISPEED** register (losing the previous value), the **QEICOUNT** is reset to 0, and counting commences on a new time period. The number of edges counted in a given time period is directly proportional to the velocity of the encoder.

Figure 20-2 on page 546 shows how the Stellaris® quadrature encoder converts the phase input signals into clock pulses, the direction signal, and how the velocity predivider operates (in Divide by 4 mode).

**Figure 20-2. Quadrature Encoder and Velocity Predivider Operation**



The period of the timer is configurable by specifying the load value for the timer in the **QEILoad** register. When the timer reaches zero, an interrupt can be triggered, and the hardware reloads the timer with the **QEILoad** value and continues to count down. At lower encoder speeds, a longer timer period is needed to be able to capture enough edges to have a meaningful result. At higher encoder speeds, both a shorter timer period and/or the velocity predivider can be used.

The following equation converts the velocity counter value into an rpm value:

$$\text{rpm} = (\text{clock} * (2 ^ \text{VelDiv}) * \text{Speed} * 60) \div (\text{Load} * \text{ppr} * \text{edges})$$

where:

**clock** is the controller clock rate

**ppr** is the number of pulses per revolution of the physical encoder

**edges** is 2 or 4, based on the capture mode set in the **QEICTL** register (2 for **CapMode** set to 0 and 4 for **CapMode** set to 1)

For example, consider a motor running at 600 rpm. A 2048 pulse per revolution quadrature encoder is attached to the motor, producing 8192 phase edges per revolution. With a velocity predivider of

$\div 1$  (VelDiv set to 0) and clocking on both PhA and PhB edges, this results in 81,920 pulses per second (the motor turns 10 times per second). If the timer were clocked at 10,000 Hz, and the load value was 2,500 ( $\frac{1}{4}$  of a second), it would count 20,480 pulses per update. Using the above equation:

$$\text{rpm} = (10000 * 1 * 20480 * 60) \div (2500 * 2048 * 4) = 600 \text{ rpm}$$

Now, consider that the motor is sped up to 3000 rpm. This results in 409,600 pulses per second, or 102,400 every  $\frac{1}{4}$  of a second. Again, the above equation gives:

$$\text{rpm} = (10000 * 1 * 102400 * 60) \div (2500 * 2048 * 4) = 3000 \text{ rpm}$$

Care must be taken when evaluating this equation since intermediate values may exceed the capacity of a 32-bit integer. In the above examples, the clock is 10,000 and the divider is 2,500; both could be predivided by 100 (at compile time if they are constants) and therefore be 100 and 25. In fact, if they were compile-time constants, they could also be reduced to a simple multiply by 4, cancelled by the  $\div 4$  for the edge-count factor.

**Important:** Reducing constant factors at compile time is the best way to control the intermediate values of this equation, as well as reducing the processing requirement of computing this equation.

The division can be avoided by selecting a timer load value such that the divisor is a power of 2; a simple shift can therefore be done in place of the division. For encoders with a power of 2 pulses per revolution, this is a simple matter of selecting a power of 2 load value. For other encoders, a load value must be selected such that the product is very close to a power of two. For example, a 100 pulse per revolution encoder could use a load value of 82, resulting in 32,800 as the divisor, which is 0.09% above  $2^{14}$ ; in this case a shift by 15 would be an adequate approximation of the divide in most cases. If absolute accuracy were required, the controller's divide instruction could be used.

The QEI module can produce a controller interrupt on several events: phase error, direction change, reception of the index pulse, and expiration of the velocity timer. Standard masking, raw interrupt status, interrupt status, and interrupt clear capabilities are provided.

## 20.3 Initialization and Configuration

The following example shows how to configure the Quadrature Encoder module to read back an absolute position:

1. Enable the QEI clock by writing a value of 0x0000.0100 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.
4. Configure the quadrature encoder to capture edges on both signals and maintain an absolute position by resetting on index pulses. Using a 1000-line encoder at four edges per line, there are 4000 pulses per revolution; therefore, set the maximum position to 3999 (0xF9F) since the count is zero-based.
  - Write the **QEICTL** register with the value of 0x0000.0018.

- Write the **QEIMAXPOS** register with the value of 0x0000.0F9F.
- 5. Enable the quadrature encoder by setting bit 0 of the **QEICTL** register.
- 6. Delay for some time.
- 7. Read the encoder position by reading the **QEIPOS** register value.

## 20.4 Register Map

Table 20-1 on page 548 lists the QEI registers. The offset listed is a hexadecimal increment to the register's address, relative to the module's base address:

- QEI0: 0x4002.C000
- QEI1: 0x4002.D000

**Table 20-1. QEI Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	QEICTL	R/W	0x0000.0000	QEI Control	549
0x004	QEISTAT	RO	0x0000.0000	QEI Status	551
0x008	QEIPOS	R/W	0x0000.0000	QEI Position	552
0x00C	QEIMAXPOS	R/W	0x0000.0000	QEI Maximum Position	553
0x010	QEILOAD	R/W	0x0000.0000	QEI Timer Load	554
0x014	QEITIME	RO	0x0000.0000	QEI Timer	555
0x018	QEICOUNT	RO	0x0000.0000	QEI Velocity Counter	556
0x01C	QEISPEED	RO	0x0000.0000	QEI Velocity	557
0x020	QEINTEN	R/W	0x0000.0000	QEI Interrupt Enable	558
0x024	QEIRIS	RO	0x0000.0000	QEI Raw Interrupt Status	559
0x028	QEISC	R/W1C	0x0000.0000	QEI Interrupt Status and Clear	560

## 20.5 Register Descriptions

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

## Register 1: QEI Control (QEICTL), offset 0x000

This register contains the configuration of the QEI module. Separate enables are provided for the quadrature encoder and the velocity capture blocks; the quadrature encoder must be enabled in order to capture the velocity, but the velocity does not need to be captured in applications that do not need it. The phase signal interpretation, phase swap, Position Update mode, Position Reset mode, and velocity predivider are all set via this register.

### QEI Control (QEICTL)

QEI0 base: 0x4002.C000  
QEI1 base: 0x4002.D000  
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			STALLEN	INVI	INVB	INVA	VelDiv			VelEn	ResMode	CapMode	SigMode	Swap	Enable
Type	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	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
31:13	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	STALLEN	R/W	0	Stall QEI  When set, the QEI stalls when the microcontroller asserts Halt.
11	INVI	R/W	0	Invert Index Pulse  When set, the input Index Pulse is inverted.
10	INVB	R/W	0	Invert PhB  When set, the PhB input is inverted.
9	INVA	R/W	0	Invert PhA  When set, the PhA input is inverted.
8:6	VelDiv	R/W	0x0	Predivide Velocity  A predivider of the input quadrature pulses before being applied to the QEICOUNT accumulator. This field can be set to the following values:

#### Value Predivider

0x0	÷1
0x1	÷2
0x2	÷4
0x3	÷8
0x4	÷16
0x5	÷32
0x6	÷64
0x7	÷128

Bit/Field	Name	Type	Reset	Description
5	VelEn	R/W	0	Capture Velocity When set, enables capture of the velocity of the quadrature encoder.
4	ResMode	R/W	0	Reset Mode The Reset mode for the position counter. When 0, the position counter is reset when it reaches the maximum; when 1, the position counter is reset when the index pulse is captured.
3	CapMode	R/W	0	Capture Mode The Capture mode defines the phase edges that are counted in the position. When 0, only the PhA edges are counted; when 1, the PhA and PhB edges are counted, providing twice the positional resolution but half the range.
2	SigMode	R/W	0	Signal Mode When 1, the PhA and PhB signals are clock and direction; when 0, they are quadrature phase signals.
1	Swap	R/W	0	Swap Signals Swaps the PhA and PhB signals.
0	Enable	R/W	0	Enable QEI Enables the quadrature encoder module.

## Register 2: QEI Status (QEISTAT), offset 0x004

This register provides status about the operation of the QEI module.

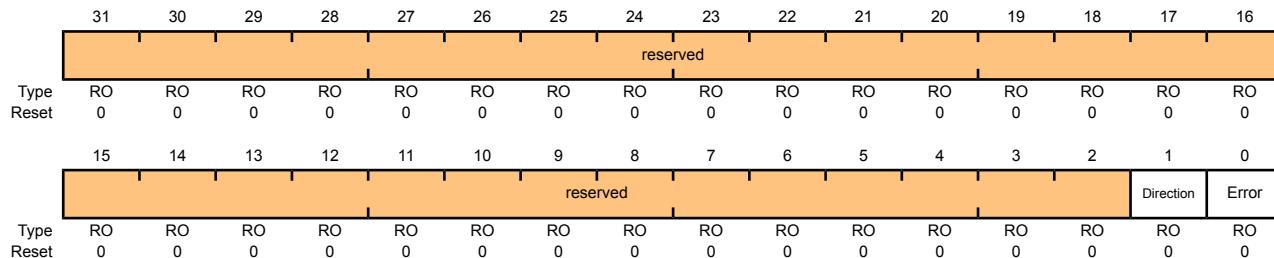
### QEI Status (QEISTAT)

QEI0 base: 0x4002.C000

QEI1 base: 0x4002.D000

Offset 0x004

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 Direction RO 0 Direction of Rotation

Indicates the direction the encoder is rotating.

The **Direction** values are defined as follows:

Value Description

0 Forward rotation

1 Reverse rotation

0 Error RO 0 Error Detected

Indicates that an error was detected in the gray code sequence (that is, both signals changing at the same time).

### Register 3: QEI Position (QEIPOS), offset 0x008

This register contains the current value of the position integrator. Its value is updated by inputs on the QEI phase inputs, and can be set to a specific value by writing to it.

#### QEI Position (QEIPOS)

QEI0 base: 0x4002.C000

QEI1 base: 0x4002.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
	Position															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Position															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	Position	R/W	0x00	Current Position Integrator Value The current value of the position integrator.

## Register 4: QEI Maximum Position (QEIMAXPOS), offset 0x00C

This register contains the maximum value of the position integrator. When moving forward, the position register resets to zero when it increments past this value. When moving backward, the position register resets to this value when it decrements from zero.

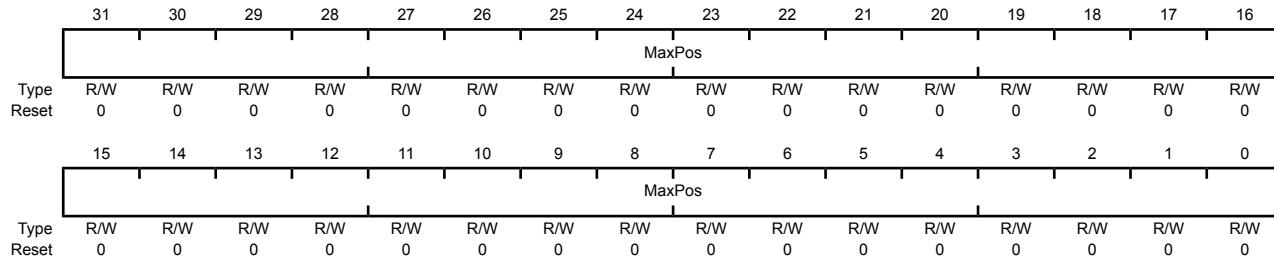
### QEI Maximum Position (QEIMAXPOS)

QEI0 base: 0x4002.C000

QEI1 base: 0x4002.D000

Offset 0x00C

Type R/W, reset 0x0000.0000



Bit/Field                  Name                  Type                  Reset                  Description

31:0                  MaxPos                  R/W                  0x00                  Maximum Position Integrator Value

The maximum value of the position integrator.

## Register 5: QEI Timer Load (QEILOAD), offset 0x010

This register contains the load value for the velocity timer. Since this value is loaded into the timer the clock cycle after the timer is zero, this value should be one less than the number of clocks in the desired period. So, for example, to have 2000 clocks per timer period, this register should contain 1999.

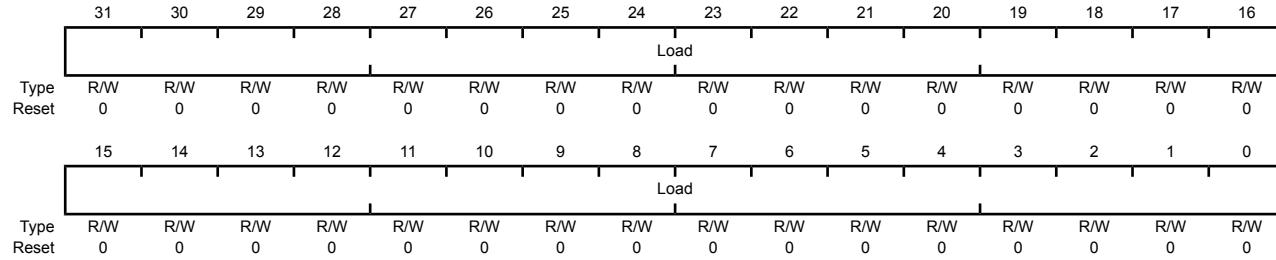
### QEI Timer Load (QEILOAD)

QEI0 base: 0x4002.C000

QEI1 base: 0x4002.D000

Offset 0x010

Type R/W, reset 0x0000.0000



Bit/Field                  Name                  Type                  Reset                  Description

31:0                  Load                  R/W                  0x00                  Velocity Timer Load Value

The load value for the velocity timer.

## Register 6: QEI Timer (QEITIME), offset 0x014

This register contains the current value of the velocity timer. This counter does not increment when **VelEn** in **QEICTL** is 0.

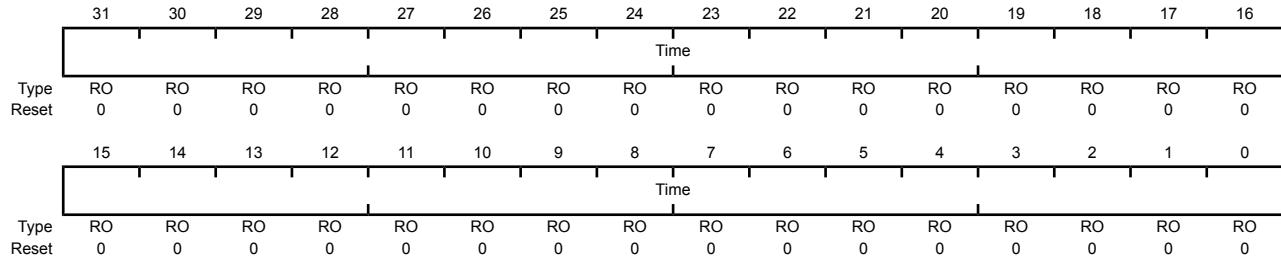
### QEI Timer (QEITIME)

QEI0 base: 0x4002.C000

QEI1 base: 0x4002.D000

Offset 0x014

Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	Time	RO	0x00	Velocity Timer Current Value The current value of the velocity timer.

## Register 7: QEI Velocity Counter (QEICOUNT), offset 0x018

This register contains the running count of velocity pulses for the current time period. Since this is a running total, the time period to which it applies cannot be known with precision (that is, a read of this register does not necessarily correspond to the time returned by the **QEITIME** register since there is a small window of time between the two reads, during which time either value may have changed). The **QEISPEED** register should be used to determine the actual encoder velocity; this register is provided for information purposes only. This counter does not increment when **VelEn** in **QEICTL** is 0.

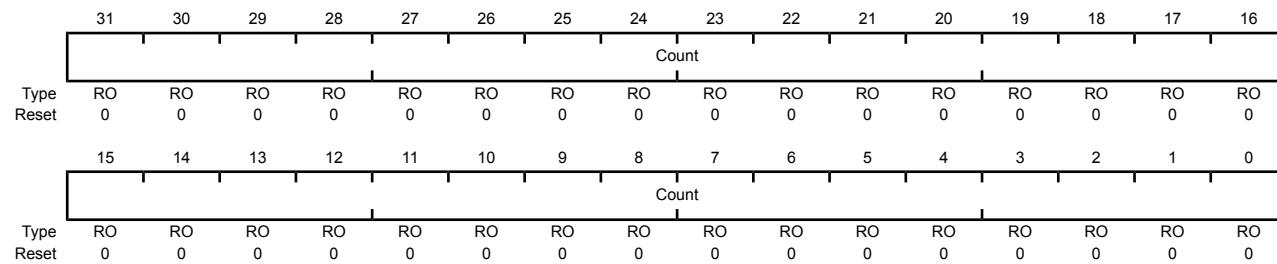
### QEI Velocity Counter (QEICOUNT)

QEI0 base: 0x4002.C000

QEI1 base: 0x4002.D000

Offset 0x018

Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	Count	RO	0x00	Velocity Pulse Count The running total of encoder pulses during this velocity timer period.

## Register 8: QEI Velocity (QEISPEED), offset 0x01C

This register contains the most recently measured velocity of the quadrature encoder. This corresponds to the number of velocity pulses counted in the previous velocity timer period. This register does not update when VelEn in QEICCTL is 0.

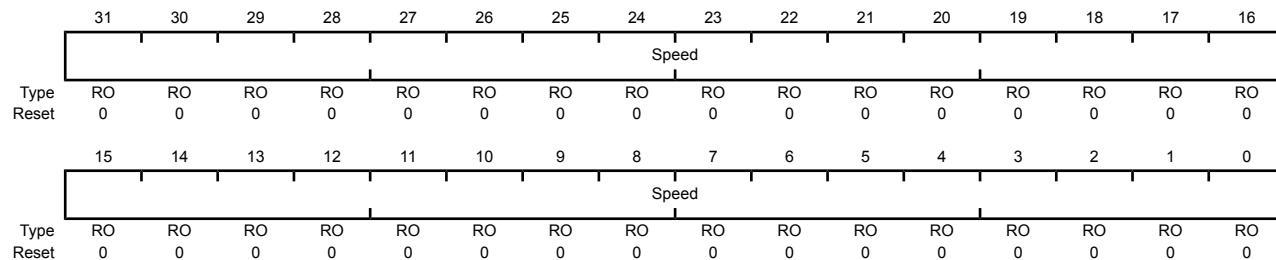
### QEI Velocity (QEISPEED)

QEI0 base: 0x4002.C000

QEI1 base: 0x4002.D000

Offset 0x01C

Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	Speed	RO	0x00	Velocity

The measured speed of the quadrature encoder in pulses per period.

## Register 9: QEI Interrupt Enable (QEINTEN), offset 0x020

This register contains enables for each of the QEI module's interrupts. An interrupt is asserted to the controller if its corresponding bit in this register is set to 1.

### QEI Interrupt Enable (QEINTEN)

QEI0 base: 0x4002.C000  
 QEI1 base: 0x4002.D000  
 Offset 0x020  
 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											
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										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
IntError IntDir IntTimer IntIndex																

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	IntError	R/W	0	Phase Error Interrupt Enable  When 1, an interrupt occurs when a phase error is detected.
2	IntDir	R/W	0	Direction Change Interrupt Enable  When 1, an interrupt occurs when the direction changes.
1	IntTimer	R/W	0	Timer Expires Interrupt Enable  When 1, an interrupt occurs when the velocity timer expires.
0	IntIndex	R/W	0	Index Pulse Detected Interrupt Enable  When 1, an interrupt occurs when the index pulse is detected.

## Register 10: QEI Raw Interrupt Status (QEIRIS), offset 0x024

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 (this is set through the **QEINTEN** register). Bits set to 1 indicate the latched events that have occurred; a zero bit indicates that the event in question has not occurred.

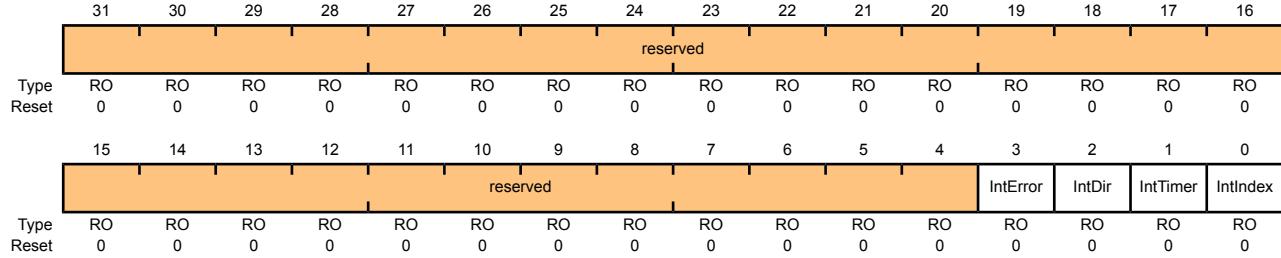
### QEI Raw Interrupt Status (QEIRIS)

QEI0 base: 0x4002.C000

QEI1 base: 0x4002.D000

Offset 0x024

Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	IntError	RO	0	Phase Error Detected Indicates that a phase error was detected.
2	IntDir	RO	0	Direction Change Detected Indicates that the direction has changed.
1	IntTimer	RO	0	Velocity Timer Expired Indicates that the velocity timer has expired.
0	IntIndex	RO	0	Index Pulse Asserted Indicates that the index pulse has occurred.

## Register 11: QEI Interrupt Status and Clear (QEIIISC), offset 0x028

This register provides the current set of interrupt sources that are asserted to the controller. Bits set to 1 indicate the latched events that have occurred; a zero bit indicates that the event in question has not occurred. This is a R/W1C register; writing a 1 to a bit position clears the corresponding interrupt reason.

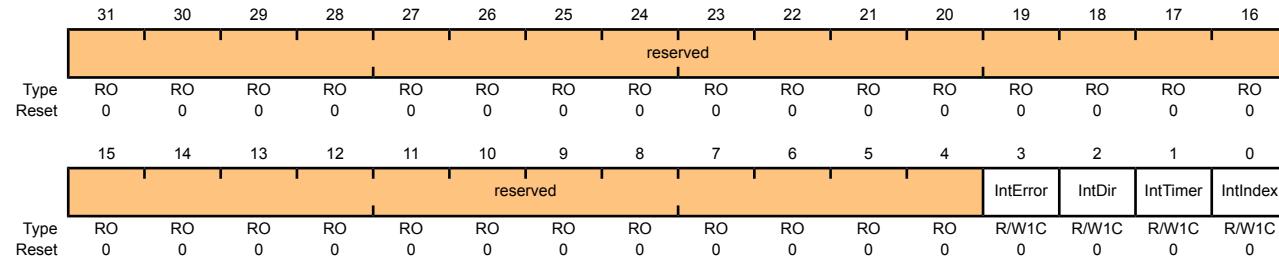
### QEI Interrupt Status and Clear (QEIIISC)

QEI0 base: 0x4002.C000

QEI1 base: 0x4002.D000

Offset 0x028

Type R/W1C, reset 0x0000.0000

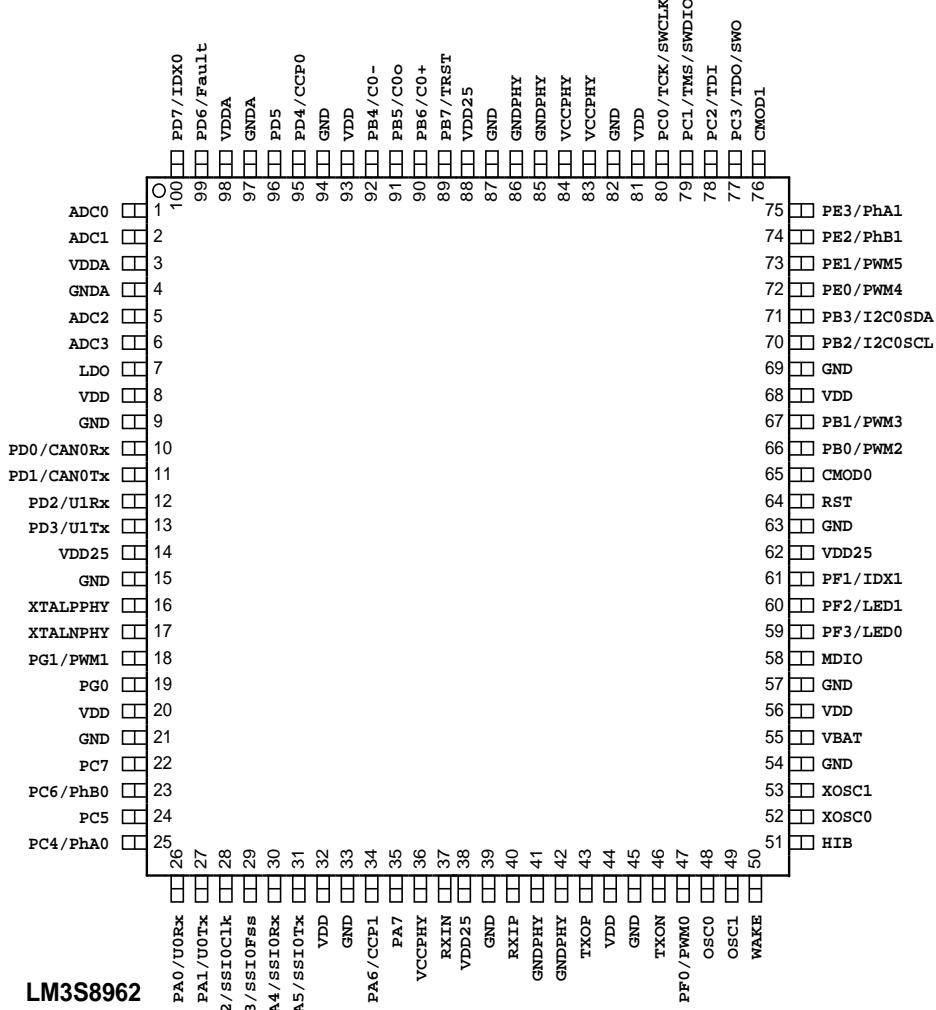


Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	IntError	R/W1C	0	Phase Error Interrupt Indicates that a phase error was detected.
2	IntDir	R/W1C	0	Direction Change Interrupt Indicates that the direction has changed.
1	IntTimer	R/W1C	0	Velocity Timer Expired Interrupt Indicates that the velocity timer has expired.
0	IntIndex	R/W1C	0	Index Pulse Interrupt Indicates that the index pulse has occurred.

## 21 Pin Diagram

Figure 21-1 on page 561 shows the pin diagram and pin-to-signal-name mapping.

**Figure 21-1. Pin Connection Diagram**



## 22 Signal Tables

The following tables list the signals available for each pin. Functionality is enabled by software with the **GPIOAFSEL** register.

**Important:** All multiplexed pins are GPIOs by default, with the exception of the five JTAG pins (**PB7** and **PC[3:0]**) which default to the JTAG functionality.

Table 22-1 on page 562 shows the pin-to-signal-name mapping, including functional characteristics of the signals. Table 22-2 on page 566 lists the signals in alphabetical order by signal name.

Table 22-3 on page 570 groups the signals by functionality, except for GPIOs. Table 22-4 on page 574 lists the GPIO pins and their alternate functionality.

**Table 22-1. Signals by Pin Number**

Pin Number	Pin Name	Pin Type	Buffer Type	Description
1	ADC0	I	Analog	Analog-to-digital converter input 0.
2	ADC1	I	Analog	Analog-to-digital converter input 1.
3	VDDA	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions.
4	GNDA	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
5	ADC2	I	Analog	Analog-to-digital converter input 2.
6	ADC3	I	Analog	Analog-to-digital converter input 3.
7	LDO	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 $\mu$ F or greater. When the on-chip LDO is used to provide power to the logic, the LDO pin must also be connected to the VDD25 pins at the board level in addition to the decoupling capacitor(s).
8	VDD	-	Power	Positive supply for I/O and some logic.
9	GND	-	Power	Ground reference for logic and I/O pins.
10	PD0	I/O	TTL	GPIO port D bit 0
	CAN0Rx	I	TTL	CAN module 0 receive
11	PD1	I/O	TTL	GPIO port D bit 1
	CAN0Tx	O	TTL	CAN module 0 transmit
12	PD2	I/O	TTL	GPIO port D bit 2
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
13	PD3	I/O	TTL	GPIO port D bit 3
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.

Pin Number	Pin Name	Pin Type	Buffer Type	Description
14	VDD25	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
15	GND	-	Power	Ground reference for logic and I/O pins.
16	XTALPPHY	O	TTL	XTALP of the Ethernet PHY
17	XTALNPHY	I	TTL	XTALN of the Ethernet PHY
18	PG1	I/O	TTL	GPIO port G bit 1
	PWM1	O	TTL	PWM 1
19	PG0	I/O	TTL	GPIO port G bit 0
20	VDD	-	Power	Positive supply for I/O and some logic.
21	GND	-	Power	Ground reference for logic and I/O pins.
22	PC7	I/O	TTL	GPIO port C bit 7
23	PC6	I/O	TTL	GPIO port C bit 6
	PhB0	I	TTL	QEI module 0 Phase B
24	PC5	I/O	TTL	GPIO port C bit 5
25	PC4	I/O	TTL	GPIO port C bit 4
	PhA0	I	TTL	QEI module 0 Phase A
26	PA0	I/O	TTL	GPIO port A bit 0
	U0Rx	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
27	PA1	I/O	TTL	GPIO port A bit 1
	U0Tx	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
28	PA2	I/O	TTL	GPIO port A bit 2
	SSI0Clk	I/O	TTL	SSI module 0 clock
29	PA3	I/O	TTL	GPIO port A bit 3
	SSI0Fss	I/O	TTL	SSI module 0 frame
30	PA4	I/O	TTL	GPIO port A bit 4
	SSI0Rx	I	TTL	SSI module 0 receive
31	PA5	I/O	TTL	GPIO port A bit 5
	SSI0Tx	O	TTL	SSI module 0 transmit
32	VDD	-	Power	Positive supply for I/O and some logic.
33	GND	-	Power	Ground reference for logic and I/O pins.
34	PA6	I/O	TTL	GPIO port A bit 6
	CCP1	I/O	TTL	Capture/Compare/PWM 1
35	PA7	I/O	TTL	GPIO port A bit 7
36	VCCPHY	I	TTL	VCC of the Ethernet PHY
37	RXIN	I	Analog	RXIN of the Ethernet PHY
38	VDD25	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
39	GND	-	Power	Ground reference for logic and I/O pins.
40	RXIP	I	Analog	RXIP of the Ethernet PHY
41	GNDPHY	I	TTL	GND of the Ethernet PHY

Pin Number	Pin Name	Pin Type	Buffer Type	Description
42	GNDPHY	I	TTL	GND of the Ethernet PHY
43	TXOP	O	Analog	TXOP of the Ethernet PHY
44	VDD	-	Power	Positive supply for I/O and some logic.
45	GND	-	Power	Ground reference for logic and I/O pins.
46	TXON	O	Analog	TXON of the Ethernet PHY
47	PF0	I/O	TTL	GPIO port F bit 0
	PWM0	O	TTL	PWM 0
48	OSC0	I	Analog	Main oscillator crystal input or an external clock reference input.
49	OSC1	I	Analog	Main oscillator crystal output.
50	WAKE	I	OD	An external input that brings the processor out of hibernate mode when asserted.
51	HIB	O	TTL	An output that indicates the processor is in hibernate mode.
52	XOSC0	I	Analog	Hibernation Module oscillator crystal input or an external clock reference input. Note that this is either a 4.19-MHz crystal or a 32.768-kHz oscillator for the Hibernation Module RTC. See the CLKSEL bit in the <b>HIBCTL</b> register.
53	XOSC1	I	Analog	Hibernation Module oscillator crystal output.
54	GND	-	Power	Ground reference for logic and I/O pins.
55	VBAT	-	Power	Power source for the Hibernation Module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation Module power-source supply.
56	VDD	-	Power	Positive supply for I/O and some logic.
57	GND	-	Power	Ground reference for logic and I/O pins.
58	MDIO	I/O	TTL	MDIO of the Ethernet PHY
59	PF3	I/O	TTL	GPIO port F bit 3
	LED0	O	TTL	MII LED 0
60	PF2	I/O	TTL	GPIO port F bit 2
	LED1	O	TTL	MII LED 1
61	PF1	I/O	TTL	GPIO port F bit 1
	IDX1	I	TTL	QEI module 1 index
62	VDD25	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
63	GND	-	Power	Ground reference for logic and I/O pins.
64	RST	I	TTL	System reset input.
65	CMOD0	I/O	TTL	CPU Mode bit 0. Input must be set to logic 0 (grounded); other encodings reserved.
66	PB0	I/O	TTL	GPIO port B bit 0
	PWM2	O	TTL	PWM 2
67	PB1	I/O	TTL	GPIO port B bit 1
	PWM3	O	TTL	PWM 3

Pin Number	Pin Name	Pin Type	Buffer Type	Description
68	VDD	-	Power	Positive supply for I/O and some logic.
69	GND	-	Power	Ground reference for logic and I/O pins.
70	PB2	I/O	TTL	GPIO port B bit 2
	I2C0SCL	I/O	OD	I2C module 0 clock
71	PB3	I/O	TTL	GPIO port B bit 3
	I2C0SDA	I/O	OD	I2C module 0 data
72	PE0	I/O	TTL	GPIO port E bit 0
	PWM4	O	TTL	PWM 4
73	PE1	I/O	TTL	GPIO port E bit 1
	PWM5	O	TTL	PWM 5
74	PE2	I/O	TTL	GPIO port E bit 2
	PhB1	I	TTL	QEI module 1 Phase B
75	PE3	I/O	TTL	GPIO port E bit 3
	PhA1	I	TTL	QEI module 1 Phase A
76	CMOD1	I/O	TTL	CPU Mode bit 1. Input must be set to logic 0 (grounded); other encodings reserved.
77	PC3	I/O	TTL	GPIO port C bit 3
	TDO	O	TTL	JTAG TDO and SWO
	SWO	O	TTL	JTAG TDO and SWO
78	PC2	I/O	TTL	GPIO port C bit 2
	TDI	I	TTL	JTAG TDI
79	PC1	I/O	TTL	GPIO port C bit 1
	TMS	I/O	TTL	JTAG TMS and SWDIO
	SWDIO	I/O	TTL	JTAG TMS and SWDIO
80	PC0	I/O	TTL	GPIO port C bit 0
	TCK	I	TTL	JTAG/SWD CLK
	SWCLK	I	TTL	JTAG/SWD CLK
81	VDD	-	Power	Positive supply for I/O and some logic.
82	GND	-	Power	Ground reference for logic and I/O pins.
83	VCCPHY	I	TTL	VCC of the Ethernet PHY
84	VCCPHY	I	TTL	VCC of the Ethernet PHY
85	GNDPHY	I	TTL	GND of the Ethernet PHY
86	GNDPHY	I	TTL	GND of the Ethernet PHY
87	GND	-	Power	Ground reference for logic and I/O pins.
88	VDD25	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
89	PB7	I/O	TTL	GPIO port B bit 7
	TRST	I	TTL	JTAG TRSTn
90	PB6	I/O	TTL	GPIO port B bit 6
	C0+	I	Analog	Analog comparator 0 positive input
91	PB5	I/O	TTL	GPIO port B bit 5
	C0o	O	TTL	Analog comparator 0 output

Pin Number	Pin Name	Pin Type	Buffer Type	Description
92	PB4	I/O	TTL	GPIO port B bit 4
	C0-	I	Analog	Analog comparator 0 negative input
93	VDD	-	Power	Positive supply for I/O and some logic.
94	GND	-	Power	Ground reference for logic and I/O pins.
95	PD4	I/O	TTL	GPIO port D bit 4
	CCP0	I/O	TTL	Capture/Compare/PWM 0
96	PD5	I/O	TTL	GPIO port D bit 5
97	GNDA	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
98	VDDA	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions.
99	PD6	I/O	TTL	GPIO port D bit 6
	Fault	I	TTL	PWM Fault
100	PD7	I/O	TTL	GPIO port D bit 7
	IDX0	I	TTL	QEI module 0 index

**Table 22-2. Signals by Signal Name**

Pin Name	Pin Number	Pin Type	Buffer Type	Description
ADC0	1	I	Analog	Analog-to-digital converter input 0.
ADC1	2	I	Analog	Analog-to-digital converter input 1.
ADC2	5	I	Analog	Analog-to-digital converter input 2.
ADC3	6	I	Analog	Analog-to-digital converter input 3.
C0+	90	I	Analog	Analog comparator 0 positive input
C0-	92	I	Analog	Analog comparator 0 negative input
C0o	91	O	TTL	Analog comparator 0 output
CAN0Rx	10	I	TTL	CAN module 0 receive
CAN0Tx	11	O	TTL	CAN module 0 transmit
CCP0	95	I/O	TTL	Capture/Compare/PWM 0
CCP1	34	I/O	TTL	Capture/Compare/PWM 1
CMOD0	65	I/O	TTL	CPU Mode bit 0. Input must be set to logic 0 (grounded); other encodings reserved.
CMOD1	76	I/O	TTL	CPU Mode bit 1. Input must be set to logic 0 (grounded); other encodings reserved.
Fault	99	I	TTL	PWM Fault
GND	9	-	Power	Ground reference for logic and I/O pins.
GND	15	-	Power	Ground reference for logic and I/O pins.
GND	21	-	Power	Ground reference for logic and I/O pins.
GND	33	-	Power	Ground reference for logic and I/O pins.
GND	39	-	Power	Ground reference for logic and I/O pins.

Pin Name	Pin Number	Pin Type	Buffer Type	Description
GND	45	-	Power	Ground reference for logic and I/O pins.
GND	54	-	Power	Ground reference for logic and I/O pins.
GND	57	-	Power	Ground reference for logic and I/O pins.
GND	63	-	Power	Ground reference for logic and I/O pins.
GND	69	-	Power	Ground reference for logic and I/O pins.
GND	82	-	Power	Ground reference for logic and I/O pins.
GND	87	-	Power	Ground reference for logic and I/O pins.
GND	94	-	Power	Ground reference for logic and I/O pins.
GNDA	4	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
GNDA	97	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
GNDPHY	41	I	TTL	GND of the Ethernet PHY
GNDPHY	42	I	TTL	GND of the Ethernet PHY
GNDPHY	85	I	TTL	GND of the Ethernet PHY
GNDPHY	86	I	TTL	GND of the Ethernet PHY
HIB	51	O	TTL	An output that indicates the processor is in hibernate mode.
I2C0SCL	70	I/O	OD	I2C module 0 clock
I2C0SDA	71	I/O	OD	I2C module 0 data
IDX0	100	I	TTL	QEI module 0 index
IDX1	61	I	TTL	QEI module 1 index
LDO	7	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 $\mu$ F or greater. When the on-chip LDO is used to provide power to the logic, the LDO pin must also be connected to the VDD25 pins at the board level in addition to the decoupling capacitor(s).
LED0	59	O	TTL	MII LED 0
LED1	60	O	TTL	MII LED 1
MDIO	58	I/O	TTL	MDIO of the Ethernet PHY
OSC0	48	I	Analog	Main oscillator crystal input or an external clock reference input.
OSC1	49	I	Analog	Main oscillator crystal output.
PA0	26	I/O	TTL	GPIO port A bit 0
PA1	27	I/O	TTL	GPIO port A bit 1
PA2	28	I/O	TTL	GPIO port A bit 2
PA3	29	I/O	TTL	GPIO port A bit 3
PA4	30	I/O	TTL	GPIO port A bit 4
PA5	31	I/O	TTL	GPIO port A bit 5

Pin Name	Pin Number	Pin Type	Buffer Type	Description
PA6	34	I/O	TTL	GPIO port A bit 6
PA7	35	I/O	TTL	GPIO port A bit 7
PB0	66	I/O	TTL	GPIO port B bit 0
PB1	67	I/O	TTL	GPIO port B bit 1
PB2	70	I/O	TTL	GPIO port B bit 2
PB3	71	I/O	TTL	GPIO port B bit 3
PB4	92	I/O	TTL	GPIO port B bit 4
PB5	91	I/O	TTL	GPIO port B bit 5
PB6	90	I/O	TTL	GPIO port B bit 6
PB7	89	I/O	TTL	GPIO port B bit 7
PC0	80	I/O	TTL	GPIO port C bit 0
PC1	79	I/O	TTL	GPIO port C bit 1
PC2	78	I/O	TTL	GPIO port C bit 2
PC3	77	I/O	TTL	GPIO port C bit 3
PC4	25	I/O	TTL	GPIO port C bit 4
PC5	24	I/O	TTL	GPIO port C bit 5
PC6	23	I/O	TTL	GPIO port C bit 6
PC7	22	I/O	TTL	GPIO port C bit 7
PD0	10	I/O	TTL	GPIO port D bit 0
PD1	11	I/O	TTL	GPIO port D bit 1
PD2	12	I/O	TTL	GPIO port D bit 2
PD3	13	I/O	TTL	GPIO port D bit 3
PD4	95	I/O	TTL	GPIO port D bit 4
PD5	96	I/O	TTL	GPIO port D bit 5
PD6	99	I/O	TTL	GPIO port D bit 6
PD7	100	I/O	TTL	GPIO port D bit 7
PE0	72	I/O	TTL	GPIO port E bit 0
PE1	73	I/O	TTL	GPIO port E bit 1
PE2	74	I/O	TTL	GPIO port E bit 2
PE3	75	I/O	TTL	GPIO port E bit 3
PF0	47	I/O	TTL	GPIO port F bit 0
PF1	61	I/O	TTL	GPIO port F bit 1
PF2	60	I/O	TTL	GPIO port F bit 2
PF3	59	I/O	TTL	GPIO port F bit 3
PG0	19	I/O	TTL	GPIO port G bit 0
PG1	18	I/O	TTL	GPIO port G bit 1
PhA0	25	I	TTL	QEI module 0 Phase A
PhA1	75	I	TTL	QEI module 1 Phase A
PhB0	23	I	TTL	QEI module 0 Phase B
PhB1	74	I	TTL	QEI module 1 Phase B
PWM0	47	O	TTL	PWM 0
PWM1	18	O	TTL	PWM 1

Pin Name	Pin Number	Pin Type	Buffer Type	Description
PWM2	66	O	TTL	PWM 2
PWM3	67	O	TTL	PWM 3
PWM4	72	O	TTL	PWM 4
PWM5	73	O	TTL	PWM 5
RST	64	I	TTL	System reset input.
RXIN	37	I	Analog	RXIN of the Ethernet PHY
RXIP	40	I	Analog	RXIP of the Ethernet PHY
SSI0Clk	28	I/O	TTL	SSI module 0 clock
SSI0Fss	29	I/O	TTL	SSI module 0 frame
SSI0Rx	30	I	TTL	SSI module 0 receive
SSI0Tx	31	O	TTL	SSI module 0 transmit
SWCLK	80	I	TTL	JTAG/SWD CLK
SWDIO	79	I/O	TTL	JTAG TMS and SWDIO
SWO	77	O	TTL	JTAG TDO and SWO
TCK	80	I	TTL	JTAG/SWD CLK
TDI	78	I	TTL	JTAG TDI
TDO	77	O	TTL	JTAG TDO and SWO
TMS	79	I/O	TTL	JTAG TMS and SWDIO
TRST	89	I	TTL	JTAG TRSTn
TXON	46	O	Analog	TXON of the Ethernet PHY
TXOP	43	O	Analog	TXOP of the Ethernet PHY
U0Rx	26	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
U0Tx	27	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
U1Rx	12	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
U1Tx	13	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
VBAT	55	-	Power	Power source for the Hibernation Module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation Module power-source supply.
VCCPHY	36	I	TTL	VCC of the Ethernet PHY
VCCPHY	83	I	TTL	VCC of the Ethernet PHY
VCCPHY	84	I	TTL	VCC of the Ethernet PHY
VDD	8	-	Power	Positive supply for I/O and some logic.
VDD	20	-	Power	Positive supply for I/O and some logic.
VDD	32	-	Power	Positive supply for I/O and some logic.
VDD	44	-	Power	Positive supply for I/O and some logic.
VDD	56	-	Power	Positive supply for I/O and some logic.
VDD	68	-	Power	Positive supply for I/O and some logic.
VDD	81	-	Power	Positive supply for I/O and some logic.
VDD	93	-	Power	Positive supply for I/O and some logic.

Pin Name	Pin Number	Pin Type	Buffer Type	Description
VDD25	14	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VDD25	38	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VDD25	62	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VDD25	88	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VDDA	3	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions.
VDDA	98	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions.
WAKE	50	I	OD	An external input that brings the processor out of hibernate mode when asserted.
XOSC0	52	I	Analog	Hibernation Module oscillator crystal input or an external clock reference input. Note that this is either a 4.19-MHz crystal or a 32.768-kHz oscillator for the Hibernation Module RTC. See the CLKSEL bit in the <b>HIBCTL</b> register.
XOSC1	53	I	Analog	Hibernation Module oscillator crystal output.
XTALNPHY	17	I	TTL	XTALN of the Ethernet PHY
XTALPPHY	16	O	TTL	XTALP of the Ethernet PHY

Table 22-3. Signals by Function, Except for GPIO

Function	Pin Name	Pin Number	Pin Type	Buffer Type	Description
ADC	ADC0	1	I	Analog	Analog-to-digital converter input 0.
	ADC1	2	I	Analog	Analog-to-digital converter input 1.
	ADC2	5	I	Analog	Analog-to-digital converter input 2.
	ADC3	6	I	Analog	Analog-to-digital converter input 3.
Analog Comparators	C0+	90	I	Analog	Analog comparator 0 positive input
	C0-	92	I	Analog	Analog comparator 0 negative input
	C0o	91	O	TTL	Analog comparator 0 output
Controller Area Network	CAN0Rx	10	I	TTL	CAN module 0 receive
	CAN0Tx	11	O	TTL	CAN module 0 transmit

Function	Pin Name	Pin Number	Pin Type	Buffer Type	Description
Ethernet PHY	GNDPHY	41	I	TTL	GND of the Ethernet PHY
	GNDPHY	42	I	TTL	GND of the Ethernet PHY
	GNDPHY	85	I	TTL	GND of the Ethernet PHY
	GNDPHY	86	I	TTL	GND of the Ethernet PHY
	LED0	59	O	TTL	MII LED 0
	LED1	60	O	TTL	MII LED 1
	MDIO	58	I/O	TTL	MDIO of the Ethernet PHY
	RXIN	37	I	Analog	RXIN of the Ethernet PHY
	RXIP	40	I	Analog	RXIP of the Ethernet PHY
	TXON	46	O	Analog	TXON of the Ethernet PHY
	TXOP	43	O	Analog	TXOP of the Ethernet PHY
	VCCPHY	36	I	TTL	VCC of the Ethernet PHY
	VCCPHY	83	I	TTL	VCC of the Ethernet PHY
	VCCPHY	84	I	TTL	VCC of the Ethernet PHY
General-Purpose Timers	XTALNPHY	17	I	TTL	XTALN of the Ethernet PHY
	XTALPPHY	16	O	TTL	XTALP of the Ethernet PHY
I2C	CCP0	95	I/O	TTL	Capture/Compare/PWM 0
	CCP1	34	I/O	TTL	Capture/Compare/PWM 1
JTAG/SWD/SWO	I2C0SCL	70	I/O	OD	I2C module 0 clock
	I2C0SDA	71	I/O	OD	I2C module 0 data
	SWCLK	80	I	TTL	JTAG/SWD CLK
	SWDIO	79	I/O	TTL	JTAG TMS and SWDIO
	SWO	77	O	TTL	JTAG TDO and SWO
	TCK	80	I	TTL	JTAG/SWD CLK
	TDI	78	I	TTL	JTAG TDI
PWM	TDO	77	O	TTL	JTAG TDO and SWO
	TMS	79	I/O	TTL	JTAG TMS and SWDIO
	Fault	99	I	TTL	PWM Fault
	PWM0	47	O	TTL	PWM 0
	PWM1	18	O	TTL	PWM 1
	PWM2	66	O	TTL	PWM 2
	PWM3	67	O	TTL	PWM 3
ADC	PWM4	72	O	TTL	PWM 4
	PWM5	73	O	TTL	PWM 5

Function	Pin Name	Pin Number	Pin Type	Buffer Type	Description
Power	GND	9	-	Power	Ground reference for logic and I/O pins.
	GND	15	-	Power	Ground reference for logic and I/O pins.
	GND	21	-	Power	Ground reference for logic and I/O pins.
	GND	33	-	Power	Ground reference for logic and I/O pins.
	GND	39	-	Power	Ground reference for logic and I/O pins.
	GND	45	-	Power	Ground reference for logic and I/O pins.
	GND	54	-	Power	Ground reference for logic and I/O pins.
	GND	57	-	Power	Ground reference for logic and I/O pins.
	GND	63	-	Power	Ground reference for logic and I/O pins.
	GND	69	-	Power	Ground reference for logic and I/O pins.
	GND	82	-	Power	Ground reference for logic and I/O pins.
	GND	87	-	Power	Ground reference for logic and I/O pins.
	GND	94	-	Power	Ground reference for logic and I/O pins.
	GNDA	4	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
	GNDA	97	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
	HIB	51	O	TTL	An output that indicates the processor is in hibernate mode.
	LDO	7	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 $\mu$ F or greater. When the on-chip LDO is used to provide power to the logic, the LDO pin must also be connected to the VDD25 pins at the board level in addition to the decoupling capacitor(s).
	VBAT	55	-	Power	Power source for the Hibernation Module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation Module power-source supply.
	VDD	8	-	Power	Positive supply for I/O and some logic.
	VDD	20	-	Power	Positive supply for I/O and some logic.
	VDD	32	-	Power	Positive supply for I/O and some logic.
	VDD	44	-	Power	Positive supply for I/O and some logic.
	VDD	56	-	Power	Positive supply for I/O and some logic.
	VDD	68	-	Power	Positive supply for I/O and some logic.
	VDD	81	-	Power	Positive supply for I/O and some logic.
	VDD	93	-	Power	Positive supply for I/O and some logic.
	VDD25	14	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
	VDD25	38	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
	VDD25	62	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.

Function	Pin Name	Pin Number	Pin Type	Buffer Type	Description
	VDD25	88	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
	VDDA	3	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions.
	VDDA	98	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions.
	WAKE	50	I	OD	An external input that brings the processor out of hibernate mode when asserted.
QEI	IDX0	100	I	TTL	QEI module 0 index
	IDX1	61	I	TTL	QEI module 1 index
	PhA0	25	I	TTL	QEI module 0 Phase A
	PhA1	75	I	TTL	QEI module 1 Phase A
	PhB0	23	I	TTL	QEI module 0 Phase B
	PhB1	74	I	TTL	QEI module 1 Phase B
SSI	SSI0Clk	28	I/O	TTL	SSI module 0 clock
	SSI0Fss	29	I/O	TTL	SSI module 0 frame
	SSI0Rx	30	I	TTL	SSI module 0 receive
	SSI0Tx	31	O	TTL	SSI module 0 transmit
System Control & Clocks	CMODO	65	I/O	TTL	CPU Mode bit 0. Input must be set to logic 0 (grounded); other encodings reserved.
	CMOD1	76	I/O	TTL	CPU Mode bit 1. Input must be set to logic 0 (grounded); other encodings reserved.
	OSC0	48	I	Analog	Main oscillator crystal input or an external clock reference input.
	OSC1	49	I	Analog	Main oscillator crystal output.
	RST	64	I	TTL	System reset input.
	TRST	89	I	TTL	JTAG TRSTn
	XOSC0	52	I	Analog	Hibernation Module oscillator crystal input or an external clock reference input. Note that this is either a 4.19-MHz crystal or a 32.768-kHz oscillator for the Hibernation Module RTC. See the CLKSEL bit in the HIBCTL register.
UART	U0Rx	26	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
	U0Tx	27	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
	U1Rx	12	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	U1Tx	13	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.

**Table 22-4. GPIO Pins and Alternate Functions**

GPIO Pin	Pin Number	Multiplexed Function	Multiplexed Function
PA0	26	U0Rx	
PA1	27	U0Tx	
PA2	28	SSI0Clk	
PA3	29	SSI0Fss	
PA4	30	SSI0Rx	
PA5	31	SSI0Tx	
PA6	34	CCP1	
PA7	35		
PB0	66	PWM2	
PB1	67	PWM3	
PB2	70	I2C0SCL	
PB3	71	I2C0SDA	
PB4	92	C0-	
PB5	91	C0o	
PB6	90	C0+	
PB7	89	TRST	
PC0	80	TCK	SWCLK
PC1	79	TMS	SWDIO
PC2	78	TDI	
PC3	77	TDO	SWO
PC4	25	PhA0	
PC5	24		
PC6	23	PhB0	
PC7	22		
PD0	10	CAN0Rx	
PD1	11	CAN0Tx	
PD2	12	U1Rx	
PD3	13	U1Tx	
PD4	95	CCP0	
PD5	96		
PD6	99	Fault	
PD7	100	IDX0	
PE0	72	PWM4	
PE1	73	PWM5	
PE2	74	PhB1	
PE3	75	PhA1	
PF0	47	PWM0	
PF1	61	IDX1	
PF2	60	LED1	
PF3	59	LED0	
PG0	19		

GPIO Pin	Pin Number	Multiplexed Function	Multiplexed Function
PG1	18	PWM1	

## 23 Operating Characteristics

**Table 23-1. Temperature Characteristics**

Characteristic	Symbol	Value	Unit
Operating temperature range <sup>a</sup>	T <sub>A</sub>	-40 to +85	°C

a. Maximum storage temperature is 150°C.

**Table 23-2. Thermal Characteristics**

Characteristic	Symbol	Value	Unit
Thermal resistance (junction to ambient) <sup>a</sup>	Θ <sub>JA</sub>	55.3	°C/W
Average junction temperature <sup>b</sup>	T <sub>J</sub>	T <sub>A</sub> + (P <sub>AVG</sub> • Θ <sub>JA</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.

## 24 Electrical Characteristics

### 24.1 DC Characteristics

#### 24.1.1 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 24-1. Maximum Ratings**

Characteristic <sup>a</sup>	Symbol	Value		Unit
		Min	Max	
I/O supply voltage ( $V_{DD}$ )	$V_{DD}$	0	4	V
Core supply voltage ( $V_{DD25}$ )	$V_{DD25}$	0	4	V
Analog supply voltage ( $V_{DDA}$ )	$V_{DDA}$	0	4	V
Battery supply voltage ( $V_{BAT}$ )	$V_{BAT}$	0	4	V
Ethernet PHY supply voltage ( $V_{CCPHY}$ )	$V_{CCPHY}$	0	4	V
Input voltage	$V_{IN}$	-0.3	5.5	V
Maximum current per output pins	I	-	25	mA

a. Voltages are measured with respect to GND.

**Important:** This device contains circuitry to protect the inputs against damage due to high-static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (for example, either GND or  $V_{DD}$ ).

#### 24.1.2 Recommended DC Operating Conditions

**Table 24-2. Recommended DC Operating Conditions**

Parameter	Parameter Name	Min	Nom	Max	Unit
$V_{DD}$	I/O supply voltage	3.0	3.3	3.6	V
$V_{DD25}$	Core supply voltage	2.25	2.5	2.75	V
$V_{DDA}$	Analog supply voltage	3.0	3.3	3.6	V
$V_{BAT}$	Battery supply voltage	2.3	3.0	3.6	V
$V_{CCPHY}$	Ethernet PHY 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
$V_{SIH}$	High-level input voltage for Schmitt trigger inputs	$0.8 * V_{DD}$	-	$V_{DD}$	V
$V_{SIL}$	Low-level input voltage for Schmitt trigger inputs	0	-	$0.2 * V_{DD}$	V
$V_{OH}$	High-level output voltage	2.4	-	-	V
$V_{OL}$	Low-level output voltage	-	-	0.4	V

Parameter	Parameter Name	Min	Nom	Max	Unit
$I_{OH}$	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

### 24.1.3 On-Chip Low Drop-Out (LDO) Regulator Characteristics

Table 24-3. LDO Regulator Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
$V_{LDOOUT}$	Programmable internal (logic) power supply output value	2.25	2.5	2.75	V
	Output voltage accuracy	-	2%	-	%
$t_{PON}$	Power-on time	-	-	100	$\mu\text{s}$
$t_{ON}$	Time on	-	-	200	$\mu\text{s}$
$t_{OFF}$	Time off	-	-	100	$\mu\text{s}$
$V_{STEP}$	Step programming incremental voltage	-	50	-	mV
$C_{LDO}$	External filter capacitor size for internal power supply	1.0	-	3.0	$\mu\text{F}$

### 24.1.4 Power Specifications

The power measurements specified in the tables that follow are run on the core processor using SRAM with the following specifications (except as noted):

- $V_{DD} = 3.3\text{ V}$
- $V_{DD25} = 2.50\text{ V}$
- $V_{BAT} = 3.0\text{ V}$
- $V_{DDA} = 3.3\text{ V}$
- $V_{DDPHY} = 3.3\text{ V}$
- Temperature =  $25^\circ\text{C}$
- Clock Source (MOSC) = 3.579545 MHz Crystal Oscillator
- Main oscillator (MOSC) = enabled
- Internal oscillator (IOSC) = disabled

**Table 24-4. Detailed Power Specifications**

Parameter	Parameter Name	Conditions	3.3 V $V_{DD}$ , $V_{DDA}$ , $V_{DDPHY}$		2.5 V $V_{DD25}$		3.0 V $V_{BAT}$		Unit
			Nom	Max	Nom	Max	Nom	Max	
$I_{DD\_RUN}$	Run mode 1 (Flash loop)	$V_{DD25} = 2.50$ V Code= while(1){} executed in Flash Peripherals = All ON System Clock = 50 MHz (with PLL)	48	pending <sup>a</sup>	108	pending <sup>a</sup>	0	pending <sup>a</sup>	mA
	Run mode 2 (Flash loop)	$V_{DD25} = 2.50$ V Code= while(1){} executed in Flash Peripherals = All OFF System Clock = 50 MHz (with PLL)	5	pending <sup>a</sup>	52	pending <sup>a</sup>	0	pending <sup>a</sup>	mA
	Run mode 1 (SRAM loop)	$V_{DD25} = 2.50$ V Code= while(1){} executed in SRAM Peripherals = All ON System Clock = 50 MHz (with PLL)	48	pending <sup>a</sup>	100	pending <sup>a</sup>	0	pending <sup>a</sup>	mA
	Run mode 2 (SRAM loop)	$V_{DD25} = 2.50$ V Code= while(1){} executed in SRAM Peripherals = All OFF System Clock = 50 MHz (with PLL)	5	pending <sup>a</sup>	45	pending <sup>a</sup>	0	pending <sup>a</sup>	mA
$I_{DD\_SLEEP}$	Sleep mode	$V_{DD25} = 2.50$ V Peripherals = All OFF System Clock = 50 MHz (with PLL)	5	pending <sup>a</sup>	16	pending <sup>a</sup>	0	pending <sup>a</sup>	mA
$I_{DD\_DEEPSLEEP}$	Deep-Sleep mode	LDO = 2.25 V Peripherals = All OFF System Clock = IOSC30KHZ/64	4.6	pending <sup>a</sup>	0.21	pending <sup>a</sup>	0	pending <sup>a</sup>	mA
$I_{DD\_HIBERNATE}$	Hibernate mode	$V_{BAT} = 3.0$ V $V_{DD} = 0$ V $V_{DD25} = 0$ V $V_{DDA} = 0$ V $V_{DDPHY} = 0$ V Peripherals = All OFF System Clock = OFF Hibernate Module = 32 kHz	0	pending <sup>a</sup>	0	pending <sup>a</sup>	16	pending <sup>a</sup>	$\mu$ A

a. Pending characterization completion.

### 24.1.5 Flash Memory Characteristics

Table 24-5. 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	-	-	μs
$T_{ERASE}$	Page erase time	20	-	-	ms
$T_{ME}$	Mass erase time	200	-	-	ms

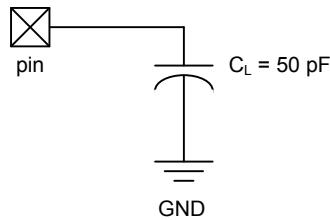
a. A program/erase cycle is defined as switching the bits from 1->0->1.

## 24.2 AC Characteristics

### 24.2.1 Load Conditions

Unless otherwise specified, the following conditions are true for all timing measurements. Timing measurements are for 4-mA drive strength.

Figure 24-1. Load Conditions



### 24.2.2 Clocks

Table 24-6. Phase Locked Loop (PLL) Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
$f_{ref\_crystal}$	Crystal reference <sup>a</sup>	3.579545	-	8.192	MHz
$f_{ref\_ext}$	External clock reference <sup>a</sup>	3.579545	-	8.192	MHz
$f_{pll}$	PLL frequency <sup>b</sup>	-	400	-	MHz
$T_{READY}$	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 24-7. Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
$f_{iosc}$	Internal 12 MHz oscillator frequency	8.4	12	15.6	MHz
$f_{iosc30KHZ}$	Internal 30 KHz oscillator frequency	21	30	39	KHz
$f_{xosc}$	Hibernation module oscillator frequency	-	4.194304	-	MHz
$f_{xosc\_xtal}$	Crystal reference for hibernation oscillator	-	4.194304	-	MHz
$f_{xosc\_ext}$	External clock reference for hibernation module	-	32.768	-	KHz

Parameter	Parameter Name	Min	Nom	Max	Unit
$f_{MOSC}$	Main oscillator frequency	1	-	8	MHz
$t_{MOSC\_per}$	Main oscillator period	125	-	1000	ns
$f_{ref\_crystal\_bypass}$ <sup>a</sup>	Crystal reference using the main oscillator (PLL in BYPASS mode)	1	-	8	MHz
$f_{ref\_ext\_bypass}$	External clock reference (PLL in BYPASS mode) <sup>a</sup>	0	-	50	MHz
$f_{system\_clock}$	System clock	0	-	50	MHz

a. The ADC must be clocked from the PLL or directly from a 14-MHz to 18-MHz clock source to operate properly.

**Table 24-8. Crystal Characteristics**

Parameter Name	Value				Units
Frequency	8	6	4	3.5	MHz
Frequency tolerance	$\pm 50$	$\pm 50$	$\pm 50$	$\pm 50$	ppm
Aging	$\pm 5$	$\pm 5$	$\pm 5$	$\pm 5$	ppm/yr
Oscillation mode	Parallel	Parallel	Parallel	Parallel	
Temperature stability (0 - 85 °C)	$\pm 25$	$\pm 25$	$\pm 25$	$\pm 25$	ppm
Motional capacitance (typ)	27.8	37.0	55.6	63.5	pF
Motional inductance (typ)	14.3	19.1	28.6	32.7	mH
Equivalent series resistance (max)	120	160	200	220	$\Omega$
Shunt capacitance (max)	10	10	10	10	pF
Load capacitance (typ)	16	16	16	16	pF
Drive level (typ)	100	100	100	100	$\mu W$

### 24.2.3 Analog-to-Digital Converter

**Table 24-9. ADC Characteristics**

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	V
	Maximum differential, full-scale analog input voltage	-	-	1.5	V
	Minimum differential, full-scale analog input voltage	-	-	-1.5	V
$C_{ADCIN}$	Equivalent input capacitance	-	1	-	pF
N	Resolution	-	10	-	bits
$f_{ADC}$	ADC internal clock frequency	7	8	9	MHz
$t_{ADCCONV}$	Conversion time	-	-	16	$t_{ADCcycles}^a$
$f_{ADCCONV}$	Conversion rate	438	500	563	k samples/s
INL	Integral nonlinearity	-	-	$\pm 1$	LSB
DNL	Differential nonlinearity	-	-	$\pm 1$	LSB
OFF	Offset	-	-	$\pm 1$	LSB
GAIN	Gain	-	-	$\pm 1$	LSB

a.  $t_{ADC} = 1/f_{ADC}$  clock

## 24.2.4 Analog Comparator

Table 24-10. 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 24-11. Analog Comparator Voltage Reference Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
$R_{HR}$	Resolution high range	-	$V_{DD}/32$	-	LSB
$R_{LR}$	Resolution low range	-	$V_{DD}/24$	-	LSB
$A_{HR}$	Absolute accuracy high range	-	-	$\pm 1/2$	LSB
$A_{LR}$	Absolute accuracy low range	-	-	$\pm 1/4$	LSB

## 24.2.5 I<sup>2</sup>C

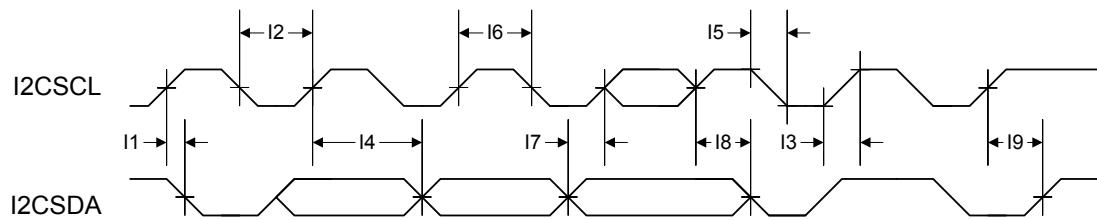
Table 24-12. I<sup>2</sup>C Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
I1 <sup>a</sup>	$t_{SCH}$	Start condition hold time	36	-	-	system clocks
I2 <sup>a</sup>	$t_{LP}$	Clock Low period	36	-	-	system clocks
I3 <sup>b</sup>	$t_{SRT}$	I <sup>2</sup> CSCL/I <sup>2</sup> CSDA rise time ( $V_{IL}=0.5$ V to $V_{IH}=2.4$ V)	-	-	(see note b)	ns
I4 <sup>a</sup>	$t_{DH}$	Data hold time	2	-	-	system clocks
I5 <sup>c</sup>	$t_{SFT}$	I <sup>2</sup> CSCL/I <sup>2</sup> CSDA fall time ( $V_{IH}=2.4$ V to $V_{IL}=0.5$ V)	-	9	10	ns
I6 <sup>a</sup>	$t_{HT}$	Clock High time	24	-	-	system clocks
I7 <sup>a</sup>	$t_{DS}$	Data setup time	18	-	-	system clocks
I8 <sup>a</sup>	$t_{SCSR}$	Start condition setup time (for repeated start condition only)	36	-	-	system clocks
I9 <sup>a</sup>	$t_{SCS}$	Stop condition setup time	24	-	-	system clocks

a. Values depend on the value programmed into the TPR bit in the **I<sup>2</sup>C Master Timer Period (I<sup>2</sup>CMTTPR)** 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 24-2. I<sup>2</sup>C Timing**

## 24.2.6 Ethernet Controller

**Table 24-13. 100BASE-TX Transmitter Characteristics<sup>a</sup>**

Parameter Name	Min	Nom	Max	Unit
Peak output amplitude	950	-	1050	mVpk
Output amplitude symmetry	0.98	-	1.02	mVpk
Output overshoot	-	-	5	%
Rise/Fall time	3	-	5	ns
Rise/Fall time imbalance	-	-	500	ps
Duty cycle distortion	-	-	-	ps
Jitter	-	-	1.4	ns

a. Measured at the line side of the transformer.

**Table 24-14. 100BASE-TX Transmitter Characteristics (informative)<sup>a</sup>**

Parameter Name	Min	Nom	Max	Unit
Return loss	16	-	-	dB
Open-circuit inductance	350	-	-	μs

a. The specifications in this table are included for information only. They are mainly a function of the external transformer and termination resistors used for measurements.

**Table 24-15. 100BASE-TX Receiver Characteristics**

Parameter Name	Min	Nom	Max	Unit
Signal detect assertion threshold	600	700	-	mVppd
Signal detect de-assertion threshold	350	425	-	mVppd
Differential input resistance	20	-	-	kΩ
Jitter tolerance (pk-pk)	4	-	-	ns
Baseline wander tracking	-75	-	+75	%
Signal detect assertion time	-	-	1000	μs
Signal detect de-assertion time	-	-	4	μs

**Table 24-16. 10BASE-T Transmitter Characteristics<sup>a</sup>**

Parameter Name	Min	Nom	Max	Unit
Peak differential output signal	2.2	-	2.8	V
Harmonic content	27	-	-	dB
Link pulse width	-	100	-	ns

Parameter Name	Min	Nom	Max	Unit
Start-of-idle pulse width	-	300 350	-	ns

a. The Manchester-encoded data pulses, the link pulse and the start-of-idle pulse are tested against the templates and using the procedures found in Clause 14 of *IEEE 802.3*.

**Table 24-17. 10BASE-T Transmitter Characteristics (informative)<sup>a</sup>**

Parameter Name	Min	Nom	Max	Unit
Output return loss	15	-	-	dB
Output impedance balance	29-17log(f/10)	-	-	dB
Peak common-mode output voltage	-	-	50	mV
Common-mode rejection	-	-	100	mV
Common-mode rejection jitter	-	-	1	ns

a. The specifications in this table are included for information only. They are mainly a function of the external transformer and termination resistors used for measurements.

**Table 24-18. 10BASE-T Receiver Characteristics**

Parameter Name	Min	Nom	Max	Unit
DLL phase acquisition time	-	10	-	BT
Jitter tolerance (pk-pk)	30	-	-	ns
Input squelched threshold	500	600	700	mVppd
Input unsquelched threshold	275	350	425	mVppd
Differential input resistance	-	20	-	kΩ
Bit error ratio	-	$10^{-10}$	-	-
Common-mode rejection	25	-	-	V

**Table 24-19. Isolation Transformers<sup>a</sup>**

Name	Value	Condition
Turns ratio	1 CT : 1 CT	+/- 5%
Open-circuit inductance	350 uH (min)	@ 10 mV, 10 kHz
Leakage inductance	0.40 uH (max)	@ 1 MHz (min)
Inter-winding capacitance	25 pF (max)	
DC resistance	0.9 Ohm (max)	
Insertion loss	0.4 dB (typ)	0-65 MHz
HIPOT	1500	Vrms

a. Two simple 1:1 isolation transformers are required at the line interface. Transformers with integrated common-mode chokes are recommended for exceeding FCC requirements. This table gives the recommended line transformer characteristics.

**Note:** The 100Base-TX amplitude specifications assume a transformer loss of 0.4 dB. For the transmit line transformer with higher insertion losses, up to 1.2 dB of insertion loss can be compensated by selecting the appropriate setting in the Transmit Amplitude Selection (TXO) bits in the **MR19** register.

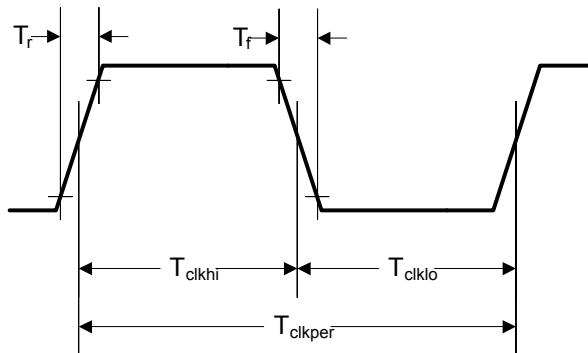
**Table 24-20. Ethernet Reference Crystal<sup>a</sup>**

Name	Value	Condition
Frequency	25.00000	MHz
Load capacitance <sup>b</sup>	4 <sup>c</sup>	pF
Frequency tolerance	±50	PPM
Aging	±2	PPM/yr
Temperature stability (0° to 70°)	±5	PPM
Oscillation mode	Parallel resonance, fundamental mode	
Parameters at 25° C ±2° C; Drive level = 0.5 mW		
Drive level (typ)	50-100	µW
Shunt capacitance (max)	10	pF
Motional capacitance (min)	10	fF
Serious resistance (max)	60	Ω
Spurious response (max)	> 5 dB below main within 500 kHz	

a. If the internal crystal oscillator is used, select a crystal with the following characteristics.

b. Equivalent differential capacitance across XTLPI/XTLN.

c. If crystal with a larger load is used, external shunt capacitors to ground should be added to make up the equivalent capacitance difference.

**Figure 24-3. External XTLP Oscillator Characteristics****Table 24-21. External XTLP Oscillator Characteristics**

Parameter Name	Symbol	Min	Nom	Max	Unit
XTLN Input Low Voltage	$XTLN_{ILV}$	-	-	0.8	-
XTLP Frequency <sup>a</sup>	$XTLP_f$	-	25.0	-	-
XTLP Period <sup>b</sup>	$T_{clkper}$	-	40	-	-
XTLP Duty Cycle	$XTLP_{DC}$	40	-	60	%
		40		60	
Rise/Fall Time	$T_r, T_f$	-	-	4.0	ns
Absolute Jitter		-	-	0.1	ns

a. IEEE 802.3 frequency tolerance ±50 ppm.

- b. IEEE 802.3 frequency tolerance  $\pm 50$  ppm.

## 24.2.7 Hibernation Module

The Hibernation Module requires special system implementation considerations since it is intended to power-down all other sections of its host device. The system power-supply distribution and interfaces of the system must be driven to 0 V<sub>DC</sub> or powered down with the same regulator controlled by /HIB.

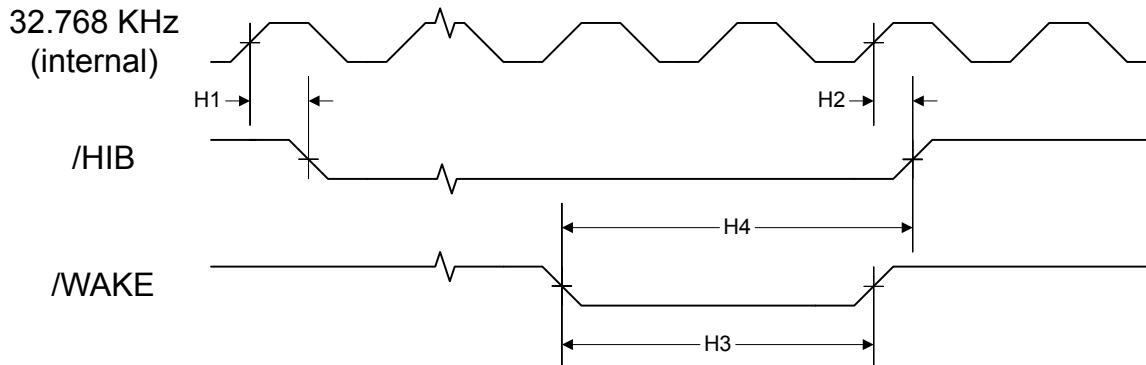
The regulators controlled by /HIB are expected to have a settling time of 250  $\mu$ s or less.

**Table 24-22. Hibernation Module Characteristics**

Parameter No	Parameter	Parameter Name	Min	Nom	Max	Unit
H1	$t_{HIB\_LOW}$	Internal 32.768 KHz clock reference rising edge to /HIB asserted	-	200	-	$\mu$ s
H2	$t_{HIB\_HIGH}$	Internal 32.768 KHz clock reference rising edge to /HIB deasserted	-	30	-	$\mu$ s
H3	$t_{WAKE\_ASSERT}$	/WAKE assertion time	62	-	-	$\mu$ s
H4	$t_{WAKETOHIB}$	/WAKE assert to /HIB desassert	62	-	124	$\mu$ s
H5	$t_{XOSC\_SETTLE}$	XOSC settling time <sup>a</sup>	20	-	-	ms
H6	$t_{HIB\_REG\_WRITE}$	Time for a write to non-volatile registers in HIB module to complete	92	-	-	$\mu$ s
H7	$t_{HIB\_TO\_VDD}$	/HIB deassert to VDD and VDD25 at minimum operational level	-	-	250	$\mu$ s

a. This parameter is highly sensitive to PCB layout and trace lengths, which may make this parameter time longer. Care must be taken in PCB design to minimize trace lengths and RLC (resistance, inductance, capacitance).

**Figure 24-4. Hibernation Module Timing**



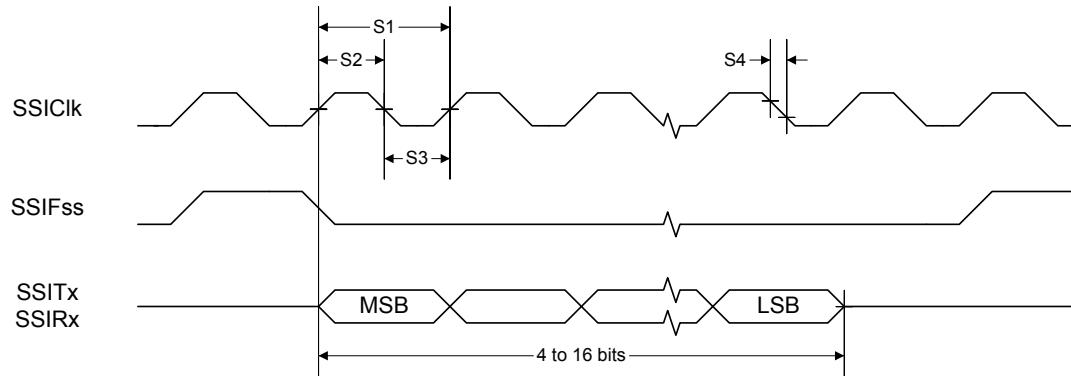
## 24.2.8 Synchronous Serial Interface (SSI)

**Table 24-23. SSI Characteristics**

Parameter No	Parameter	Parameter Name	Min	Nom	Max	Unit
S1	$t_{clk\_per}$	SSIClk cycle time	2	-	65024	system clocks
S2	$t_{clk\_high}$	SSIClk high time	-	1/2	-	$t_{clk\_per}$
S3	$t_{clk\_low}$	SSIClk low time	-	1/2	-	$t_{clk\_per}$
S4	$t_{clkrf}$	SSIClk rise/fall time	-	7.4	26	ns
S5	$t_{DMd}$	Data from master valid delay time	0	-	20	ns
S6	$t_{DMs}$	Data from master setup time	20	-	-	ns
S7	$t_{DMh}$	Data from master hold time	40	-	-	ns

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
S8	$t_{DSS}$	Data from slave setup time	20	-	-	ns
S9	$t_{DSh}$	Data from slave hold time	40	-	-	ns

**Figure 24-5. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement**



**Figure 24-6. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer**

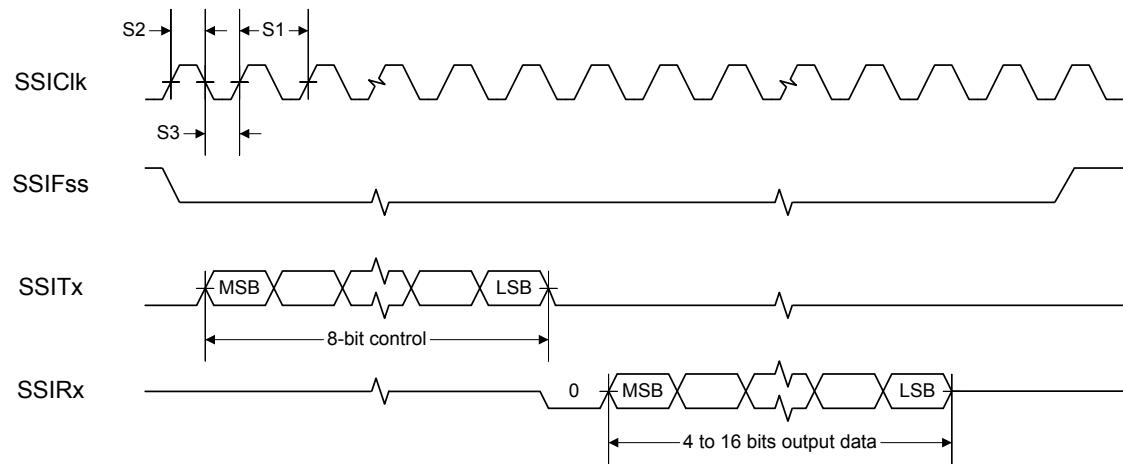
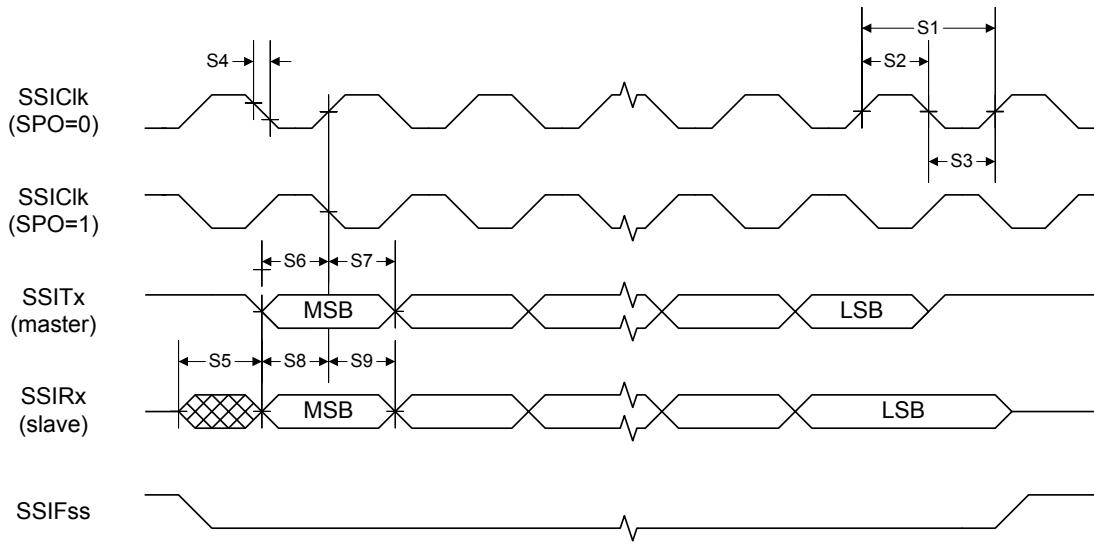


Figure 24-7. SSI Timing for SPI Frame Format (FRF=00), with SPH=1



## 24.2.9 JTAG and Boundary Scan

Table 24-24. JTAG Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
J1	$f_{TCK}$	TCK operational clock frequency	0	-	10	MHz
J2	$t_{TCK}$	TCK operational clock period	100	-	-	ns
J3	$t_{TCK\_LOW}$	TCK clock Low time	-	$t_{TCK}$	-	ns
J4	$t_{TCK\_HIGH}$	TCK clock High time	-	$t_{TCK}$	-	ns
J5	$t_{TCK\_R}$	TCK rise time	0	-	10	ns
J6	$t_{TCK\_F}$	TCK fall time	0	-	10	ns
J7	$t_{TMS\_SU}$	TMS setup time to TCK rise	20	-	-	ns
J8	$t_{TMS\_HLD}$	TMS hold time from TCK rise	20	-	-	ns
J9	$t_{TDI\_SU}$	TDI setup time to TCK rise	25	-	-	ns
J10	$t_{TDI\_HLD}$	TDI hold time from TCK rise	25	-	-	ns
J11 $t_{TDO\_ZDV}$	TCK fall to Data Valid from High-Z	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

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
J13 $t_{TDO\_DVZ}$	TCK fall to High-Z from Data Valid	2-mA drive	-	9	11	ns
		4-mA drive		7	9	ns
		8-mA drive		6	8	ns
		8-mA drive with slew rate control		7	9	ns
J14	$t_{TRST}$	TRST assertion time	100	-	-	ns
J15	$t_{TRST\_SU}$	TRST setup time to TCK rise	10	-	-	ns

Figure 24-8. JTAG Test Clock Input Timing

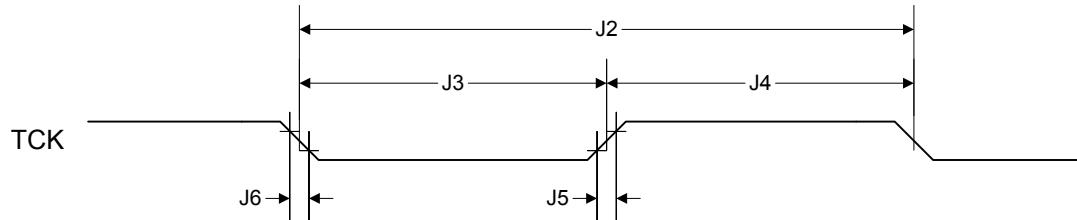


Figure 24-9. JTAG Test Access Port (TAP) Timing

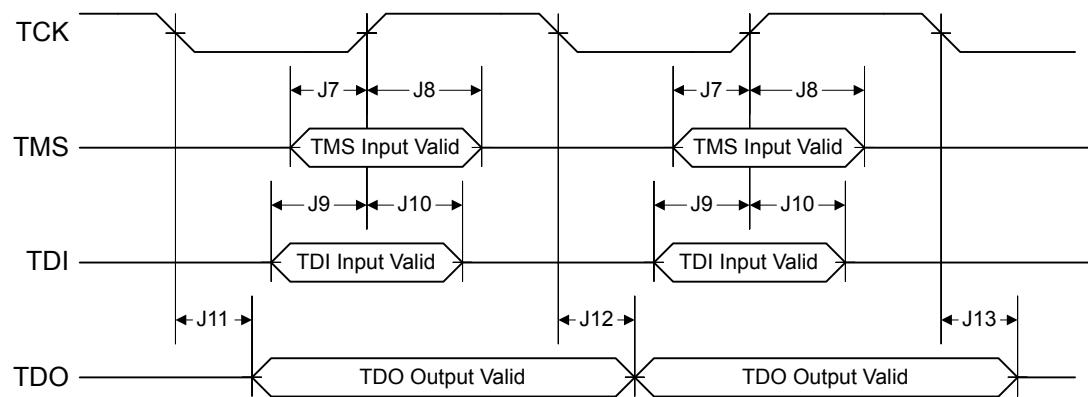
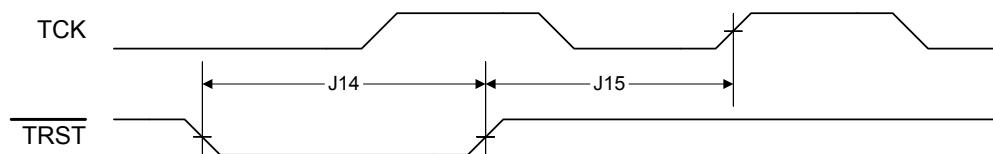


Figure 24-10. JTAG TRST Timing



## 24.2.10 General-Purpose I/O

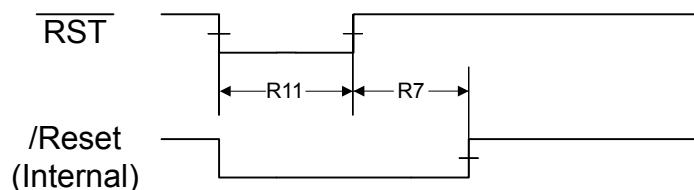
**Note:** All GPIOs are 5 V-tolerant.

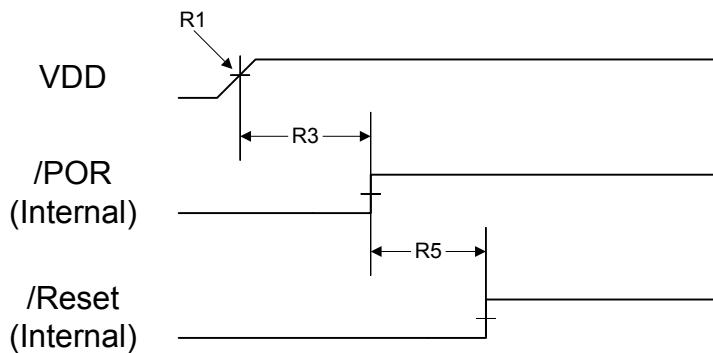
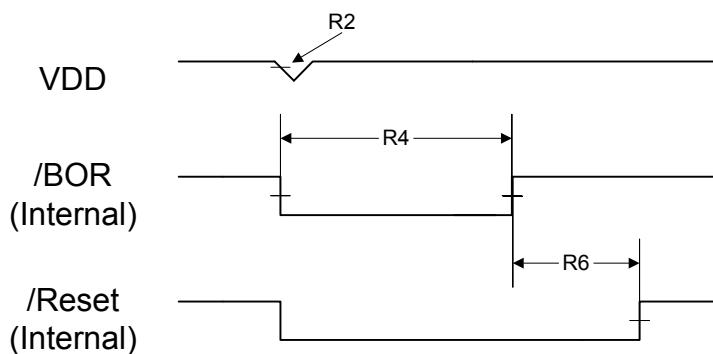
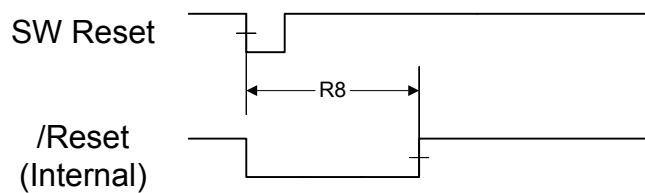
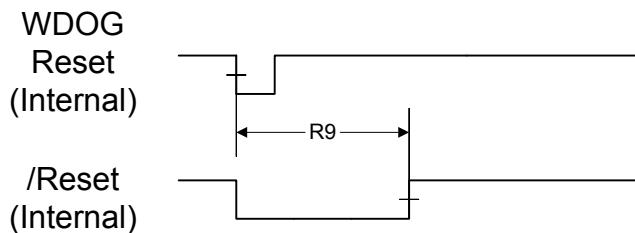
**Table 24-25. GPIO Characteristics**

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

**24.2.11 Reset****Table 24-26. Reset Characteristics**

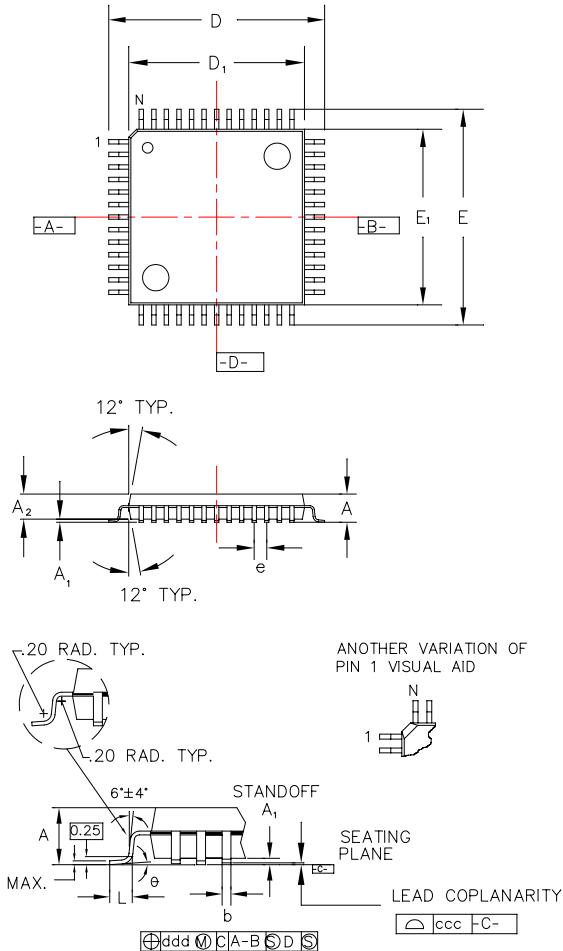
Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
R1	$V_{TH}$	Reset threshold	-	2.0	-	V
R2	$V_{BTH}$	Brown-Out threshold	2.85	2.9	2.95	V
R3	$T_{POR}$	Power-On Reset timeout	-	10	-	ms
R4	$T_{BOR}$	Brown-Out timeout	-	500	-	$\mu$ s
R5	$T_{IRPOR}$	Internal reset timeout after POR	6	-	11	ms
R6	$T_{IRBOR}$	Internal reset timeout after BOR <sup>a</sup>	0	-	1	$\mu$ s
R7	$T_{IRHWR}$	Internal reset timeout after hardware reset ( $\overline{RST}$ pin)	0	-	1	ms
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	$T_{VDDRISE}$	Supply voltage ( $V_{DD}$ ) rise time (0V-3.3V)	-	-	100	ms
R11	$T_{MIN}$	Minimum $\overline{RST}$ pulse width	2	-	-	$\mu$ s

a.  $20 * t_{MOSC\_per}$ **Figure 24-11. External Reset Timing ( $\overline{RST}$ )**

**Figure 24-12. Power-On Reset Timing****Figure 24-13. Brown-Out Reset Timing****Figure 24-14. Software Reset Timing****Figure 24-15. Watchdog Reset Timing**

## 25 Package Information

Figure 25-1. 100-Pin LQFP Package



**Note:** The following notes apply to the package drawing.

1. All dimensions shown in mm.
2. Dimensions shown are nominal with tolerances indicated.
3. Foot length 'L' is measured at gage plane 0.25 mm above seating plane.

Body +2.00 mm Footprint, 1.4 mm package thickness		
Symbols	Leads	100L
A	Max.	1.60
$A_1$		0.05 Min./0.15 Max.
$A_2$	$\pm 0.05$	1.40
D	$\pm 0.20$	16.00
$D_1$	$\pm 0.05$	14.00
E	$\pm 0.20$	16.00
$E_1$	$\pm 0.05$	14.00
L	$\pm 0.15/-0.10$	0.60
e	BASIC	0.50
b	$\pm 0.05$	0.22
$\theta$	====	$0^\circ \sim 7^\circ$
ddd	Max.	0.08
ccc	Max.	0.08
JEDEC Reference Drawing		MS-026
Variation Designator		BED

## 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 341 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 597).

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>System Control</b>															
Base 0x400F.E000															
DID0, type RO, offset 0x000, reset -															
VER														CLASS	
	MAJOR													MINOR	
PBORCTL, type R/W, offset 0x030, reset 0x0000.7FFD															
															BORIOR
LDOPCTL, type R/W, offset 0x034, reset 0x0000.0000															
														VADJ	
RIS, type RO, offset 0x050, reset 0x0000.0000															
														PLLLRIS	
IMC, type R/W, offset 0x054, reset 0x0000.0000															
														PLLLIM	
MISC, type R/W1C, offset 0x058, reset 0x0000.0000															
														PLLLMIS	
RESC, type R/W, offset 0x05C, reset -															
														LDO	SW
														WDT	BOR
														POR	EXT
RCC, type R/W, offset 0x060, reset 0x07AE.3AD1															
			ACG		SYSDIV		USESYSDIV		USEPVMIDIV		PWMDIV				
PWRDN		BYPASS			XTAL			OSCSRC					IOSCDIS	MOSCDIS	
PLLCFG, type RO, offset 0x064, reset -															
													F	R	
RCC2, type R/W, offset 0x070, reset 0x0780.2800															
USERCC2			SYSDIV2												
	PWRDN2	BYPASS2						OSCSRC2							
DSLPCLKCFG, type R/W, offset 0x144, reset 0x0780.0000															
			DSDIVORIDE										DSOSCSRC		
DID1, type RO, offset 0x004, reset -															
VER		FAM											PARTNO		
PINCOUNT								TEMP					PKG	ROHS	QUAL
DC0, type RO, offset 0x008, reset 0x00FF.007F															
			SRAMSZ					FLASHSZ							
DC1, type RO, offset 0x010, reset 0x0111.32FF															
			CAN0					PWM						ADC	
MINSYSDIV		MAXADCSPD			MPU	HIB	TEMPSNS	PLL	WDT	SWO	SWD		JTAG		
DC2, type RO, offset 0x014, reset 0x010F.1313															
	I2C0	QEI1	QEI0					SSI0					UART1	UART0	
DC3, type RO, offset 0x018, reset 0x030F.81FF															
PWMFAULT		CCP1	CCP0					ADC3	ADC2	ADC1	ADC0				
	C00	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 0x5100.007F</b>															
EPHY0	EMAC0				E1588				GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
<b>RCGC0, type R/W, offset 0x100, reset 0x00000040</b>															
				CANO				PWM							ADC
			MAXADCSPD			HIB			WDT						
<b>SCGC0, type R/W, offset 0x110, reset 0x00000040</b>															
				CANO				PWM							ADC
			MAXADCSPD			HIB			WDT						
<b>DCGC0, type R/W, offset 0x120, reset 0x00000040</b>															
				CANO				PWM							ADC
			MAXADCSPD			HIB			WDT						
<b>RCGC1, type R/W, offset 0x104, reset 0x00000000</b>															
				COMP0							TIMER3	TIMER2	TIMER1	TIMER0	
I2C0			QEI1	QEIO				SSI0					UART1	UART0	
<b>SCGC1, type R/W, offset 0x114, reset 0x00000000</b>															
				COMP0							TIMER3	TIMER2	TIMER1	TIMER0	
I2C0			QEI1	QEIO				SSI0					UART1	UART0	
<b>DCGC1, type R/W, offset 0x124, reset 0x00000000</b>															
				COMP0							TIMER3	TIMER2	TIMER1	TIMER0	
I2C0			QEI1	QEIO				SSI0					UART1	UART0	
<b>RCGC2, type R/W, offset 0x108, reset 0x00000000</b>															
EPHY0	EMAC0								GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
<b>SCGC2, type R/W, offset 0x118, reset 0x00000000</b>															
EPHY0	EMAC0								GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
<b>DCGC2, type R/W, offset 0x128, reset 0x00000000</b>															
EPHY0	EMAC0								GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
<b>SRCR0, type R/W, offset 0x040, reset 0x00000000</b>															
				CANO				PWM							ADC
				HIB											
<b>SRCR1, type R/W, offset 0x044, reset 0x00000000</b>															
				COMP0							TIMER3	TIMER2	TIMER1	TIMER0	
I2C0			QEI1	QEIO				SSI0					UART1	UART0	
<b>SRCR2, type R/W, offset 0x048, reset 0x00000000</b>															
EPHY0	EMAC0								GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
<b>Hibernation Module</b>															
Base 0x400FC000															
<b>HIBRTCC, type RO, offset 0x000, reset 0x0000.0000</b>															
					RTCC										
					RTCC										
<b>HIBRTCM0, type R/W, offset 0x004, reset 0xFFFF.FFFF</b>															
					RTCM0										
					RTCM0										
<b>HIBRTCM1, type R/W, offset 0x008, reset 0xFFFF.FFFF</b>															
					RTCM1										
					RTCM1										
<b>HIBRTCLD, type R/W, offset 0x00C, reset 0xFFFF.FFFF</b>															
					RTCLD										
					RTCLD										

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>HIBCTL</b> , type R/W, offset 0x010, reset 0x0000.0000																
								VABORT	CLK32EN	LOWBATEN	PINWEN	RTCWEN	CLKSEL	HIBREQ	RTCE	
<b>HIBIM</b> , type R/W, offset 0x014, reset 0x0000.0000																
												EXTW	LOWBAT	RTCA1	RTCA0	
<b>HIBRIS</b> , type RO, offset 0x018, reset 0x0000.0000																
												EXTW	LOWBAT	RTCA1	RTCA0	
<b>HIBMIS</b> , type RO, offset 0x01C, reset 0x0000.0000																
												EXTW	LOWBAT	RTCA1	RTCA0	
<b>HIBIC</b> , type R/W1C, offset 0x020, reset 0x0000.0000																
												EXTW	LOWBAT	RTCA1	RTCA0	
<b>HIBRTCT</b> , type R/W, offset 0x024, reset 0x0000.7FFF																
								TRIM								
<b>HIBDATA</b> , type R/W, offset 0x030-0x12C, reset 0x0000.0000																
					RTD			RTD								
<b>Internal Memory</b>																
<b>Flash Control Offset</b>																
Base 0x400F.D000																
<b>FMA</b> , type R/W, offset 0x000, reset 0x0000.0000																
								OFFSET								
<b>FMD</b> , type R/W, offset 0x004, reset 0x0000.0000								DATA								
								DATA								
<b>FMC</b> , type R/W, offset 0x008, reset 0x0000.0000								WRKEY					COMT	MERASE	ERASE	WRITE
<b>FCRIS</b> , type RO, offset 0x00C, reset 0x0000.0000																
													PRIS	ARIS		
<b>FCIM</b> , type R/W, offset 0x010, reset 0x0000.0000																
													PMASK	AMASK		
<b>FCMISC</b> , type R/W1C, offset 0x014, reset 0x0000.0000																
													PMISC	AMISC		
<b>Internal Memory</b>																
<b>System Control Offset</b>																
Base 0x400F.E000																
<b>USECRL</b> , type R/W, offset 0x140, reset 0x31																
									USEC							
<b>FMPRE0</b> , type R/W, offset 0x130 and 0x200, reset 0xFFFF.FFFF																
					READ_ENABLE			READ_ENABLE								

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>FMPPE0</b> , type R/W, offset 0x134 and 0x400, reset 0xFFFF.FFFF															
PROG_ENABLE															
PROG_ENABLE															
<b>USER_DBG</b> , type R/W, offset 0x1D0, reset 0xFFFF.FFFE															
NW								DATA							
								DATA						DBG1	DBG0
<b>USER_REG0</b> , type R/W, offset 0x1E0, reset 0xFFFF.FFFF															
NW								DATA							
								DATA							
<b>USER_REG1</b> , type R/W, offset 0x1E4, reset 0xFFFF.FFFF															
NW								DATA							
								DATA							
<b>FMPRE1</b> , type R/W, offset 0x204, reset 0xFFFF.FFFF															
								READ_ENABLE							
								READ_ENABLE							
<b>FMPRE2</b> , type R/W, offset 0x208, reset 0xFFFF.FFFF															
								READ_ENABLE							
								READ_ENABLE							
<b>FMPRE3</b> , type R/W, offset 0x20C, reset 0xFFFF.FFFF															
								READ_ENABLE							
								READ_ENABLE							
<b>FMPPE1</b> , type R/W, offset 0x404, reset 0xFFFF.FFFF															
								PROG_ENABLE							
								PROG_ENABLE							
<b>FMPPE2</b> , type R/W, offset 0x408, reset 0xFFFF.FFFF															
								PROG_ENABLE							
								PROG_ENABLE							
<b>FMPPE3</b> , type R/W, offset 0x40C, reset 0xFFFF.FFFF															
								PROG_ENABLE							
								PROG_ENABLE							
<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															
GPIO Port F base: 0x4002.5000															
GPIO Port G base: 0x4002.6000															
<b>GPIODATA</b> , type R/W, offset 0x000, reset 0x0000.0000															
															DATA
<b>GPIODIR</b> , type R/W, offset 0x400, reset 0x0000.0000															
															DIR
<b>GPIOIS</b> , type R/W, offset 0x404, reset 0x0000.0000															
															IS
<b>GPIOIBE</b> , type R/W, offset 0x408, reset 0x0000.0000															
															IBE
<b>GPIOIEV</b> , type R/W, offset 0x40C, reset 0x0000.0000															
															IEV

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>GPIOIM, type R/W, offset 0x410, reset 0x0000.0000</b>															
IME															
<b>GPIORIS, type RO, offset 0x414, reset 0x0000.0000</b>															
RIS															
<b>GPIOMIS, type RO, offset 0x418, reset 0x0000.0000</b>															
MIS															
<b>GPIOICR, type W1C, offset 0x41C, reset 0x0000.0000</b>															
IC															
<b>GPIOAFSEL, type R/W, offset 0x420, reset -</b>															
AFSEL															
<b>GPIODR2R, type R/W, offset 0x500, reset 0x0000.00FF</b>															
DRV2															
<b>GPIODR4R, type R/W, offset 0x504, reset 0x0000.0000</b>															
DRV4															
<b>GPIODR8R, type R/W, offset 0x508, reset 0x0000.0000</b>															
DRV8															
<b>GPIOODR, type R/W, offset 0x50C, reset 0x0000.0000</b>															
ODE															
<b>GPIOPUR, type R/W, offset 0x510, reset -</b>															
PUE															
<b>GPIOPDR, type R/W, offset 0x514, reset 0x0000.0000</b>															
PDE															
<b>GPIOSLR, type R/W, offset 0x518, reset 0x0000.0000</b>															
SRL															
<b>GPIODEN, type R/W, offset 0x51C, reset -</b>															
DEN															
<b>GPIOLOCK, type R/W, offset 0x520, reset 0x0000.0001</b>															
LOCK															
<b>GPIOCR, type -, offset 0x524, reset -</b>															
CR															
<b>GPIOPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000</b>															
PID4															
<b>GPIOPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000</b>															
PID5															

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>GPIOPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000</b>															
PID6															
<b>GPIOPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000</b>															
PID7															
<b>GPIOPeriphID0, type RO, offset 0xFE0, reset 0x0000.0061</b>															
PID0															
<b>GPIOPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000</b>															
PID1															
<b>GPIOPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018</b>															
PID2															
<b>GPIOPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001</b>															
PID3															
<b>GPIOCellID0, type RO, offset 0xFF0, reset 0x0000.000D</b>															
CID0															
<b>GPIOCellID1, type RO, offset 0xFF4, reset 0x0000.00F0</b>															
CID1															
<b>GPIOCellID2, type RO, offset 0xFF8, reset 0x0000.0005</b>															
CID2															
<b>GPIOCellID3, type RO, offset 0xFFC, reset 0x0000.00B1</b>															
CID3															
<b>General-Purpose Timers</b>															
Timer0 base: 0x4003.0000															
Timer1 base: 0x4003.1000															
Timer2 base: 0x4003.2000															
Timer3 base: 0x4003.3000															
<b>GPTMCFG, type R/W, offset 0x000, reset 0x0000.0000</b>															
GPTMCFG															
<b>GPTMTAMR, type R/W, offset 0x004, reset 0x0000.0000</b>															
TAAMS    TACMR    TAMR															
<b>GPTMTBMR, type R/W, offset 0x008, reset 0x0000.0000</b>															
TBAMS    TBCMR    TBMR															
<b>GPTMCTL, type R/W, offset 0x00C, reset 0x0000.0000</b>															
TBPWML    TBOTE    TBEVENT    TBSTALL    TBEN    TAPWML    TAOTE    RTCEN    TAEVENT    TASTALL    TAEN															
<b>GPTMIMR, type R/W, offset 0x018, reset 0x0000.0000</b>															
CBEIM    CBMIM    TBTOIM    RTCIM    CAEIM    CAMIM    TATOIM															
<b>GPTMRIS, type RO, offset 0x01C, reset 0x0000.0000</b>															
CBERIS    CBMRIS    BTORIS    RTCRIS    CAERIS    CAMRIS    TATORIS															

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>GPTMMIS</b> , type RO, offset 0x020, reset 0x0000.0000															
					CBEMIS	CBMMIS	TBTOMIS					RTCMIS	CAEMIS	CAMMIS	TATOMIS
<b>GPTMICR</b> , type W1C, offset 0x024, reset 0x0000.0000															
				CBECINT	CBMCINT	TBTOCINT						RTCCINT	CAECINT	CAMCINT	TATOCINT
<b>GPTMTAILR</b> , type R/W, offset 0x028, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)															
							TAILRH								
							TAILRL								
<b>GPTMTBILR</b> , type R/W, offset 0x02C, reset 0x0000.FFFF															
								TBILRL							
<b>GPTMTAMATCHR</b> , type R/W, offset 0x030, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)															
								TAMRH							
							TAMRL								
<b>GPTMTBMATCHR</b> , type R/W, offset 0x034, reset 0x0000.FFFF															
								TBMRL							
<b>GPTMTAPR</b> , type R/W, offset 0x038, reset 0x0000.0000															
									TAPSР						
<b>GPTMTBPR</b> , type R/W, offset 0x03C, reset 0x0000.0000															
									TBPSR						
<b>GPTMTAPMR</b> , type R/W, offset 0x040, reset 0x0000.0000															
									TAPSMR						
<b>GPTMTBPMR</b> , type R/W, offset 0x044, reset 0x0000.0000															
									TBPSMR						
<b>GPTMTAR</b> , type RO, offset 0x048, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)															
								TARH							
							TARL								
<b>GPTMTBR</b> , type RO, offset 0x04C, reset 0x0000.FFFF															
								TBRL							
<b>Watchdog Timer</b>															
Base 0x4000.0000															
<b>WDTLOAD</b> , type R/W, offset 0x000, reset 0xFFFF.FFFF															
								WDTLoad							
								WDTLoad							
<b>WDTVVALUE</b> , type RO, offset 0x004, reset 0xFFFF.FFFF									WDTValue						
								WDTValue							
<b>WDTCTL</b> , type R/W, offset 0x008, reset 0x0000.0000															
														RESEN	INTEN
<b>WDTICR</b> , type WO, offset 0x00C, reset -															
									WDTIntClr						
									WDTIntClr						
<b>WDTRIS</b> , type RO, offset 0x010, reset 0x0000.0000															
															WDTRIS



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>ADCIM, type R/W, offset 0x008, reset 0x0000.0000</b>																	
														MASK3	MASK2	MASK1	MASK0
<b>ADCISC, type R/W1C, offset 0x00C, reset 0x0000.0000</b>																	
														IN3	IN2	IN1	IN0
<b>ADCOSTAT, type R/W1C, offset 0x010, reset 0x0000.0000</b>																	
														OV3	OV2	OV1	OV0
<b>ADCEMUX, type R/W, offset 0x014, reset 0x0000.0000</b>																	
	EM3			EM2				EM1				EM0					
<b>ADCUSTAT, type R/W1C, offset 0x018, reset 0x0000.0000</b>																	
														UV3	UV2	UV1	UV0
<b>ADCSSPRI, type R/W, offset 0x020, reset 0x0000.3210</b>																	
		SS3			SS2			SS1				SS0					
<b>ADCPSSI, type WO, offset 0x028, reset -</b>																	
														SS3	SS2	SS1	SS0
<b>ADCSAC, type R/W, offset 0x030, reset 0x0000.0000</b>																	
																AVG	
<b>ADCSSMUX0, type R/W, offset 0x040, reset 0x0000.0000</b>																	
		MUX7			MUX6			MUX5				MUX4					
		MUX3			MUX2			MUX1				MUX0					
<b>ADCSSCTL0, type R/W, offset 0x044, reset 0x0000.0000</b>																	
TS7	IE7	END7	D7	TS6	IE6	END6	D6	TS5	IE5	END5	D5	TS4	IE4	END4	D4		
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0		
<b>ADCSSFIFO0, type RO, offset 0x048, reset 0x0000.0000</b>																	
															DATA		
<b>ADCSSFIFO1, type RO, offset 0x068, reset 0x0000.0000</b>																	
															DATA		
<b>ADCSSFIFO2, type RO, offset 0x088, reset 0x0000.0000</b>																	
															DATA		
<b>ADCSSFIFO3, type RO, offset 0x0A8, reset 0x0000.0000</b>																	
															DATA		
<b>ADCSSFSTAT0, type RO, offset 0x04C, reset 0x0000.0100</b>																	
		FULL				EMPTY			HPTR				T PTR				
<b>ADCSSFSTAT1, type RO, offset 0x06C, reset 0x0000.0100</b>																	
		FULL				EMPTY			HPTR				T PTR				
<b>ADCSSFSTAT2, type RO, offset 0x08C, reset 0x0000.0100</b>																	
		FULL				EMPTY			HPTR				T PTR				

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>ADCSSFSTAT3, type RO, offset 0x0AC, reset 0x0000.0100</b>															
			FULL			EMPTY		HPTR			TPTR				
<b>ADCSSMUX1, type RO, offset 0x060, reset 0x0000.0000</b>															
		MUX3			MUX2			MUX1			MUX0				
<b>ADCSSMUX2, type RO, offset 0x080, reset 0x0000.0000</b>															
	MUX3			MUX2			MUX1			MUX0					
<b>ADCSSCTL1, type RO, offset 0x064, reset 0x0000.0000</b>															
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
<b>ADCSSCTL2, type RO, offset 0x084, reset 0x0000.0000</b>															
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
<b>ADCSSMUX3, type R/W, offset 0xA0, reset 0x0000.0000</b>															
															MUX0
<b>ADCSSCTL3, type R/W, offset 0xA4, reset 0x0000.0002</b>															
															TS0 IE0 END0 D0
<b>ADCTMLB, type RO, offset 0x100, reset 0x0000.0000</b>								CNT		CONT	DIFF	TS		MUX	
															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</b>															
				OE	BE	PE	FE								DATA
<b>UARTRSR/UARTECR, type RO, offset 0x004, reset 0x0000.0000</b>															
															OE BE PE FE
<b>UARTRSR/UARTECR, type WO, offset 0x004, reset 0x0000.0000</b>															
															DATA
<b>UARTFR, type RO, offset 0x018, reset 0x0000.0090</b>															
								TXFE	RXFF	TXFF	RXFE	BUSY			
<b>UARTILPR, type R/W, offset 0x020, reset 0x0000.0000</b>															
															ILPDVSR
<b>UARTIBRD, type R/W, offset 0x024, reset 0x0000.0000</b>															
								DIVINT							
<b>UARTFBRD, type R/W, offset 0x028, reset 0x0000.0000</b>															
															DIVFRAC

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>UARTLCRH</b> , type R/W, offset 0x02C, reset 0x0000.0000															
								SPS	WLEN	FEN	STP2	EPS	PEN	BRK	
<b>UARTCTL</b> , type R/W, offset 0x030, reset 0x0000.0300															
					RXE	TXE	LBE					SIRLP	SIREN	UARTEN	
<b>UARTIFLS</b> , type R/W, offset 0x034, reset 0x0000.0012															
									RXIFLSEL					TXIFLSEL	
<b>UARTIM</b> , type R/W, offset 0x038, reset 0x0000.0000															
				OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM					
<b>UARTRIS</b> , type RO, offset 0x03C, reset 0x0000.000F															
				OERIS	BERIS	PERIS	FERIS	RTRIS	TXRIS	RXRIS					
<b>UARTMIS</b> , type RO, offset 0x040, reset 0x0000.0000															
				OEMIS	BEMIS	PEMIS	FEMIS	RTMIS	TXMIS	RXMIS					
<b>UARTICR</b> , type W1C, offset 0x044, reset 0x0000.0000															
				OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC					
<b>UARTPeriphID4</b> , type RO, offset 0xFD0, reset 0x0000.0000															
															PID4
<b>UARTPeriphID5</b> , type RO, offset 0xFD4, reset 0x0000.0000															
															PID5
<b>UARTPeriphID6</b> , type RO, offset 0xFD8, reset 0x0000.0000															
															PID6
<b>UARTPeriphID7</b> , type RO, offset 0xFDC, reset 0x0000.0000															
															PID7
<b>UARTPeriphID0</b> , type RO, offset 0xFE0, reset 0x0000.0011															
															PID0
<b>UARTPeriphID1</b> , type RO, offset 0xFE4, reset 0x0000.0000															
															PID1
<b>UARTPeriphID2</b> , type RO, offset 0xFE8, reset 0x0000.0018															
															PID2
<b>UARTPeriphID3</b> , type RO, offset 0xFEC, reset 0x0000.0001															
															PID3
<b>UARTPCellID0</b> , type RO, offset 0xFF0, reset 0x0000.000D															
															CID0
<b>UARTPCellID1</b> , type RO, offset 0xFF4, reset 0x0000.00F0															
															CID1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>UARTPCellID2, type RO, offset 0xFF8, reset 0x0000.0005</b>															
CID2															
<b>UARTPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1</b>															
<b>Synchronous Serial Interface (SSI)</b>															
SSI0 base: 0x4000.8000															
<b>SSICR0, type R/W, offset 0x000, reset 0x0000.0000</b>															
SCR															
SPH															
SPO															
FRF															
DSS															
<b>SSICR1, type R/W, offset 0x004, reset 0x0000.0000</b>															
SOD															
MS															
SSE															
LBM															
<b>SSIDR, type R/W, offset 0x008, reset 0x0000.0000</b>															
DATA															
<b>SSISR, type RO, offset 0x00C, reset 0x0000.0003</b>															
BSY															
RFF															
RNE															
TNF															
TFE															
<b>SSICPSR, type R/W, offset 0x010, reset 0x0000.0000</b>															
CPDVSR															
<b>SSIIM, type R/W, offset 0x014, reset 0x0000.0000</b>															
TXIM															
RXIM															
RTIM															
RORIM															
<b>SSIRIS, type RO, offset 0x018, reset 0x0000.0008</b>															
TXRIS															
RXRIS															
RTRIS															
RORMIS															
<b>SSIICR, type W1C, offset 0x020, reset 0x0000.0000</b>															
RTIC															
<b>SSIPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000</b>															
PID4															
<b>SSIPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000</b>															
PID5															
<b>SSIPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000</b>															
PID6															
<b>SSIPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000</b>															
PID7															
<b>SSIPeriphID0, type RO, offset 0xFE0, reset 0x0000.0022</b>															
PID0															
<b>SSIPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000</b>															
PID1															

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>SSIPeriphID2</b> , type RO, offset 0xFE8, reset 0x0000.0018															
PID2															
<b>SSIPeriphID3</b> , type RO, offset 0xFEC, reset 0x0000.0001															
PID3															
<b>SSIPCellID0</b> , type RO, offset 0xFF0, reset 0x0000.000D															
CID0															
<b>SSIPCellID1</b> , type RO, offset 0xFF4, reset 0x0000.00F0															
CID1															
<b>SSIPCellID2</b> , type RO, offset 0xFF8, reset 0x0000.0005															
CID2															
<b>SSIPCellID3</b> , type RO, offset 0xFFC, reset 0x0000.00B1															
CID3															
<b>Inter-Integrated Circuit (I<sup>2</sup>C) Interface</b>															
<b>I<sup>2</sup>C Master</b>															
I <sup>2</sup> C Master 0 base: 0x4002.0000															
<b>I<sup>2</sup>CMSCA</b> , type R/W, offset 0x000, reset 0x0000.0000															
SA															
<b>I<sup>2</sup>CMCS</b> , type RO, offset 0x004, reset 0x0000.0000															
BUSBSY IDLE ARBLST DATAACK ADRACK ERROR BUSY															
<b>I<sup>2</sup>CMCS</b> , type WO, offset 0x004, reset 0x0000.0000															
ACK STOP START RUN															
<b>I<sup>2</sup>CMDR</b> , type R/W, offset 0x008, reset 0x0000.0000															
DATA															
<b>I<sup>2</sup>CMTPR</b> , type R/W, offset 0x00C, reset 0x0000.0001															
TPR															
<b>I<sup>2</sup>CMIMR</b> , type R/W, offset 0x010, reset 0x0000.0000															
IM															
<b>I<sup>2</sup>CMRIS</b> , type RO, offset 0x014, reset 0x0000.0000															
RIS															
<b>I<sup>2</sup>CMMIS</b> , type RO, offset 0x018, reset 0x0000.0000															
MIS															
<b>I<sup>2</sup>CMICR</b> , type WO, offset 0x01C, reset 0x0000.0000															
IC															
<b>I<sup>2</sup>CMCR</b> , type R/W, offset 0x020, reset 0x0000.0000															
SFE MFE LPBK															
<b>Inter-Integrated Circuit (I<sup>2</sup>C) Interface</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>I<sup>2</sup>C Slave</b>															
I <sup>2</sup> C Slave 0 base: 0x4002.0800															
I <sup>2</sup> C SOAR, type R/W, offset 0x000, reset 0x0000.0000															
OAR															
I <sup>2</sup> C CSCSR, type RO, offset 0x004, reset 0x0000.0000															
FBR TREQ RREQ															
I <sup>2</sup> C CSCSR, type WO, offset 0x004, reset 0x0000.0000															
DA															
I <sup>2</sup> C CSDR, type R/W, offset 0x008, reset 0x0000.0000															
DATA															
I <sup>2</sup> C SIMR, type R/W, offset 0x00C, reset 0x0000.0000															
IM															
I <sup>2</sup> C SRIS, type RO, offset 0x010, reset 0x0000.0000															
RIS															
I <sup>2</sup> C SSMIS, type RO, offset 0x014, reset 0x0000.0000															
MIS															
I <sup>2</sup> C SICR, type WO, offset 0x018, reset 0x0000.0000															
IC															
<b>Controller Area Network (CAN) Module</b>															
CAN0 base: 0x4004.0000															
CANCTL, type R/W, offset 0x000, reset 0x0000.0001															
Test CCE DAR EIE SIE IE INIT															
CANSTS, type R/W, offset 0x004, reset 0x0000.0000															
BOff EWarn EPass RxOK TxOK LEC															
CANERR, type RO, offset 0x008, reset 0x0000.0000															
RP REC TEC															
CANBIT, type R/W, offset 0x00C, reset 0x0000.2301															
TSeg2 TSeg1 SJW BRP															
CANINT, type RO, offset 0x010, reset 0x0000.0000															
IntId															
CANTST, type R/W, offset 0x014, reset 0x0000.0000															
Rx Tx LBack Silent Basic															
CANBRPE, type R/W, offset 0x018, reset 0x0000.0000															
BRPE															
CANIF1CRQ, type RO, offset 0x020, reset 0x0000.0001															
Busy MNUM															

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>CANIF2CRQ, type RO, offset 0x080, reset 0x0000.0001</b>															
Busy															
MNUM															
<b>CANIF1CMSK, type RO, offset 0x024, reset 0x0000.0000</b>															
WRNRD															
Mask															
Arb															
Control															
ClrIntPnd															
TxRqstNewDat															
DataA															
<b>CANIF2CMSK, type RO, offset 0x084, reset 0x0000.0000</b>															
WRNRD															
Mask															
Arb															
Control															
ClrIntPnd															
<b>CANIF1MSK1, type RO, offset 0x028, reset 0x0000.FFFF</b>															
Msk															
<b>CANIF2MSK1, type RO, offset 0x088, reset 0x0000.FFFF</b>															
Msk															
<b>CANIF1MSK2, type RO, offset 0x02C, reset 0x0000.FFFF</b>															
Msk															
<b>CANIF2MSK2, type RO, offset 0x08C, reset 0x0000.FFFF</b>															
Msk															
<b>CANIF1ARB1, type RO, offset 0x030, reset 0x0000.0000</b>															
ID															
<b>CANIF2ARB1, type RO, offset 0x090, reset 0x0000.0000</b>															
ID															
<b>CANIF1ARB2, type RO, offset 0x034, reset 0x0000.0000</b>															
ID															
MsgVal	Xtd	Dir													ID
<b>CANIF2ARB2, type RO, offset 0x094, reset 0x0000.0000</b>															
MsgVal	Xtd	Dir													ID
<b>CANIF1MCTL, type RO, offset 0x038, reset 0x0000.0000</b>															
NewDat	MsgLst	IntPnd	UMask	TxE	RxE	RmtEn	TxRqst	EoB							DLC
<b>CANIF2MCTL, type RO, offset 0x098, reset 0x0000.0000</b>															
NewDat	MsgLst	IntPnd	UMask	TxE	RxE	RmtEn	TxRqst	EoB							DLC
<b>CANIF1DA1, type R/W, offset 0x03C, reset 0x0000.0000</b>															
Data															
<b>CANIF1DA2, type R/W, offset 0x040, reset 0x0000.0000</b>															
Data															
<b>CANIF1DB1, type R/W, offset 0x044, reset 0x0000.0000</b>															
Data															
<b>CANIF1DB2, type R/W, offset 0x048, reset 0x0000.0000</b>															
Data															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>CANIF2DA1, type R/W, offset 0x09C, reset 0x0000.0000</b>															
Data															
<b>CANIF2DA2, type R/W, offset 0x0A0, reset 0x0000.0000</b>															
Data															
<b>CANIF2DB1, type R/W, offset 0x0A4, reset 0x0000.0000</b>															
Data															
<b>CANIF2DB2, type R/W, offset 0x0A8, reset 0x0000.0000</b>															
Data															
<b>CANTXRQ1, type RO, offset 0x100, reset 0x0000.0000</b>															
TxRqst															
<b>CANTXRQ2, type RO, offset 0x104, reset 0x0000.0000</b>															
TxRqst															
<b>CANNWDA1, type RO, offset 0x120, reset 0x0000.0000</b>															
NewDat															
<b>CANNWDA2, type RO, offset 0x124, reset 0x0000.0000</b>															
NewDat															
<b>CANMSG1INT, type RO, offset 0x140, reset 0x0000.0000</b>															
IntPnd															
<b>CANMSG2INT, type RO, offset 0x144, reset 0x0000.0000</b>															
IntPnd															
<b>CANMSG1VAL, type RO, offset 0x160, reset 0x0000.0000</b>															
MsgVal															
<b>CANMSG2VAL, type RO, offset 0x164, reset 0x0000.0000</b>															
MsgVal															
<b>Ethernet Controller</b>															
<b>Ethernet MAC</b>															
Base 0x4004.8000															
<b>MACRIS, type RO, offset 0x000, reset 0x0000.0000</b>															
PHYINT MDINT RXER FOV TXEMP TXER RXINT															
<b>MACIACK, type W1C, offset 0x000, reset 0x0000.0000</b>															
PHYINT MDINT RXER FOV TXEMP TXER RXINT															
<b>MACIM, type R/W, offset 0x004, reset 0x0000.007F</b>															
PHYINTM MDINTM RXERM FOVM TXEMPM TXERM RXINTM															
<b>MACRCTL, type R/W, offset 0x008, reset 0x0000.0008</b>															
RSTFIFO BADCRC PRMS AMUL RXEN															

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>MACTCTL</b> , type R/W, offset 0x00C, reset 0x0000.0000																		
												DUPLEX		CRC	PADEN	TXEN		
<b>MACDATA</b> , type RO, offset 0x010, reset 0x0000.0000																		
												RXDATA						
												RXDATA						
<b>MACDATA</b> , type WO, offset 0x010, reset 0x0000.0000																		
												TXDATA						
												TXDATA						
<b>MACIA0</b> , type R/W, offset 0x014, reset 0x0000.0000																		
												MACOCT4			MACOCT3			
												MACOCT2			MACOCT1			
<b>MACIA1</b> , type R/W, offset 0x018, reset 0x0000.0000																		
												MACOCT6			MACOCT5			
<b>MACTHR</b> , type R/W, offset 0x01C, reset 0x0000.003F																		
															THRESH			
<b>MACMCTL</b> , type R/W, offset 0x020, reset 0x0000.0000																		
															REGADR			
															WRITE	START		
<b>MACMDV</b> , type R/W, offset 0x024, reset 0x0000.0080																		
															DIV			
<b>MACMTXD</b> , type R/W, offset 0x02C, reset 0x0000.0000																		
															MDTX			
<b>MACMRXD</b> , type R/W, offset 0x030, reset 0x0000.0000																		
															MDRX			
<b>MACNP</b> , type RO, offset 0x034, reset 0x0000.0000																		
															NPR			
<b>MACTR</b> , type R/W, offset 0x038, reset 0x0000.0000																		
															NEWTX			
<b>MACTS</b> , type R/W, offset 0x03C, reset 0x0000.0000																		
															TSEN			
<b>Ethernet Controller</b>																		
<b>MII Management</b>																		
Base 0x4004.8000																		
MR0	type R/W, address 0x00, reset 0x3100																	
RESET	LOOPBK	SPEEDSL	ANEGEN	PWRDN	ISO	RANEG	DUPLEX	COLT										
MR1	type RO, address 0x01, reset 0x7849																	
100X_F	100X_H	10T_F	10T_H									MFPS	ANEGC	RFAULT	ANEGA	LINK	JAB	EXTD
MR2	type RO, address 0x02, reset 0x000E																	
												OUI[21:6]						
MR3	type RO, address 0x03, reset 0x7237																	
												OUI[5:0]		MN		RN		
MR4	type R/W, address 0x04, reset 0x01E1																	
NP		RF							A3	A2	A1	A0			S[4:0]			

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>MR5, type RO, address 0x05, reset 0x0000</b>															
NP ACK RF A[7:0] S[4:0]															
<b>MR6, type RO, address 0x06, reset 0x0000</b>															
PDF LPNPA PRX LPNEGA															
<b>MR16, type R/W, address 0x10, reset 0x0140</b>															
RPTR INPOL TXHIM SQEI NL10 APOL RVSPOL PCSBP RXCC															
<b>MR17, type R/W, address 0x11, reset 0x0000</b>															
JABBER_IE RXER_IE PRX_IE PDF_IE LPACK_IE LSCHG_IE RFAULT_IE ANECCOMP_E JABBER_INT RXER_INT PRX_INT PDF_INT LPACK_INT LSCHG_INT RFAULT_INT ANECCOMP_INT															
<b>MR18, type RO, address 0x12, reset 0x0000</b>															
ANEFG DPLX RATE RXSD RX_LOCK															
<b>MR19, type R/W, address 0x13, reset 0x4000</b>															
TXO[1:0]															
<b>MR23, type R/W, address 0x17, reset 0x0010</b>															
LED1[3:0] LED0[3:0]															
<b>MR24, type R/W, address 0x18, reset 0x00C0</b>															
PD_MODE AUTO_SW MDIX MDIX_CM MDIX_SD															
<b>Analog Comparator</b>															
Base 0x4003.C000															
<b>ACMIS, type R/W1C, offset 0x00, reset 0x0000.0000</b>															
INO															
<b>ACRIS, type RO, offset 0x04, reset 0x0000.0000</b>															
INO															
<b>ACINTEN, type R/W, offset 0x08, reset 0x0000.0000</b>															
INO															
<b>ACREFCTL, type R/W, offset 0x10, reset 0x0000.0000</b>															
EN RNG VREF															
<b>ACSTAT0, type RO, offset 0x20, reset 0x0000.0000</b>															
OVAL															
<b>ACCTL0, type R/W, offset 0x24, reset 0x0000.0000</b>															
TOEN ASRCP TSLVAL TSEN ISLVAL ISEN CINV															
<b>Pulse Width Modulator (PWM)</b>															
Base 0x4002.8000															
<b>PWMCTL, type R/W, offset 0x00, reset 0x0000.0000</b>															
GlobalSync2 GlobalSync1 GlobalSync0															
<b>PWMSYNC, type R/W, offset 0x04, reset 0x0000.0000</b>															
Sync2 Sync1 Sync0															
<b>PWMENABLE, type R/W, offset 0x08, reset 0x0000.0000</b>															
PWM5En PWM4En PWM3En PWM2En PWM1En PWM0En															
<b>PWMINVERT, type R/W, offset 0x0C, reset 0x0000.0000</b>															
PWM5Inv PWM4Inv PWM3Inv PWM2Inv PWM1Inv PWM0Inv															

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>PWMFAULT</b> , type R/W, offset 0x010, reset 0x0000.0000															
										Fault5	Fault4	Fault3	Fault2	Fault1	Fault0
<b>PWMINTEN</b> , type R/W, offset 0x014, reset 0x0000.0000															
														IntFault	
<b>PWMRIS</b> , type RO, offset 0x018, reset 0x0000.0000															
														IntFault	
<b>PWMISC</b> , type R/W1C, offset 0x01C, reset 0x0000.0000															
														IntFault	
<b>PWMSTATUS</b> , type RO, offset 0x020, reset 0x0000.0000															
															Fault
<b>PWM0CTL</b> , type R/W, offset 0x040, reset 0x0000.0000															
										CmpBUpd	CmpAUpd	LoadUpd	Debug	Mode	Enable
<b>PWM1CTL</b> , type R/W, offset 0x080, reset 0x0000.0000															
										CmpBUpd	CmpAUpd	LoadUpd	Debug	Mode	Enable
<b>PWM2CTL</b> , type R/W, offset 0x0C0, reset 0x0000.0000															
										CmpBUpd	CmpAUpd	LoadUpd	Debug	Mode	Enable
<b>PWM0INTEN</b> , type R/W, offset 0x044, reset 0x0000.0000															
										TrCmpBD	TrCmpBU	TrCmpAD	TrCmpAU	TrCntLoad	TrCntZero
<b>PWM1INTEN</b> , type R/W, offset 0x084, reset 0x0000.0000															
										TrCmpBD	TrCmpBU	TrCmpAD	TrCmpAU	TrCntLoad	TrCntZero
<b>PWM2INTEN</b> , type R/W, offset 0x0C4, reset 0x0000.0000															
										TrCmpBD	TrCmpBU	TrCmpAD	TrCmpAU	TrCntLoad	TrCntZero
<b>PWM0RIS</b> , type RO, offset 0x048, reset 0x0000.0000															
															IntCmpBD
<b>PWM1RIS</b> , type RO, offset 0x088, reset 0x0000.0000															
															IntCmpBD
<b>PWM2RIS</b> , type RO, offset 0x0C8, reset 0x0000.0000															
															IntCmpBD
<b>PWM0ISC</b> , type R/W1C, offset 0x04C, reset 0x0000.0000															
															IntCmpBD
<b>PWM1ISC</b> , type R/W1C, offset 0x08C, reset 0x0000.0000															
															IntCmpBD
<b>PWM2ISC</b> , type R/W1C, offset 0x0CC, reset 0x0000.0000															
															IntCmpBD

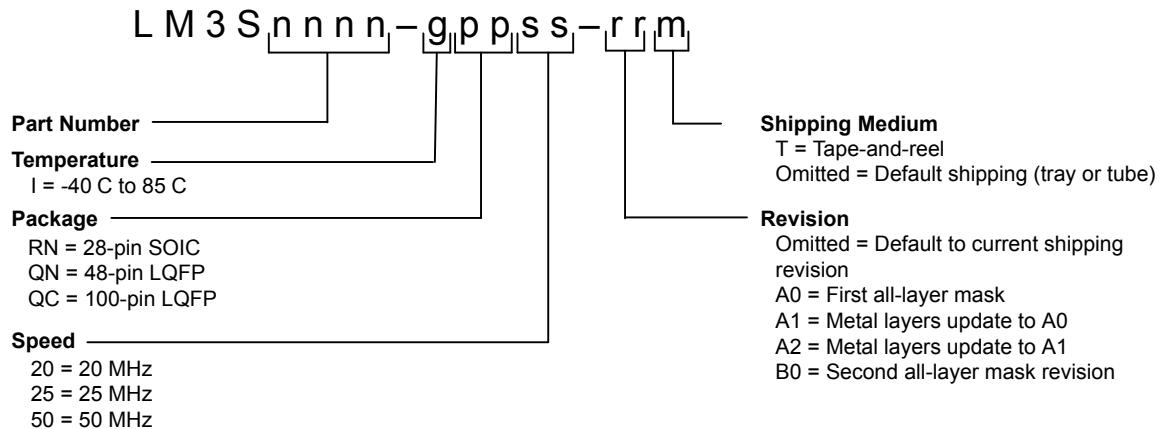
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>PWM0LOAD</b> , type R/W, offset 0x050, reset 0x0000.0000															
Load															
<b>PWM1LOAD</b> , type R/W, offset 0x090, reset 0x0000.0000															
Load															
<b>PWM2LOAD</b> , type R/W, offset 0x0D0, reset 0x0000.0000															
Load															
<b>PWM0COUNT</b> , type RO, offset 0x054, reset 0x0000.0000															
Count															
<b>PWM1COUNT</b> , type RO, offset 0x094, reset 0x0000.0000															
Count															
<b>PWM2COUNT</b> , type RO, offset 0x0D4, reset 0x0000.0000															
Count															
<b>PWM0CMPA</b> , type R/W, offset 0x058, reset 0x0000.0000															
CompA															
<b>PWM1CMPA</b> , type R/W, offset 0x098, reset 0x0000.0000															
CompA															
<b>PWM2CMPA</b> , type R/W, offset 0x0D8, reset 0x0000.0000															
CompA															
<b>PWM0CMPB</b> , type R/W, offset 0x05C, reset 0x0000.0000															
CompB															
<b>PWM1CMPB</b> , type R/W, offset 0x09C, reset 0x0000.0000															
CompB															
<b>PWM2CMPB</b> , type R/W, offset 0x0DC, reset 0x0000.0000															
CompB															
<b>PWM0GENA</b> , type R/W, offset 0x060, reset 0x0000.0000															
ActCmpBD															
<b>PWM1GENA</b> , type R/W, offset 0x0A0, reset 0x0000.0000															
ActCmpBD															
<b>PWM2GENA</b> , type R/W, offset 0xE0, reset 0x0000.0000															
ActCmpBD															
<b>PWM0GENB</b> , type R/W, offset 0x064, reset 0x0000.0000															
ActCmpBD															
<b>PWM1GENB</b> , type R/W, offset 0x0A4, reset 0x0000.0000															
ActCmpBD															

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>PWM2GENB, type R/W, offset 0x0E4, reset 0x0000.0000</b>															
				ActCmpBD	ActCmpBU	ActCmpAD	ActCmpAU		ActLoad		ActZero				
<b>PWM0DBCTL, type R/W, offset 0x068, reset 0x0000.0000</b>															
															Enable
<b>PWM1DBCTL, type R/W, offset 0x0A8, reset 0x0000.0000</b>															
															Enable
<b>PWM2DBCTL, type R/W, offset 0x0E8, reset 0x0000.0000</b>															
															Enable
<b>PWM0DBRISE, type R/W, offset 0x06C, reset 0x0000.0000</b>															
															RiseDelay
<b>PWM1DBRISE, type R/W, offset 0x0AC, reset 0x0000.0000</b>															
															RiseDelay
<b>PWM2DBRISE, type R/W, offset 0x0EC, reset 0x0000.0000</b>															
															RiseDelay
<b>PWM0DBFALL, type R/W, offset 0x070, reset 0x0000.0000</b>															
															FallDelay
<b>PWM1DBFALL, type R/W, offset 0x0B0, reset 0x0000.0000</b>															
															FallDelay
<b>PWM2DBFALL, type R/W, offset 0x0F0, reset 0x0000.0000</b>															
															FallDelay
<b>Quadrature Encoder Interface (QEI)</b>															
QEIO base: 0x4002.C000															
QEI1 base: 0x4002.D000															
<b>QEICTL, type R/W, offset 0x000, reset 0x0000.0000</b>															
			STALLEN	INVI	INVB	INVA		VelDiv		VelEn	ResMode	CapMode	SigMode	Swap	Enable
<b>QEISTAT, type RO, offset 0x004, reset 0x0000.0000</b>															
															Direction Error
<b>QEIPOS, type R/W, offset 0x008, reset 0x0000.0000</b>															
		Position													
<b>QEIMAXPOS, type R/W, offset 0x00C, reset 0x0000.0000</b>															
		MaxPos													
<b>QEILOAD, type R/W, offset 0x010, reset 0x0000.0000</b>															
		Load													
<b>QEITIME, type RO, offset 0x014, reset 0x0000.0000</b>															
		Time													

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>QEICOUNT</b> , type RO, offset 0x018, reset 0x0000.0000																	
Count																	
Count																	
<b>QEISPEED</b> , type RO, offset 0x01C, reset 0x0000.0000																	
Speed																	
Speed																	
<b>QEINTEN</b> , type R/W, offset 0x020, reset 0x0000.0000																	
														IntError	IntDir	IntTimer	IntIndex
<b>QEIRIS</b> , type RO, offset 0x024, reset 0x0000.0000																	
														IntError	IntDir	IntTimer	IntIndex
<b>QEISCI</b> , type R/W1C, offset 0x028, reset 0x0000.0000																	
														IntError	IntDir	IntTimer	IntIndex

## C Ordering and Contact Information

### C.1 Ordering Information



**Table C-1. Part Ordering Information**

Orderable Part Number	Description
LM3S8962-IQC50	Stellaris® LM3S8962 Microcontroller
LM3S8962-IQC50(T)	Stellaris® LM3S8962 Microcontroller

### C.2 Kits

The Luminary Micro Stellaris® Family provides the hardware and software tools that engineers need to begin development quickly.

- Reference Design Kits accelerate product development by providing ready-to-run hardware, and comprehensive documentation including hardware design files:  
[http://www.luminarmicro.com/products/reference\\_design\\_kits/](http://www.luminarmicro.com/products/reference_design_kits/)
- Evaluation Kits provide a low-cost and effective means of evaluating Stellaris® microcontrollers before purchase:  
[http://www.luminarmicro.com/products/evaluation\\_kits/](http://www.luminarmicro.com/products/evaluation_kits/)
- Development Kits provide you with all the tools you need to develop and prototype embedded applications right out of the box:  
<http://www.luminarmicro.com/products/boards.html>

See the Luminary Micro website for the latest tools available or ask your Luminary Micro distributor.

### C.3 Company Information

Luminary Micro, Inc. designs, markets, and sells ARM Cortex-M3-based microcontrollers (MCUs). Austin, Texas-based Luminary Micro is the lead partner for the Cortex-M3 processor, delivering the world's first silicon implementation of the Cortex-M3 processor. Luminary Micro's introduction of the

Stellaris® family of products provides 32-bit performance for the same price as current 8- and 16-bit microcontroller designs. With entry-level pricing at \$1.00 for an ARM technology-based MCU, Luminary Micro's Stellaris product line allows for standardization that eliminates future architectural upgrades or software tool changes.

Luminary Micro, Inc.  
108 Wild Basin, Suite 350  
Austin, TX 78746  
Main: +1-512-279-8800  
Fax: +1-512-279-8879  
<http://www.luminarmicro.com>  
[sales@luminarmicro.com](mailto:sales@luminarmicro.com)

## C.4 Support Information

For support on Luminary Micro products, contact:  
[support@luminarmicro.com](mailto:support@luminarmicro.com) +1-512-279-8800, ext. 3