



**LUMINARY** MICRO®

---

# LM3S5791 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 © 2007-2009 Luminary Micro, Inc. All rights reserved. Stellaris, Luminary Micro, and the Luminary Micro logo are registered trademarks of Luminary Micro, Inc. or its subsidiaries in the United States and other countries. ARM and Thumb are registered trademarks and Cortex is a trademark of ARM Limited. Other names and brands may be claimed as the property of others.

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



**LUMINARY** MICRO



**Cortex**  
Intelligent Processors by ARM<sup>®</sup>

# Table of Contents

<b>About This Document .....</b>	<b>29</b>
Audience .....	29
About This Manual .....	29
Related Documents .....	29
Documentation Conventions .....	29
<b>1 Architectural Overview .....</b>	<b>32</b>
1.1 Functional Overview .....	34
1.1.1 ARM Cortex™-M3 .....	34
1.1.2 On-Chip Memory .....	35
1.1.3 External Peripheral Interface .....	37
1.1.4 Serial Communications Peripherals .....	38
1.1.5 System Integration .....	43
1.1.6 Advanced Motion Control .....	48
1.1.7 Analog .....	50
1.1.8 JTAG and ARM Serial Wire Debug .....	51
1.1.9 Packaging and Temperature .....	52
1.2 Target Applications .....	52
1.3 High-Level Block Diagram .....	52
1.4 Additional Features .....	54
1.4.1 Memory Map .....	54
1.4.2 Hardware Details .....	54
<b>2 ARM Cortex-M3 Processor Core .....</b>	<b>55</b>
2.1 Block Diagram .....	56
2.2 Functional Description .....	56
2.2.1 Programming Model .....	56
2.2.2 Serial Wire and JTAG Debug .....	63
2.2.3 Embedded Trace Macrocell (ETM) .....	63
2.2.4 Trace Port Interface Unit (TPIU) .....	64
2.2.5 ROM Table .....	64
2.2.6 Memory Protection Unit (MPU) .....	64
2.2.7 Nested Vectored Interrupt Controller (NVIC) .....	64
2.2.8 System Timer (SysTick) .....	65
<b>3 Memory Map .....</b>	<b>68</b>
<b>4 Interrupts .....</b>	<b>71</b>
<b>5 JTAG Interface .....</b>	<b>74</b>
5.1 Block Diagram .....	75
5.2 Functional Description .....	75
5.2.1 JTAG Interface Pins .....	76
5.2.2 JTAG TAP Controller .....	77
5.2.3 Shift Registers .....	78
5.2.4 Operational Considerations .....	78
5.3 Initialization and Configuration .....	81
5.4 Register Descriptions .....	81
5.4.1 Instruction Register (IR) .....	81

5.4.2	Data Registers .....	83
<b>6</b>	<b>System Control .....</b>	<b>86</b>
6.1	Functional Description .....	86
6.1.1	Device Identification .....	86
6.1.2	Reset Control .....	86
6.1.3	Non-Maskable Interrupt .....	89
6.1.4	Power Control .....	90
6.1.5	Clock Control .....	90
6.1.6	System Control .....	95
6.2	Initialization and Configuration .....	95
6.3	Register Map .....	96
6.4	Register Descriptions .....	97
<b>7</b>	<b>Internal Memory .....</b>	<b>188</b>
7.1	Block Diagram .....	188
7.2	Functional Description .....	189
7.2.1	SRAM .....	189
7.2.2	ROM .....	189
7.2.3	Flash Memory .....	189
7.3	Flash Memory Initialization and Configuration .....	190
7.3.1	Flash Programming .....	190
7.3.2	32-Word Flash Write Buffer .....	191
7.3.3	Nonvolatile Register Programming .....	192
7.4	Register Map .....	192
7.5	Flash Register Descriptions (Flash Control Offset) .....	193
7.6	Memory Register Descriptions (System Control Offset) .....	203
<b>8</b>	<b>Micro Direct Memory Access (μDMA) .....</b>	<b>219</b>
8.1	Block Diagram .....	220
8.2	Functional Description .....	220
8.2.1	Channel Assignments .....	221
8.2.2	Priority .....	222
8.2.3	Arbitration Size .....	222
8.2.4	Request Types .....	222
8.2.5	Channel Configuration .....	223
8.2.6	Transfer Modes .....	224
8.2.7	Transfer Size and Increment .....	233
8.2.8	Peripheral Interface .....	233
8.2.9	Software Request .....	233
8.2.10	Interrupts and Errors .....	234
8.3	Initialization and Configuration .....	234
8.3.1	Module Initialization .....	234
8.3.2	Configuring a Memory-to-Memory Transfer .....	234
8.3.3	Configuring a Peripheral for Simple Transmit .....	236
8.3.4	Configuring a Peripheral for Ping-Pong Receive .....	237
8.3.5	Configuring Alternate Channels .....	240
8.4	Register Map .....	240
8.5	μDMA Channel Control Structure .....	241
8.6	μDMA Register Descriptions .....	248



<b>9</b>	<b>General-Purpose Input/Outputs (GPIOs)</b>	<b>284</b>
9.1	Functional Description	284
9.1.1	Data Control	286
9.1.2	Interrupt Control	287
9.1.3	Mode Control	288
9.1.4	Commit Control	288
9.1.5	Pad Control	289
9.1.6	Identification	289
9.2	Initialization and Configuration	289
9.3	Register Map	290
9.4	Register Descriptions	293
<b>10</b>	<b>External Peripheral Interface (EPI)</b>	<b>335</b>
10.1	EPI Block Diagram	336
10.2	Functional Description	337
10.2.1	Non-blocking reads	337
10.2.2	DMA Operation	338
10.3	Initialization and Configuration	338
10.3.1	SDRAM mode	340
10.3.2	Host Bus Mode	341
10.3.3	General-Purpose Mode	343
10.4	Register Map	345
10.5	Register Descriptions	346
<b>11</b>	<b>General-Purpose Timers</b>	<b>381</b>
11.1	Block Diagram	381
11.2	Functional Description	382
11.2.1	GPTM Reset Conditions	383
11.2.2	32-Bit Timer Operating Modes	383
11.2.3	16-Bit Timer Operating Modes	384
11.2.4	DMA Operation	389
11.3	Initialization and Configuration	389
11.3.1	32-Bit One-Shot/Periodic Timer Mode	389
11.3.2	32-Bit Real-Time Clock (RTC) Mode	390
11.3.3	16-Bit One-Shot/Periodic Timer Mode	390
11.3.4	16-Bit Input Edge-Count Mode	391
11.3.5	16-Bit Input Edge Timing Mode	392
11.3.6	16-Bit PWM Mode	392
11.4	Register Map	393
11.5	Register Descriptions	393
<b>12</b>	<b>Watchdog Timer</b>	<b>420</b>
12.1	Block Diagram	421
12.2	Functional Description	421
12.2.1	Register Access Timing	422
12.3	Initialization and Configuration	422
12.4	Register Map	422
12.5	Register Descriptions	423
<b>13</b>	<b>Analog-to-Digital Converter (ADC)</b>	<b>445</b>
13.1	Block Diagram	446

13.2	Functional Description .....	447
13.2.1	Sample Sequencers .....	447
13.2.2	Module Control .....	448
13.2.3	Hardware Sample Averaging Circuit .....	449
13.2.4	Analog-to-Digital Converter .....	449
13.2.5	Differential Sampling .....	451
13.2.6	Internal Temperature Sensor .....	454
13.2.7	Digital Comparator Unit .....	454
13.3	Initialization and Configuration .....	458
13.3.1	Module Initialization .....	458
13.3.2	Sample Sequencer Configuration .....	458
13.4	Register Map .....	459
13.5	Register Descriptions .....	461
<b>14</b>	<b>Universal Asynchronous Receivers/Transmitters (UARTs) .....</b>	<b>513</b>
14.1	Block Diagram .....	514
14.2	Functional Description .....	514
14.2.1	Transmit/Receive Logic .....	514
14.2.2	Baud-Rate Generation .....	515
14.2.3	Data Transmission .....	516
14.2.4	Serial IR (SIR) .....	516
14.2.5	ISO 7816 Support .....	517
14.2.6	LIN Support .....	517
14.2.7	FIFO Operation .....	518
14.2.8	Interrupts .....	518
14.2.9	Loopback Operation .....	519
14.2.10	DMA Operation .....	519
14.2.11	IrDA SIR block .....	520
14.3	Initialization and Configuration .....	520
14.4	Register Map .....	521
14.5	Register Descriptions .....	522
<b>15</b>	<b>Synchronous Serial Interface (SSI) .....</b>	<b>563</b>
15.1	Block Diagram .....	564
15.2	Functional Description .....	564
15.2.1	Bit Rate Generation .....	565
15.2.2	FIFO Operation .....	565
15.2.3	Interrupts .....	565
15.2.4	Frame Formats .....	566
15.2.5	DMA Operation .....	573
15.3	Initialization and Configuration .....	574
15.4	Register Map .....	575
15.5	Register Descriptions .....	576
<b>16</b>	<b>Inter-Integrated Circuit (I<sup>2</sup>C) Interface .....</b>	<b>603</b>
16.1	Block Diagram .....	604
16.2	Functional Description .....	604
16.2.1	I <sup>2</sup> C Bus Functional Overview .....	604
16.2.2	Available Speed Modes .....	606
16.2.3	Interrupts .....	607
16.2.4	Loopback Operation .....	608

16.2.5	Command Sequence Flow Charts .....	608
16.3	Initialization and Configuration .....	615
16.4	Register Map .....	616
16.5	Register Descriptions (I <sup>2</sup> C Master) .....	617
16.6	Register Descriptions (I <sup>2</sup> C Slave) .....	629
<b>17</b>	<b>Inter-Integrated Circuit Sound (I<sup>2</sup>S) Interface .....</b>	<b>638</b>
17.1	Block Diagram .....	639
17.2	Functional Description .....	639
17.2.1	Transmit .....	641
17.2.2	Receive .....	645
17.3	Initialization and Configuration .....	647
17.4	Register Map .....	648
17.5	Register Descriptions .....	649
<b>18</b>	<b>Controller Area Network (CAN) Module .....</b>	<b>670</b>
18.1	Block Diagram .....	671
18.2	Functional Description .....	671
18.2.1	Initialization .....	672
18.2.2	Operation .....	673
18.2.3	Transmitting Message Objects .....	674
18.2.4	Configuring a Transmit Message Object .....	674
18.2.5	Updating a Transmit Message Object .....	675
18.2.6	Accepting Received Message Objects .....	676
18.2.7	Receiving a Data Frame .....	676
18.2.8	Receiving a Remote Frame .....	676
18.2.9	Receive/Transmit Priority .....	677
18.2.10	Configuring a Receive Message Object .....	677
18.2.11	Handling of Received Message Objects .....	678
18.2.12	Handling of Interrupts .....	681
18.2.13	Test Mode .....	681
18.2.14	Bit Timing Configuration Error Considerations .....	683
18.2.15	Bit Time and Bit Rate .....	683
18.2.16	Calculating the Bit Timing Parameters .....	685
18.3	Register Map .....	687
18.4	CAN Register Descriptions .....	688
<b>19</b>	<b>Universal Serial Bus (USB) Controller .....</b>	<b>716</b>
19.1	Block Diagram .....	717
19.2	Functional Description .....	717
19.2.1	Operation as a Device .....	717
19.2.2	Operation as a Host .....	723
19.2.3	OTG Mode .....	727
19.2.4	DMA Operation .....	728
19.3	Initialization and Configuration .....	729
19.3.1	Pin Configuration .....	730
19.3.2	Endpoint Configuration .....	730
19.4	Register Map .....	730
19.5	Register Descriptions .....	734

<b>20</b>	<b>Analog Comparators .....</b>	<b>816</b>
20.1	Block Diagram .....	817
20.2	Functional Description .....	817
20.2.1	Internal Reference Programming .....	818
20.3	Initialization and Configuration .....	819
20.4	Register Map .....	819
20.5	Register Descriptions .....	820
<b>21</b>	<b>Pulse Width Modulator (PWM) .....</b>	<b>828</b>
21.1	Block Diagram .....	829
21.2	Functional Description .....	830
21.2.1	PWM Timer .....	830
21.2.2	PWM Comparators .....	830
21.2.3	PWM Signal Generator .....	831
21.2.4	Dead-Band Generator .....	832
21.2.5	Interrupt/ADC-Trigger Selector .....	832
21.2.6	Synchronization Methods .....	833
21.2.7	Fault Conditions .....	834
21.2.8	Output Control Block .....	835
21.3	Initialization and Configuration .....	835
21.4	Register Map .....	836
21.5	Register Descriptions .....	838
<b>22</b>	<b>Quadrature Encoder Interface (QEI) .....</b>	<b>887</b>
22.1	Block Diagram .....	887
22.2	Functional Description .....	888
22.3	Initialization and Configuration .....	890
22.4	Register Map .....	891
22.5	Register Descriptions .....	891
<b>23</b>	<b>Pin Diagram .....</b>	<b>904</b>
<b>24</b>	<b>Signal Tables .....</b>	<b>905</b>
<b>25</b>	<b>Operating Characteristics .....</b>	<b>940</b>
<b>26</b>	<b>Electrical Characteristics .....</b>	<b>941</b>
26.1	DC Characteristics .....	941
26.1.1	Maximum Ratings .....	941
26.1.2	Recommended DC Operating Conditions .....	941
26.1.3	On-Chip Low Drop-Out (LDO) Regulator Characteristics .....	942
26.1.4	Flash Memory Characteristics .....	942
26.1.5	GPIO Module Characteristics .....	942
26.1.6	USB Module Characteristics .....	943
26.1.7	Current Specifications .....	943
26.2	AC Characteristics .....	946
26.2.1	Load Conditions .....	946
26.2.2	Clocks .....	947
26.2.3	JTAG and Boundary Scan .....	949
26.2.4	Reset .....	950
26.2.5	General-Purpose I/O (GPIO) .....	952
26.2.6	External Peripheral Interface (EPI) .....	952
26.2.7	Analog-to-Digital Converter .....	960

26.2.8	Synchronous Serial Interface (SSI)	961
26.2.9	Inter-Integrated Circuit (I <sup>2</sup> C) Interface	962
26.2.10	Inter-Integrated Circuit Sound (I <sup>2</sup> S) Interface	963
26.2.11	Universal Serial Bus (USB) Controller	964
26.2.12	Analog Comparator	964
<b>27</b>	<b>Package Information</b>	<b>965</b>
<b>A</b>	<b>Boot Loader</b>	<b>967</b>
A.1	Boot Loader	967
A.2	Interfaces	967
A.2.1	UART	967
A.2.2	SSI	968
A.2.3	I <sup>2</sup> C	968
A.3	Packet Handling	968
A.3.1	Packet Format	968
A.3.2	Sending Packets	968
A.3.3	Receiving Packets	969
A.4	Commands	969
A.4.1	COMMAND_PING (0x20)	969
A.4.2	COMMAND_DOWNLOAD (0x21)	969
A.4.3	COMMAND_RUN (0x22)	970
A.4.4	COMMAND_GET_STATUS (0x23)	970
A.4.5	COMMAND_SEND_DATA (0x24)	970
A.4.6	COMMAND_RESET (0x25)	971
<b>B</b>	<b>ROM DriverLib Functions</b>	<b>972</b>
B.1	DriverLib Functions Included in the Integrated ROM	972
<b>C</b>	<b>Advance Encryption Standard and Cyclic Redundancy Check Software in ROM</b>	<b>991</b>
C.1	Advanced Encryption Standard Software	991
C.2	Cyclic Redundancy Check Software	991
<b>D</b>	<b>Register Quick Reference</b>	<b>992</b>
<b>E</b>	<b>Ordering and Contact Information</b>	<b>1028</b>
E.1	Ordering Information	1028
E.2	Kits	1028
E.3	Company Information	1029
E.4	Support Information	1029

## List of Figures

Figure 1-1.	Stellaris® LM3S5791 Microcontroller High-Level Block Diagram .....	53
Figure 2-1.	CPU Block Diagram .....	56
Figure 2-2.	TPIU Block Diagram .....	64
Figure 5-1.	JTAG Module Block Diagram .....	75
Figure 5-2.	Test Access Port State Machine .....	78
Figure 5-3.	IDCODE Register Format .....	84
Figure 5-4.	BYPASS Register Format .....	84
Figure 5-5.	Boundary Scan Register Format .....	84
Figure 6-1.	External Circuitry to Extend Reset .....	87
Figure 6-2.	Power Architecture .....	90
Figure 6-3.	Main Clock Tree .....	92
Figure 7-1.	Flash Block Diagram .....	188
Figure 8-1.	μDMA Block Diagram .....	220
Figure 8-2.	Example of Ping-Pong DMA Transaction .....	226
Figure 8-3.	Memory Scatter-Gather, Setup and Configuration .....	228
Figure 8-4.	Memory Scatter-Gather, μDMA Copy Sequence .....	229
Figure 8-5.	Peripheral Scatter-Gather, Setup and Configuration .....	231
Figure 8-6.	Peripheral Scatter-Gather, μDMA Copy Sequence .....	232
Figure 9-1.	Digital I/O Pads .....	285
Figure 9-2.	Analog/Digital I/O Pads .....	286
Figure 9-3.	GPIO DATA Write Example .....	287
Figure 9-4.	GPIO DATA Read Example .....	287
Figure 10-1.	EPI Block Diagram .....	336
Figure 11-1.	GPTM Module Block Diagram .....	382
Figure 11-2.	16-Bit Input Edge-Count Mode Example .....	387
Figure 11-3.	16-Bit Input Edge-Time Mode Example .....	388
Figure 11-4.	16-Bit PWM Mode Example .....	389
Figure 12-1.	WDT Module Block Diagram .....	421
Figure 13-1.	Implementation of Two ADC Blocks .....	446
Figure 13-2.	ADC Module Block Diagram .....	446
Figure 13-3.	Internal Voltage Conversion Result .....	450
Figure 13-4.	External Voltage Conversion Result .....	451
Figure 13-5.	Differential Sampling Range, $V_{IN\_ODD} = 1.5\text{ V}$ .....	452
Figure 13-6.	Differential Sampling Range, $V_{IN\_ODD} = 0.75\text{ V}$ .....	453
Figure 13-7.	Differential Sampling Range, $V_{IN\_ODD} = 2.25\text{ V}$ .....	453
Figure 13-8.	Internal Temperature Sensor Characteristic .....	454
Figure 13-9.	Low-Band Operation (CIC=0x0 and/or CTC=0x0) .....	456
Figure 13-10.	Mid-Band Operation (CIC=0x1 and/or CTC=0x1) .....	457
Figure 13-11.	High-Band Operation (CIC=0x3 and/or CTC=0x3) .....	458
Figure 14-1.	UART Module Block Diagram .....	514
Figure 14-2.	UART Character Frame .....	515
Figure 14-3.	IrDA Data Modulation .....	517
Figure 15-1.	SSI Module Block Diagram .....	564
Figure 15-2.	TI Synchronous Serial Frame Format (Single Transfer) .....	567
Figure 15-3.	TI Synchronous Serial Frame Format (Continuous Transfer) .....	567

Figure 15-4.	Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0 .....	568
Figure 15-5.	Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0 .....	568
Figure 15-6.	Freescale SPI Frame Format with SPO=0 and SPH=1 .....	569
Figure 15-7.	Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0 .....	570
Figure 15-8.	Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0 .....	570
Figure 15-9.	Freescale SPI Frame Format with SPO=1 and SPH=1 .....	571
Figure 15-10.	MICROWIRE Frame Format (Single Frame) .....	572
Figure 15-11.	MICROWIRE Frame Format (Continuous Transfer) .....	573
Figure 15-12.	MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements .....	573
Figure 16-1.	I <sup>2</sup> C Block Diagram .....	604
Figure 16-2.	I <sup>2</sup> C Bus Configuration .....	604
Figure 16-3.	START and STOP Conditions .....	605
Figure 16-4.	Complete Data Transfer with a 7-Bit Address .....	605
Figure 16-5.	R/S Bit in First Byte .....	605
Figure 16-6.	Data Validity During Bit Transfer on the I <sup>2</sup> C Bus .....	606
Figure 16-7.	Master Single SEND .....	609
Figure 16-8.	Master Single RECEIVE .....	610
Figure 16-9.	Master Burst SEND .....	611
Figure 16-10.	Master Burst RECEIVE .....	612
Figure 16-11.	Master Burst RECEIVE after Burst SEND .....	613
Figure 16-12.	Master Burst SEND after Burst RECEIVE .....	614
Figure 16-13.	Slave Command Sequence .....	615
Figure 17-1.	I <sup>2</sup> S Block Diagram .....	639
Figure 17-2.	I <sup>2</sup> S Data Transfer .....	641
Figure 17-3.	Left-Justified Data Transfer .....	641
Figure 17-4.	Right-Justified Data Transfer .....	641
Figure 18-1.	CAN Controller Block Diagram .....	671
Figure 18-2.	CAN Data/Remote Frame .....	672
Figure 18-3.	Message Objects in a FIFO Buffer .....	680
Figure 18-4.	CAN Bit Time .....	684
Figure 19-1.	USB Module Block Diagram .....	717
Figure 20-1.	Analog Comparator Module Block Diagram .....	817
Figure 20-2.	Structure of Comparator Unit .....	818
Figure 20-3.	Comparator Internal Reference Structure .....	818
Figure 21-1.	PWM Unit Diagram .....	829
Figure 21-2.	PWM Module Block Diagram .....	830
Figure 21-3.	PWM Count-Down Mode .....	831
Figure 21-4.	PWM Count-Up/Down Mode .....	831
Figure 21-5.	PWM Generation Example In Count-Up/Down Mode .....	832
Figure 21-6.	PWM Dead-Band Generator .....	832
Figure 22-1.	QEI Block Diagram .....	888
Figure 22-2.	Quadrature Encoder and Velocity Predivider Operation .....	889
Figure 23-1.	100-Pin LQFP Package Pin Diagram .....	904
Figure 26-1.	Typical Current Across Frequency .....	945
Figure 26-2.	Typical Current Across Temperature .....	946
Figure 26-3.	Load Conditions .....	947
Figure 26-4.	JTAG Test Clock Input Timing .....	949
Figure 26-5.	JTAG Test Access Port (TAP) Timing .....	950

Figure 26-6. External Reset Timing ( $\overline{\text{RST}}$ ) .....	950
Figure 26-7. Power-On Reset Timing .....	951
Figure 26-8. Brown-Out Reset Timing .....	951
Figure 26-9. Software Reset Timing .....	951
Figure 26-10. Watchdog Reset Timing .....	951
Figure 26-11. MOSC Failure Reset Timing .....	952
Figure 26-12. SDRAM Initialization and Load Mode Register Timing .....	953
Figure 26-13. SDRAM Read Command Timing .....	954
Figure 26-14. SDRAM Write Command Timing .....	955
Figure 26-15. SDRAM Write Burst Timing .....	956
Figure 26-16. SDRAM Precharge Command Timing .....	957
Figure 26-17. SDRAM CAS Latency Timing .....	958
Figure 26-18. SDRAM Active Row Bank Timing .....	959
Figure 26-19. SRAM Nor Read Timing .....	959
Figure 26-20. General-Purpose Mode Read Timing .....	960
Figure 26-21. General-Purpose Mode Write Timing .....	960
Figure 26-22. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement .....	961
Figure 26-23. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer .....	962
Figure 26-24. SSI Timing for SPI Frame Format (FRF=00), with SPH=1 .....	962
Figure 26-25. I <sup>2</sup> C Timing .....	962
Figure 27-1. 100-Pin LQFP Package .....	965



## List of Tables

Table 1.	Documentation Conventions .....	29
Table 2-1.	16-Bit Cortex-M3 Instruction Set Summary .....	57
Table 2-2.	32-Bit Cortex-M3 Instruction Set Summary .....	59
Table 3-1.	Memory Map .....	68
Table 4-1.	Exception Types .....	71
Table 4-2.	Interrupts .....	72
Table 5-1.	JTAG Port Pins State after Power-On Reset or $\overline{\text{RST}}$ assertion .....	76
Table 5-2.	JTAG Instruction Register Commands .....	81
Table 6-1.	Reset Sources .....	86
Table 6-2.	Clock Source Options .....	91
Table 6-3.	System Control Register Map .....	96
Table 6-4.	Examples of Possible System Clock Frequencies .....	117
Table 7-1.	Flash Protection Policy Combinations .....	190
Table 7-2.	User-Programmable Flash Resident Registers .....	192
Table 7-3.	Flash Register Map .....	193
Table 8-1.	DMA Channel Assignments .....	221
Table 8-2.	Request Type Support .....	222
Table 8-3.	Control Structure Memory Map .....	223
Table 8-4.	Channel Control Structure .....	224
Table 8-5.	$\mu$ DMA Read Example: 8-Bit Peripheral .....	233
Table 8-6.	$\mu$ DMA Interrupt Assignments .....	234
Table 8-7.	Channel Control Structure Offsets for Channel 30 .....	235
Table 8-8.	Channel Control Word Configuration for Memory Transfer Example .....	235
Table 8-9.	Channel Control Structure Offsets for Channel 7 .....	236
Table 8-10.	Channel Control Word Configuration for Peripheral Transmit Example .....	237
Table 8-11.	Primary and Alternate Channel Control Structure Offsets for Channel 8 .....	238
Table 8-12.	Channel Control Word Configuration for Peripheral Ping-Pong Receive Example .....	239
Table 8-13.	$\mu$ DMA Register Map .....	240
Table 9-1.	GPIO Pins With Non-Zero Reset Values .....	285
Table 9-2.	GPIO Pad Configuration Examples .....	289
Table 9-3.	GPIO Interrupt Configuration Example .....	290
Table 9-4.	GPIO Pins With Non-Zero Reset Values .....	291
Table 9-5.	GPIO Register Map .....	292
Table 9-6.	GPIO Pins With Non-Zero Reset Values .....	304
Table 9-7.	GPIO Pins With Non-Zero Reset Values .....	310
Table 9-8.	GPIO Pins With Non-Zero Reset Values .....	321
Table 10-1.	EPI Signal Connections .....	339
Table 10-2.	External Peripheral Interface (EPI) Register Map .....	345
Table 11-1.	Available CCP Pins .....	382
Table 11-2.	16-Bit Timer With Prescaler Configurations .....	385
Table 11-3.	Timers Register Map .....	393
Table 12-1.	Watchdog Timer Register Map .....	423
Table 13-1.	Samples and FIFO Depth of Sequencers .....	447
Table 13-2.	Differential Sampling Pairs .....	451
Table 13-3.	ADC Register Map .....	459
Table 14-1.	UART Register Map .....	521

Table 15-1.	SSI Register Map .....	575
Table 16-1.	Examples of I <sup>2</sup> C Master Timer Period versus Speed Mode .....	607
Table 16-2.	Inter-Integrated Circuit (I <sup>2</sup> C) Interface Register Map .....	616
Table 16-3.	Write Field Decoding for I2CMCS[3:0] Field (Sheet 1 of 3) .....	621
Table 17-1.	I <sup>2</sup> S Transmit FIFO Interface .....	642
Table 17-2.	Crystal Frequency (Values from 3.5795 MHz to 5 MHz) .....	643
Table 17-3.	Crystal Frequency (Values from 5.12 MHz to 8.192 MHz) .....	643
Table 17-4.	Crystal Frequency (Values from 10 MHz to 14.3181 MHz) .....	644
Table 17-5.	Crystal Frequency (Values from 16 MHz to 16.384 MHz) .....	644
Table 17-6.	I <sup>2</sup> S Receive FIFO Interface .....	646
Table 17-7.	Audio Formats Configuration .....	648
Table 17-8.	Inter-Integrated Circuit Sound (I <sup>2</sup> S) Interface Register Map .....	648
Table 18-1.	CAN Protocol Ranges .....	684
Table 18-2.	CAN Register Map .....	687
Table 19-1.	Remainder (RxMaxP/4) .....	729
Table 19-2.	Actual Bytes Read .....	729
Table 19-3.	Packet Sizes That Will Clear RXRDY .....	729
Table 19-4.	Universal Serial Bus (USB) Controller Register Map .....	731
Table 20-1.	Internal Reference Voltage and ACREFCTL Field Values .....	818
Table 20-2.	Analog Comparators Register Map .....	820
Table 21-1.	PWM Register Map .....	836
Table 22-1.	QEI Register Map .....	891
Table 24-1.	GPIO Pins With Default Alternate Functions .....	905
Table 24-2.	Signals by Pin Number .....	905
Table 24-3.	Signals by Signal Name .....	917
Table 24-4.	Signals by Function, Except for GPIO .....	928
Table 24-5.	GPIO Pins and Alternate Functions .....	937
Table 25-1.	Temperature Characteristics .....	940
Table 25-2.	Thermal Characteristics .....	940
Table 25-3.	ESD Absolute Maximum Ratings .....	940
Table 26-1.	Maximum Ratings .....	941
Table 26-2.	Recommended DC Operating Conditions .....	941
Table 26-3.	LDO Regulator Characteristics .....	942
Table 26-4.	Flash Memory Characteristics .....	942
Table 26-5.	GPIO Module DC Characteristics .....	942
Table 26-6.	USB Controller DC Characteristics .....	943
Table 26-7.	Detailed Current Specifications .....	944
Table 26-8.	Typical Peripheral Current Consumption .....	946
Table 26-9.	Phase Locked Loop (PLL) Characteristics .....	947
Table 26-10.	Actual PLL Frequency .....	947
Table 26-11.	PIOSC Clock Characteristics .....	948
Table 26-12.	30-kHz Clock Characteristics .....	948
Table 26-13.	Main Oscillator Clock Characteristics .....	948
Table 26-14.	MOSC Oscillator Input Characteristics .....	948
Table 26-15.	JTAG Characteristics .....	949
Table 26-16.	Reset Characteristics .....	950
Table 26-17.	GPIO Characteristics .....	952
Table 26-18.	EPI Characteristics .....	952

---

Table 26-19.	ADC Characteristics .....	960
Table 26-20.	SSI Characteristics .....	961
Table 26-21.	I2S Master Clock (Receive and Transmit) .....	963
Table 26-22.	I2S Slave Clock (Receive and Transmit) .....	963
Table 26-23.	I2S Master Mode .....	963
Table 26-24.	I2S Slave Mode .....	963
Table 26-25.	Analog Comparator Characteristics .....	964
Table 26-26.	Analog Comparator Voltage Reference Characteristics .....	964
Table E-1.	Part Ordering Information .....	1028

## List of Registers

<b>System Control .....</b>	<b>86</b>
Register 1: Device Identification 0 (DID0), offset 0x000 .....	98
Register 2: Brown-Out Reset Control (PBORCTL), offset 0x030 .....	100
Register 3: Raw Interrupt Status (RIS), offset 0x050 .....	101
Register 4: Interrupt Mask Control (IMC), offset 0x054 .....	103
Register 5: Masked Interrupt Status and Clear (MISC), offset 0x058 .....	105
Register 6: Reset Cause (RESC), offset 0x05C .....	107
Register 7: Run-Mode Clock Configuration (RCC), offset 0x060 .....	109
Register 8: XTAL to PLL Translation (PLLCFG), offset 0x064 .....	114
Register 9: GPIO Host-Bus Control (GPIOHBCTL), offset 0x06C .....	115
Register 10: Run-Mode Clock Configuration 2 (RCC2), offset 0x070 .....	117
Register 11: Main Oscillator Control (MOSCCTL), offset 0x07C .....	120
Register 12: Deep Sleep Clock Configuration (DSLPCCLKCFG), offset 0x144 .....	121
Register 13: Deep Sleep Flash Configuration (DSFLASHCFG), offset 0x14C .....	123
Register 14: Precision Internal Oscillator Calibration (PIOSCCAL), offset 0x150 .....	124
Register 15: I <sup>2</sup> S MCLK Configuration (I2SMCLKCFG), offset 0x170 .....	125
Register 16: Device Identification 1 (DID1), offset 0x004 .....	127
Register 17: Device Capabilities 0 (DC0), offset 0x008 .....	129
Register 18: Device Capabilities 1 (DC1), offset 0x010 .....	130
Register 19: Device Capabilities 2 (DC2), offset 0x014 .....	133
Register 20: Device Capabilities 3 (DC3), offset 0x018 .....	136
Register 21: Device Capabilities 4 (DC4), offset 0x01C .....	139
Register 22: Device Capabilities 5 (DC5), offset 0x020 .....	141
Register 23: Device Capabilities 6 (DC6), offset 0x024 .....	143
Register 24: Device Capabilities 7 (DC7), offset 0x028 .....	144
Register 25: Device Capabilities 8 ADC Channels (DC8), offset 0x02C .....	148
Register 26: Device Capabilities 9 ADC Digital Comparators (DC9), offset 0x190 .....	151
Register 27: Non-Volatile Memory Information (NVMSTAT), offset 0x1A0 .....	153
Register 28: Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100 .....	154
Register 29: Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110 .....	157
Register 30: Deep Sleep Mode Clock Gating Control Register 0 (DCGC0), offset 0x120 .....	160
Register 31: Run Mode Clock Gating Control Register 1 (RCGC1), offset 0x104 .....	163
Register 32: Sleep Mode Clock Gating Control Register 1 (SCGC1), offset 0x114 .....	167
Register 33: Deep-Sleep Mode Clock Gating Control Register 1 (DCGC1), offset 0x124 .....	171
Register 34: Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108 .....	175
Register 35: Sleep Mode Clock Gating Control Register 2 (SCGC2), offset 0x118 .....	177
Register 36: Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128 .....	179
Register 37: Software Reset Control 0 (SRCR0), offset 0x040 .....	181
Register 38: Software Reset Control 1 (SRCR1), offset 0x044 .....	183
Register 39: Software Reset Control 2 (SRCR2), offset 0x048 .....	186
<b>Internal Memory .....</b>	<b>188</b>
Register 1: Flash Memory Address (FMA), offset 0x000 .....	194
Register 2: Flash Memory Data (FMD), offset 0x004 .....	195
Register 3: Flash Memory Control (FMC), offset 0x008 .....	196
Register 4: Flash Controller Raw Interrupt Status (FCRIS), offset 0x00C .....	198

Register 5:	Flash Controller Interrupt Mask (FCIM), offset 0x010	199
Register 6:	Flash Controller Masked Interrupt Status and Clear (FCMISC), offset 0x014	200
Register 7:	Flash Memory Control 2 (FMC2), offset 0x020	201
Register 8:	Flash Write Buffer Valid (FWBVAL), offset 0x030	202
Register 9:	Flash Write Buffer n (FWBn), offset 0x100 - 0x13C	203
Register 10:	ROM Control (RMCTL), offset 0x0F0	204
Register 11:	ROM Version Register (RMVER), offset 0x0F4	205
Register 12:	Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x130 and 0x200	206
Register 13:	Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x134 and 0x400	207
Register 14:	User Debug (USER_DBG), offset 0x1D0	208
Register 15:	User Register 0 (USER_REG0), offset 0x1E0	209
Register 16:	User Register 1 (USER_REG1), offset 0x1E4	210
Register 17:	User Register 2 (USER_REG2), offset 0x1E8	211
Register 18:	User Register 3 (USER_REG3), offset 0x1EC	212
Register 19:	Flash Memory Protection Read Enable 1 (FMPRE1), offset 0x204	213
Register 20:	Flash Memory Protection Read Enable 2 (FMPRE2), offset 0x208	214
Register 21:	Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C	215
Register 22:	Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404	216
Register 23:	Flash Memory Protection Program Enable 2 (FMPPE2), offset 0x408	217
Register 24:	Flash Memory Protection Program Enable 3 (FMPPE3), offset 0x40C	218
<b>Micro Direct Memory Access (<math>\mu</math>DMA)</b>		<b>219</b>
Register 1:	DMA Channel Source Address End Pointer (DMASRCENDP), offset 0x000	242
Register 2:	DMA Channel Destination Address End Pointer (DMADSTENDP), offset 0x004	243
Register 3:	DMA Channel Control Word (DMACHCTL), offset 0x008	244
Register 4:	DMA Status (DMASTAT), offset 0x000	249
Register 5:	DMA Configuration (DMACFG), offset 0x004	251
Register 6:	DMA Channel Control Base Pointer (DMACTLBASE), offset 0x008	252
Register 7:	DMA Alternate Channel Control Base Pointer (DMAALTBASE), offset 0x00C	253
Register 8:	DMA Channel Wait-on-Request Status (DMAWAITSTAT), offset 0x010	254
Register 9:	DMA Channel Software Request (DMASWREQ), offset 0x014	255
Register 10:	DMA Channel Useburst Set (DMAUSEBURSTSET), offset 0x018	256
Register 11:	DMA Channel Useburst Clear (DMAUSEBURSTCLR), offset 0x01C	258
Register 12:	DMA Channel Request Mask Set (DMAREQMASKSET), offset 0x020	259
Register 13:	DMA Channel Request Mask Clear (DMAREQMASKCLR), offset 0x024	261
Register 14:	DMA Channel Enable Set (DMAENASET), offset 0x028	262
Register 15:	DMA Channel Enable Clear (DMAENACL), offset 0x02C	264
Register 16:	DMA Channel Primary Alternate Set (DMAALTSET), offset 0x030	265
Register 17:	DMA Channel Primary Alternate Clear (DMAALTCLR), offset 0x034	267
Register 18:	DMA Channel Priority Set (DMAPRIOSET), offset 0x038	268
Register 19:	DMA Channel Priority Clear (DMAPRIOCLR), offset 0x03C	270
Register 20:	DMA Bus Error Clear (DMAERRCLR), offset 0x04C	271
Register 21:	DMA Channel Alternate Select (DMACHALT), offset 0x500	273
Register 22:	DMA Channel Interrupt Status (DMACHIS), offset 0x504	274
Register 23:	DMA Peripheral Identification 0 (DMAPeriphID0), offset 0xFE0	275
Register 24:	DMA Peripheral Identification 1 (DMAPeriphID1), offset 0xFE4	276
Register 25:	DMA Peripheral Identification 2 (DMAPeriphID2), offset 0xFE8	277
Register 26:	DMA Peripheral Identification 3 (DMAPeriphID3), offset 0xFEC	278
Register 27:	DMA Peripheral Identification 4 (DMAPeriphID4), offset 0xFD0	279

Register 28:	DMA PrimeCell Identification 0 (DMAPCellID0), offset 0xFF0 .....	280
Register 29:	DMA PrimeCell Identification 1 (DMAPCellID1), offset 0xFF4 .....	281
Register 30:	DMA PrimeCell Identification 2 (DMAPCellID2), offset 0xFF8 .....	282
Register 31:	DMA PrimeCell Identification 3 (DMAPCellID3), offset 0xFFC .....	283
<b>General-Purpose Input/Outputs (GPIOs) .....</b>		<b>284</b>
Register 1:	GPIO Data (GPIODATA), offset 0x000 .....	294
Register 2:	GPIO Direction (GPIODIR), offset 0x400 .....	295
Register 3:	GPIO Interrupt Sense (GPIOIS), offset 0x404 .....	296
Register 4:	GPIO Interrupt Both Edges (GPIOIBE), offset 0x408 .....	297
Register 5:	GPIO Interrupt Event (GPIOIEV), offset 0x40C .....	298
Register 6:	GPIO Interrupt Mask (GPIOIM), offset 0x410 .....	299
Register 7:	GPIO Raw Interrupt Status (GPIORIS), offset 0x414 .....	300
Register 8:	GPIO Masked Interrupt Status (GPIOMIS), offset 0x418 .....	301
Register 9:	GPIO Interrupt Clear (GPIOICR), offset 0x41C .....	303
Register 10:	GPIO Alternate Function Select (GPIOAFSEL), offset 0x420 .....	304
Register 11:	GPIO 2-mA Drive Select (GPIODR2R), offset 0x500 .....	306
Register 12:	GPIO 4-mA Drive Select (GPIODR4R), offset 0x504 .....	307
Register 13:	GPIO 8-mA Drive Select (GPIODR8R), offset 0x508 .....	308
Register 14:	GPIO Open Drain Select (GPIODR), offset 0x50C .....	309
Register 15:	GPIO Pull-Up Select (GPIOPUR), offset 0x510 .....	310
Register 16:	GPIO Pull-Down Select (GPIOPDR), offset 0x514 .....	312
Register 17:	GPIO Slew Rate Control Select (GPIOSLR), offset 0x518 .....	313
Register 18:	GPIO Digital Enable (GPIODEN), offset 0x51C .....	314
Register 19:	GPIO Lock (GPIOLOCK), offset 0x520 .....	316
Register 20:	GPIO Commit (GPIOCR), offset 0x524 .....	317
Register 21:	GPIO Analog Mode Select (GPIOAMSEL), offset 0x528 .....	319
Register 22:	GPIO Port Control (GPIOPCTL), offset 0x52C .....	321
Register 23:	GPIO Peripheral Identification 4 (GPIOPeriphID4), offset 0xFD0 .....	323
Register 24:	GPIO Peripheral Identification 5 (GPIOPeriphID5), offset 0xFD4 .....	324
Register 25:	GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8 .....	325
Register 26:	GPIO Peripheral Identification 7 (GPIOPeriphID7), offset 0xFDC .....	326
Register 27:	GPIO Peripheral Identification 0 (GPIOPeriphID0), offset 0xFE0 .....	327
Register 28:	GPIO Peripheral Identification 1 (GPIOPeriphID1), offset 0xFE4 .....	328
Register 29:	GPIO Peripheral Identification 2 (GPIOPeriphID2), offset 0xFE8 .....	329
Register 30:	GPIO Peripheral Identification 3 (GPIOPeriphID3), offset 0xFEC .....	330
Register 31:	GPIO PrimeCell Identification 0 (GPIOPCellID0), offset 0xFF0 .....	331
Register 32:	GPIO PrimeCell Identification 1 (GPIOPCellID1), offset 0xFF4 .....	332
Register 33:	GPIO PrimeCell Identification 2 (GPIOPCellID2), offset 0xFF8 .....	333
Register 34:	GPIO PrimeCell Identification 3 (GPIOPCellID3), offset 0xFFC .....	334
<b>External Peripheral Interface (EPI) .....</b>		<b>335</b>
Register 1:	EPI Configuration (EPICFG), offset 0x000 .....	347
Register 2:	EPI Main Baud Rate (EPIBAUD), offset 0x004 .....	348
Register 3:	EPI SDRAM Configuration (EPISDRAMCFG), offset 0x010 .....	349
Register 4:	EPI Host-Bus 8 Configuration (EPIHB8CFG), offset 0x010 .....	351
Register 5:	EPI General-Purpose Configuration (EPIGPCFG), offset 0x010 .....	355
Register 6:	EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2), offset 0x014 .....	359
Register 7:	EPI General-Purpose Configuration 2 (EPIGPCFG2), offset 0x014 .....	361
Register 8:	EPI Address Map (EPIADDRMAP), offset 0x01C .....	362

Register 9:	EPI Read Size 0 (EPIRSIZE0), offset 0x020 .....	364
Register 10:	EPI Read Size 1 (EPIRSIZE1), offset 0x030 .....	364
Register 11:	EPI Read Address 0 (EPIRADDR0), offset 0x024 .....	365
Register 12:	EPI Read Address 1 (EPIRADDR1), offset 0x034 .....	365
Register 13:	EPI Non-Blocking Read Data 0 (EPIRPSTD0), offset 0x028 .....	366
Register 14:	EPI Non-Blocking Read Data 1 (EPIRPSTD1), offset 0x038 .....	366
Register 15:	EPI Status (EPISTAT), offset 0x060 .....	368
Register 16:	EPI Read FIFO Count (EPIRFIFOCNT), offset 0x06C .....	370
Register 17:	EPI Read FIFO (EPIREADFIFO), offset 0x070 .....	371
Register 18:	EPI Read FIFO Alias 1 (EPIREADFIFO1), offset 0x074 .....	371
Register 19:	EPI Read FIFO Alias 2 (EPIREADFIFO2), offset 0x078 .....	371
Register 20:	EPI Read FIFO Alias 3 (EPIREADFIFO3), offset 0x07C .....	371
Register 21:	EPI Read FIFO Alias 4 (EPIREADFIFO4), offset 0x080 .....	371
Register 22:	EPI Read FIFO Alias 5 (EPIREADFIFO5), offset 0x084 .....	371
Register 23:	EPI Read FIFO Alias 6 (EPIREADFIFO6), offset 0x088 .....	371
Register 24:	EPI Read FIFO Alias 7 (EPIREADFIFO7), offset 0x08C .....	371
Register 25:	EPI FIFO Level Selects (EPIFIFOLVL), offset 0x200 .....	372
Register 26:	EPI Write FIFO Count (EPIWFIFOCNT), offset 0x204 .....	374
Register 27:	EPI Interrupt Mask (EPIIM), offset 0x210 .....	375
Register 28:	EPI Raw Interrupt Status (EPIRIS), offset 0x214 .....	376
Register 29:	EPI Masked Interrupt Status (EPIMIS), offset 0x218 .....	378
Register 30:	EPI Error Interrupt Status and Clear (EPIEISC), offset 0x21C .....	379
<b>General-Purpose Timers .....</b>	<b>381</b>	
Register 1:	GPTM Configuration (GPTMCFG), offset 0x000 .....	394
Register 2:	GPTM Timer A Mode (GPTMTAMR), offset 0x004 .....	395
Register 3:	GPTM Timer B Mode (GPTMTBMR), offset 0x008 .....	397
Register 4:	GPTM Control (GPTMCTL), offset 0x00C .....	399
Register 5:	GPTM Interrupt Mask (GPTMIMR), offset 0x018 .....	402
Register 6:	GPTM Raw Interrupt Status (GPTMRIS), offset 0x01C .....	404
Register 7:	GPTM Masked Interrupt Status (GPTMMIS), offset 0x020 .....	406
Register 8:	GPTM Interrupt Clear (GPTMICR), offset 0x024 .....	408
Register 9:	GPTM Timer A Interval Load (GPTMTAILR), offset 0x028 .....	410
Register 10:	GPTM Timer B Interval Load (GPTMTBILR), offset 0x02C .....	411
Register 11:	GPTM Timer A Match (GPTMTAMATCHR), offset 0x030 .....	412
Register 12:	GPTM Timer B Match (GPTMTBMATCHR), offset 0x034 .....	413
Register 13:	GPTM Timer A Prescale (GPTMTAPR), offset 0x038 .....	414
Register 14:	GPTM Timer B Prescale (GPTMTBPR), offset 0x03C .....	415
Register 15:	GPTM Timer A (GPTMTAR), offset 0x048 .....	416
Register 16:	GPTM Timer B (GPTMTBR), offset 0x04C .....	417
Register 17:	GPTM Timer A Value (GPTMTAV), offset 0x050 .....	418
Register 18:	GPTM Timer B Value (GPTMTBV), offset 0x054 .....	419
<b>Watchdog Timer .....</b>	<b>420</b>	
Register 1:	Watchdog Load (WDTLOAD), offset 0x000 .....	424
Register 2:	Watchdog Value (WDTVALUE), offset 0x004 .....	425
Register 3:	Watchdog Control (WDTCTL), offset 0x008 .....	426
Register 4:	Watchdog Interrupt Clear (WDTICR), offset 0x00C .....	428
Register 5:	Watchdog Raw Interrupt Status (WDTRIS), offset 0x010 .....	429
Register 6:	Watchdog Masked Interrupt Status (WDTMIS), offset 0x014 .....	430

Register 7:	Watchdog Test (WDTTEST), offset 0x418 .....	431
Register 8:	Watchdog Lock (WDTLOCK), offset 0xC00 .....	432
Register 9:	Watchdog Peripheral Identification 4 (WDTPeriphID4), offset 0xFD0 .....	433
Register 10:	Watchdog Peripheral Identification 5 (WDTPeriphID5), offset 0xFD4 .....	434
Register 11:	Watchdog Peripheral Identification 6 (WDTPeriphID6), offset 0xFD8 .....	435
Register 12:	Watchdog Peripheral Identification 7 (WDTPeriphID7), offset 0xFDC .....	436
Register 13:	Watchdog Peripheral Identification 0 (WDTPeriphID0), offset 0xFE0 .....	437
Register 14:	Watchdog Peripheral Identification 1 (WDTPeriphID1), offset 0xFE4 .....	438
Register 15:	Watchdog Peripheral Identification 2 (WDTPeriphID2), offset 0xFE8 .....	439
Register 16:	Watchdog Peripheral Identification 3 (WDTPeriphID3), offset 0xFEC .....	440
Register 17:	Watchdog PrimeCell Identification 0 (WDTPCellID0), offset 0xFF0 .....	441
Register 18:	Watchdog PrimeCell Identification 1 (WDTPCellID1), offset 0xFF4 .....	442
Register 19:	Watchdog PrimeCell Identification 2 (WDTPCellID2), offset 0xFF8 .....	443
Register 20:	Watchdog PrimeCell Identification 3 (WDTPCellID3), offset 0xFFC .....	444
<b>Analog-to-Digital Converter (ADC) .....</b>		<b>445</b>
Register 1:	ADC Active Sample Sequencer (ADCACTSS), offset 0x000 .....	462
Register 2:	ADC Raw Interrupt Status (ADCRIS), offset 0x004 .....	463
Register 3:	ADC Interrupt Mask (ADCIM), offset 0x008 .....	465
Register 4:	ADC Interrupt Status and Clear (ADCISC), offset 0x00C .....	467
Register 5:	ADC Overflow Status (ADCOSTAT), offset 0x010 .....	469
Register 6:	ADC Event Multiplexer Select (ADCEMUX), offset 0x014 .....	470
Register 7:	ADC Underflow Status (ADCUSTAT), offset 0x018 .....	474
Register 8:	ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020 .....	475
Register 9:	ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028 .....	477
Register 10:	ADC Sample Averaging Control (ADCSAC), offset 0x030 .....	479
Register 11:	ADC Digital Comparator Interrupt Status and Clear (ADCDCISC), offset 0x034 .....	480
Register 12:	ADC Control (ADCCTL), offset 0x038 .....	482
Register 13:	ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040 .....	483
Register 14:	ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044 .....	485
Register 15:	ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048 .....	488
Register 16:	ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068 .....	488
Register 17:	ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088 .....	488
Register 18:	ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8 .....	488
Register 19:	ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C .....	489
Register 20:	ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C .....	489
Register 21:	ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C .....	489
Register 22:	ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC .....	489
Register 23:	ADC Sample Sequence 0 Operation (ADCSSOP0), offset 0x050 .....	491
Register 24:	ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0), offset 0x054 .....	493
Register 25:	ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060 .....	495
Register 26:	ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080 .....	495
Register 27:	ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064 .....	496
Register 28:	ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084 .....	496
Register 29:	ADC Sample Sequence 1 Operation (ADCSSOP1), offset 0x070 .....	498
Register 30:	ADC Sample Sequence 2 Operation (ADCSSOP2), offset 0x090 .....	498
Register 31:	ADC Sample Sequence 1 Digital Comparator Select (ADCSSDC1), offset 0x074 .....	499
Register 32:	ADC Sample Sequence 2 Digital Comparator Select (ADCSSDC2), offset 0x094 .....	499
Register 33:	ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0 .....	501



Register 34:	ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4 .....	502
Register 35:	ADC Sample Sequence 3 Operation (ADCSSOP3), offset 0x0B0 .....	503
Register 36:	ADC Sample Sequence 3 Digital Comparator Select (ADCSSDC3), offset 0x0B4 .....	504
Register 37:	ADC Digital Comparator Reset Initial Conditions (ADCDCRIC), offset 0xD00 .....	505
Register 38:	ADC Digital Comparator Control 0 (ADCDCCTL0), offset 0xE00 .....	509
Register 39:	ADC Digital Comparator Control 1 (ADCDCCTL1), offset 0xE04 .....	509
Register 40:	ADC Digital Comparator Control 2 (ADCDCCTL2), offset 0xE08 .....	509
Register 41:	ADC Digital Comparator Control 3 (ADCDCCTL3), offset 0xE0C .....	509
Register 42:	ADC Digital Comparator Control 4 (ADCDCCTL4), offset 0xE10 .....	509
Register 43:	ADC Digital Comparator Control 5 (ADCDCCTL5), offset 0xE14 .....	509
Register 44:	ADC Digital Comparator Control 6 (ADCDCCTL6), offset 0xE18 .....	509
Register 45:	ADC Digital Comparator Control 7 (ADCDCCTL7), offset 0xE1C .....	509
Register 46:	ADC Digital Comparator Range 0 (ADCDCCMP0), offset 0xE40 .....	512
Register 47:	ADC Digital Comparator Range 1 (ADCDCCMP1), offset 0xE44 .....	512
Register 48:	ADC Digital Comparator Range 2 (ADCDCCMP2), offset 0xE48 .....	512
Register 49:	ADC Digital Comparator Range 3 (ADCDCCMP3), offset 0xE4C .....	512
Register 50:	ADC Digital Comparator Range 4 (ADCDCCMP4), offset 0xE50 .....	512
Register 51:	ADC Digital Comparator Range 5 (ADCDCCMP5), offset 0xE54 .....	512
Register 52:	ADC Digital Comparator Range 6 (ADCDCCMP6), offset 0xE58 .....	512
Register 53:	ADC Digital Comparator Range 7 (ADCDCCMP7), offset 0xE5C .....	512
<b>Universal Asynchronous Receivers/Transmitters (UARTs) .....</b>		<b>513</b>
Register 1:	UART Data (UARTDR), offset 0x000 .....	523
Register 2:	UART Receive Status/Error Clear (UARTRSR/UARTECR), offset 0x004 .....	525
Register 3:	UART Flag (UARTFR), offset 0x018 .....	527
Register 4:	UART IrDA Low-Power Register (UARTILPR), offset 0x020 .....	529
Register 5:	UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024 .....	530
Register 6:	UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028 .....	531
Register 7:	UART Line Control (UARTLCRH), offset 0x02C .....	532
Register 8:	UART Control (UARTCTL), offset 0x030 .....	534
Register 9:	UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034 .....	537
Register 10:	UART Interrupt Mask (UARTIM), offset 0x038 .....	539
Register 11:	UART Raw Interrupt Status (UARTRIS), offset 0x03C .....	541
Register 12:	UART Masked Interrupt Status (UARTMIS), offset 0x040 .....	543
Register 13:	UART Interrupt Clear (UARTICR), offset 0x044 .....	545
Register 14:	UART DMA Control (UARTDMACTL), offset 0x048 .....	547
Register 15:	UART LIN Control (UARTLCTL), offset 0x090 .....	548
Register 16:	UART LIN Snap Shot (UARTLSS), offset 0x094 .....	549
Register 17:	UART LIN Timer (UARTLTIM), offset 0x098 .....	550
Register 18:	UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0 .....	551
Register 19:	UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4 .....	552
Register 20:	UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8 .....	553
Register 21:	UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC .....	554
Register 22:	UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0 .....	555
Register 23:	UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4 .....	556
Register 24:	UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8 .....	557
Register 25:	UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC .....	558
Register 26:	UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0 .....	559
Register 27:	UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4 .....	560

Register 28: UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8 .....	561
Register 29: UART PrimeCell Identification 3 (UARTPCellID3), offset 0xFFC .....	562
<b>Synchronous Serial Interface (SSI) .....</b>	<b>563</b>
Register 1: SSI Control 0 (SSICR0), offset 0x000 .....	577
Register 2: SSI Control 1 (SSICR1), offset 0x004 .....	579
Register 3: SSI Data (SSIDR), offset 0x008 .....	581
Register 4: SSI Status (SSISR), offset 0x00C .....	582
Register 5: SSI Clock Prescale (SSICPSR), offset 0x010 .....	584
Register 6: SSI Interrupt Mask (SSIIM), offset 0x014 .....	585
Register 7: SSI Raw Interrupt Status (SSIRIS), offset 0x018 .....	587
Register 8: SSI Masked Interrupt Status (SSIMIS), offset 0x01C .....	588
Register 9: SSI Interrupt Clear (SSIICR), offset 0x020 .....	589
Register 10: SSI DMA Control (SSIDMACTL), offset 0x024 .....	590
Register 11: SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0 .....	591
Register 12: SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4 .....	592
Register 13: SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8 .....	593
Register 14: SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC .....	594
Register 15: SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0 .....	595
Register 16: SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4 .....	596
Register 17: SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8 .....	597
Register 18: SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC .....	598
Register 19: SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0 .....	599
Register 20: SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4 .....	600
Register 21: SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8 .....	601
Register 22: SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC .....	602
<b>Inter-Integrated Circuit (I<sup>2</sup>C) Interface .....</b>	<b>603</b>
Register 1: I <sup>2</sup> C Master Slave Address (I2CMSA), offset 0x000 .....	618
Register 2: I <sup>2</sup> C Master Control/Status (I2CMCS), offset 0x004 .....	619
Register 3: I <sup>2</sup> C Master Data (I2CMDR), offset 0x008 .....	623
Register 4: I <sup>2</sup> C Master Timer Period (I2CMTPR), offset 0x00C .....	624
Register 5: I <sup>2</sup> C Master Interrupt Mask (I2CMIMR), offset 0x010 .....	625
Register 6: I <sup>2</sup> C Master Raw Interrupt Status (I2CMRIS), offset 0x014 .....	626
Register 7: I <sup>2</sup> C Master Masked Interrupt Status (I2CMMIS), offset 0x018 .....	627
Register 8: I <sup>2</sup> C Master Interrupt Clear (I2CMICR), offset 0x01C .....	628
Register 9: I <sup>2</sup> C Master Configuration (I2CMCR), offset 0x020 .....	629
Register 10: I <sup>2</sup> C Slave Own Address (I2CSOAR), offset 0x000 .....	630
Register 11: I <sup>2</sup> C Slave Control/Status (I2CSCSR), offset 0x004 .....	631
Register 12: I <sup>2</sup> C Slave Data (I2CSDR), offset 0x008 .....	633
Register 13: I <sup>2</sup> C Slave Interrupt Mask (I2CSIMR), offset 0x00C .....	634
Register 14: I <sup>2</sup> C Slave Raw Interrupt Status (I2CSRIS), offset 0x010 .....	635
Register 15: I <sup>2</sup> C Slave Masked Interrupt Status (I2CSMIS), offset 0x014 .....	636
Register 16: I <sup>2</sup> C Slave Interrupt Clear (I2CSICR), offset 0x018 .....	637
<b>Inter-Integrated Circuit Sound (I<sup>2</sup>S) Interface .....</b>	<b>638</b>
Register 1: I <sup>2</sup> S Transmit FIFO Data (I2STXFIFO), offset 0x000 .....	650
Register 2: I <sup>2</sup> S Transmit FIFO Configuration (I2STXFIFOCFG), offset 0x004 .....	651
Register 3: I <sup>2</sup> S Transmit Module Configuration (I2STXCFG), offset 0x008 .....	652
Register 4: I <sup>2</sup> S Transmit FIFO Limit (I2STXLIMIT), offset 0x00C .....	654

Register 5:	I <sup>2</sup> S Transmit Interrupt Status and Mask (I2STXISM), offset 0x010 .....	655
Register 6:	I <sup>2</sup> S Transmit FIFO Level (I2STXLEV), offset 0x018 .....	656
Register 7:	I <sup>2</sup> S Receive FIFO Data (I2SRXFIFO), offset 0x800 .....	657
Register 8:	I <sup>2</sup> S Receive FIFO Configuration (I2SRXFIFOCFG), offset 0x804 .....	658
Register 9:	I <sup>2</sup> S Receive Module Configuration (I2SRXCFG), offset 0x808 .....	659
Register 10:	I <sup>2</sup> S Receive FIFO Limit (I2SRXLIMIT), offset 0x80C .....	661
Register 11:	I <sup>2</sup> S Receive Interrupt Status and Mask (I2SRXISM), offset 0x810 .....	662
Register 12:	I <sup>2</sup> S Receive FIFO Level (I2SRXLEV), offset 0x818 .....	663
Register 13:	I <sup>2</sup> S Module Configuration (I2SCFG), offset 0xC00 .....	664
Register 14:	I <sup>2</sup> S Interrupt Mask (I2SIM), offset 0xC10 .....	665
Register 15:	I <sup>2</sup> S Raw Interrupt Status (I2SRIS), offset 0xC14 .....	666
Register 16:	I <sup>2</sup> S Masked Interrupt Status (I2SMIS), offset 0xC18 .....	668
Register 17:	I <sup>2</sup> S Interrupt Clear (I2SIC), offset 0xC1C .....	669
<b>Controller Area Network (CAN) Module .....</b>		<b>670</b>
Register 1:	CAN Control (CANCTL), offset 0x000 .....	689
Register 2:	CAN Status (CANSTS), offset 0x004 .....	691
Register 3:	CAN Error Counter (CANERR), offset 0x008 .....	694
Register 4:	CAN Bit Timing (CANBIT), offset 0x00C .....	695
Register 5:	CAN Interrupt (CANINT), offset 0x010 .....	697
Register 6:	CAN Test (CANTST), offset 0x014 .....	698
Register 7:	CAN Baud Rate Prescaler Extension (CANBRPE), offset 0x018 .....	700
Register 8:	CAN IF1 Command Request (CANIF1CRQ), offset 0x020 .....	701
Register 9:	CAN IF2 Command Request (CANIF2CRQ), offset 0x080 .....	701
Register 10:	CAN IF1 Command Mask (CANIF1CMSK), offset 0x024 .....	702
Register 11:	CAN IF2 Command Mask (CANIF2CMSK), offset 0x084 .....	702
Register 12:	CAN IF1 Mask 1 (CANIF1MSK1), offset 0x028 .....	704
Register 13:	CAN IF2 Mask 1 (CANIF2MSK1), offset 0x088 .....	704
Register 14:	CAN IF1 Mask 2 (CANIF1MSK2), offset 0x02C .....	705
Register 15:	CAN IF2 Mask 2 (CANIF2MSK2), offset 0x08C .....	705
Register 16:	CAN IF1 Arbitration 1 (CANIF1ARB1), offset 0x030 .....	706
Register 17:	CAN IF2 Arbitration 1 (CANIF2ARB1), offset 0x090 .....	706
Register 18:	CAN IF1 Arbitration 2 (CANIF1ARB2), offset 0x034 .....	707
Register 19:	CAN IF2 Arbitration 2 (CANIF2ARB2), offset 0x094 .....	707
Register 20:	CAN IF1 Message Control (CANIF1MCTL), offset 0x038 .....	709
Register 21:	CAN IF2 Message Control (CANIF2MCTL), offset 0x098 .....	709
Register 22:	CAN IF1 Data A1 (CANIF1DA1), offset 0x03C .....	711
Register 23:	CAN IF1 Data A2 (CANIF1DA2), offset 0x040 .....	711
Register 24:	CAN IF1 Data B1 (CANIF1DB1), offset 0x044 .....	711
Register 25:	CAN IF1 Data B2 (CANIF1DB2), offset 0x048 .....	711
Register 26:	CAN IF2 Data A1 (CANIF2DA1), offset 0x09C .....	711
Register 27:	CAN IF2 Data A2 (CANIF2DA2), offset 0x0A0 .....	711
Register 28:	CAN IF2 Data B1 (CANIF2DB1), offset 0x0A4 .....	711
Register 29:	CAN IF2 Data B2 (CANIF2DB2), offset 0x0A8 .....	711
Register 30:	CAN Transmission Request 1 (CANTXRQ1), offset 0x100 .....	712
Register 31:	CAN Transmission Request 2 (CANTXRQ2), offset 0x104 .....	712
Register 32:	CAN New Data 1 (CANNWDA1), offset 0x120 .....	713
Register 33:	CAN New Data 2 (CANNWDA2), offset 0x124 .....	713

Register 34:	CAN Message 1 Interrupt Pending (CANMSG1INT), offset 0x140 .....	714
Register 35:	CAN Message 2 Interrupt Pending (CANMSG2INT), offset 0x144 .....	714
Register 36:	CAN Message 1 Valid (CANMSG1VAL), offset 0x160 .....	715
Register 37:	CAN Message 2 Valid (CANMSG2VAL), offset 0x164 .....	715
<b>Universal Serial Bus (USB) Controller .....</b>		<b>716</b>
Register 1:	USB Device Functional Address (USBFADDR), offset 0x000 .....	735
Register 2:	USB Power (USBPOWER), offset 0x001 .....	736
Register 3:	USB Transmit Interrupt Status (USBTXIS), offset 0x002 .....	738
Register 4:	USB Receive Interrupt Status (USBRXIS), offset 0x004 .....	739
Register 5:	USB Transmit Interrupt Enable (USBTXIE), offset 0x006 .....	740
Register 6:	USB Receive Interrupt Enable (USBRXIE), offset 0x008 .....	741
Register 7:	USB General Interrupt Status (USBIS), offset 0x00A .....	742
Register 8:	USB Interrupt Enable (USBIE), offset 0x00B .....	744
Register 9:	USB Frame Value (USBFRAME), offset 0x00C .....	746
Register 10:	USB Endpoint Index (USBEPIDX), offset 0x00E .....	747
Register 11:	USB Test Mode (USBTEST), offset 0x00F .....	748
Register 12:	USB FIFO Endpoint 0 (USBFIFO0), offset 0x020 .....	750
Register 13:	USB FIFO Endpoint 1 (USBFIFO1), offset 0x024 .....	750
Register 14:	USB FIFO Endpoint 2 (USBFIFO2), offset 0x028 .....	750
Register 15:	USB FIFO Endpoint 3 (USBFIFO3), offset 0x02C .....	750
Register 16:	USB Device Control (USBDEVCTL), offset 0x060 .....	751
Register 17:	USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ), offset 0x062 .....	754
Register 18:	USB Receive Dynamic FIFO Sizing (USBRXFIFOSZ), offset 0x063 .....	754
Register 19:	USB Transmit FIFO Start Address (USBTXFIFOADD), offset 0x064 .....	755
Register 20:	USB Receive FIFO Start Address (USBRXFIFOADD), offset 0x066 .....	755
Register 21:	USB Connect Timing (USBCONTIM), offset 0x07A .....	756
Register 22:	USB OTG VBus Pulse Timing (USBVPLEN), offset 0x07B .....	757
Register 23:	USB Full-Speed Last Transaction to End of Frame Timing (USBFSEOF), offset 0x07D .....	758
Register 24:	USB Low-Speed Last Transaction to End of Frame Timing (USBLSEOF), offset 0x07E .....	759
Register 25:	USB Transmit Functional Address Endpoint 0 (USBTXFUNCADDR0), offset 0x080 .....	760
Register 26:	USB Transmit Functional Address Endpoint 1 (USBTXFUNCADDR1), offset 0x088 .....	760
Register 27:	USB Transmit Functional Address Endpoint 2 (USBTXFUNCADDR2), offset 0x090 .....	760
Register 28:	USB Transmit Functional Address Endpoint 3 (USBTXFUNCADDR3), offset 0x098 .....	760
Register 29:	USB Transmit Hub Address Endpoint 0 (USBTXHUBADDR0), offset 0x082 .....	761
Register 30:	USB Transmit Hub Address Endpoint 1 (USBTXHUBADDR1), offset 0x08A .....	761
Register 31:	USB Transmit Hub Address Endpoint 2 (USBTXHUBADDR2), offset 0x092 .....	761
Register 32:	USB Transmit Hub Address Endpoint 3 (USBTXHUBADDR3), offset 0x09A .....	761
Register 33:	USB Transmit Hub Port Endpoint 0 (USBTXHUBPORT0), offset 0x083 .....	762
Register 34:	USB Transmit Hub Port Endpoint 1 (USBTXHUBPORT1), offset 0x08B .....	762
Register 35:	USB Transmit Hub Port Endpoint 2 (USBTXHUBPORT2), offset 0x093 .....	762
Register 36:	USB Transmit Hub Port Endpoint 3 (USBTXHUBPORT3), offset 0x09B .....	762
Register 37:	USB Receive Functional Address Endpoint 1 (USBRXFUNCADDR1), offset 0x08C .....	763
Register 38:	USB Receive Functional Address Endpoint 2 (USBRXFUNCADDR2), offset 0x094 .....	763
Register 39:	USB Receive Functional Address Endpoint 3 (USBRXFUNCADDR3), offset 0x09C .....	763
Register 40:	USB Receive Hub Address Endpoint 1 (USBRXHUBADDR1), offset 0x08E .....	764
Register 41:	USB Receive Hub Address Endpoint 2 (USBRXHUBADDR2), offset 0x096 .....	764
Register 42:	USB Receive Hub Address Endpoint 3 (USBRXHUBADDR3), offset 0x09E .....	764
Register 43:	USB Receive Hub Port Endpoint 1 (USBRXHUBPORT1), offset 0x08F .....	765

Register 44:	USB Receive Hub Port Endpoint 2 (USBRXHUBPORT2), offset 0x097 .....	765
Register 45:	USB Receive Hub Port Endpoint 3 (USBRXHUBPORT3), offset 0x09F .....	765
Register 46:	USB Maximum Transmit Data Endpoint 1 (USBTXMAXP1), offset 0x110 .....	766
Register 47:	USB Maximum Transmit Data Endpoint 2 (USBTXMAXP2), offset 0x120 .....	766
Register 48:	USB Maximum Transmit Data Endpoint 3 (USBTXMAXP3), offset 0x130 .....	766
Register 49:	USB Control and Status Endpoint 0 Low (USBCSRL0), offset 0x102 .....	767
Register 50:	USB Control and Status Endpoint 0 High (USBCSRH0), offset 0x103 .....	770
Register 51:	USB Receive Byte Count Endpoint 0 (USBCOUNT0), offset 0x108 .....	772
Register 52:	USB Type Endpoint 0 (USBTYP0), offset 0x10A .....	773
Register 53:	USB NAK Limit (USBNAKLMT), offset 0x10B .....	774
Register 54:	USB Transmit Control and Status Endpoint 1 Low (USBTXCSRL1), offset 0x112 .....	775
Register 55:	USB Transmit Control and Status Endpoint 2 Low (USBTXCSRL2), offset 0x122 .....	775
Register 56:	USB Transmit Control and Status Endpoint 3 Low (USBTXCSRL3), offset 0x132 .....	775
Register 57:	USB Transmit Control and Status Endpoint 1 High (USBTXCSRH1), offset 0x113 .....	778
Register 58:	USB Transmit Control and Status Endpoint 2 High (USBTXCSRH2), offset 0x123 .....	778
Register 59:	USB Transmit Control and Status Endpoint 3 High (USBTXCSRH3), offset 0x133 .....	778
Register 60:	USB Maximum Receive Data Endpoint 1 (USBRXMAXP1), offset 0x114 .....	781
Register 61:	USB Maximum Receive Data Endpoint 2 (USBRXMAXP2), offset 0x124 .....	781
Register 62:	USB Maximum Receive Data Endpoint 3 (USBRXMAXP3), offset 0x134 .....	781
Register 63:	USB Receive Control and Status Endpoint 1 Low (USBRXCSRL1), offset 0x116 .....	782
Register 64:	USB Receive Control and Status Endpoint 2 Low (USBRXCSRL2), offset 0x126 .....	782
Register 65:	USB Receive Control and Status Endpoint 3 Low (USBRXCSRL3), offset 0x136 .....	782
Register 66:	USB Receive Control and Status Endpoint 1 High (USBRXCSRH1), offset 0x117 .....	785
Register 67:	USB Receive Control and Status Endpoint 2 High (USBRXCSRH2), offset 0x127 .....	785
Register 68:	USB Receive Control and Status Endpoint 3 High (USBRXCSRH3), offset 0x137 .....	785
Register 69:	USB Receive Byte Count Endpoint 1 (USBRXCOUNT1), offset 0x118 .....	788
Register 70:	USB Receive Byte Count Endpoint 2 (USBRXCOUNT2), offset 0x128 .....	788
Register 71:	USB Receive Byte Count Endpoint 3 (USBRXCOUNT3), offset 0x138 .....	788
Register 72:	USB Host Transmit Configure Type Endpoint 1 (USBTXTYPE1), offset 0x11A .....	789
Register 73:	USB Host Transmit Configure Type Endpoint 2 (USBTXTYPE2), offset 0x12A .....	789
Register 74:	USB Host Transmit Configure Type Endpoint 3 (USBTXTYPE3), offset 0x13A .....	789
Register 75:	USB Host Transmit Interval Endpoint 1 (USBTXINTERVAL1), offset 0x11B .....	791
Register 76:	USB Host Transmit Interval Endpoint 2 (USBTXINTERVAL2), offset 0x12B .....	791
Register 77:	USB Host Transmit Interval Endpoint 3 (USBTXINTERVAL3), offset 0x13B .....	791
Register 78:	USB Host Configure Receive Type Endpoint 1 (USBRXTYPE1), offset 0x11C .....	792
Register 79:	USB Host Configure Receive Type Endpoint 2 (USBRXTYPE2), offset 0x12C .....	792
Register 80:	USB Host Configure Receive Type Endpoint 3 (USBRXTYPE3), offset 0x13C .....	792
Register 81:	USB Host Receive Polling Interval Endpoint 1 (USBRXINTERVAL1), offset 0x11D .....	794
Register 82:	USB Host Receive Polling Interval Endpoint 2 (USBRXINTERVAL2), offset 0x12D .....	794
Register 83:	USB Host Receive Polling Interval Endpoint 3 (USBRXINTERVAL3), offset 0x13D .....	794
Register 84:	USB Request Packet Count in Block Transfer Endpoint 1 (USBRQPKTCOUNT1), offset 0x304 .....	795
Register 85:	USB Request Packet Count in Block Transfer Endpoint 2 (USBRQPKTCOUNT2), offset 0x308 .....	795
Register 86:	USB Request Packet Count in Block Transfer Endpoint 3 (USBRQPKTCOUNT3), offset 0x30C .....	795
Register 87:	USB Receive Double Packet Buffer Disable (USBRXDPKTBUFDIS), offset 0x340 .....	796
Register 88:	USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFDIS), offset 0x342 .....	797
Register 89:	USB External Power Control (USBEPCC), offset 0x400 .....	798

Register 90:	USB External Power Control Raw Interrupt Status (USBEPCRIS), offset 0x404	801
Register 91:	USB External Power Control Interrupt Mask (USBPCIM), offset 0x408	802
Register 92:	USB External Power Control Interrupt Status and Clear (USBPCISC), offset 0x40C	803
Register 93:	USB Device Resume Raw Interrupt Status (USBDRRIS), offset 0x410	804
Register 94:	USB Device Resume Interrupt Mask (USBDRIM), offset 0x414	805
Register 95:	USB Device Resume Interrupt Status and Clear (USBDRISC), offset 0x418	806
Register 96:	USB VBUS Droop Control (USBVDC), offset 0x430	807
Register 97:	USB VBUS Droop Control Raw Interrupt Status (USBVDCRIS), offset 0x434	808
Register 98:	USB VBUS Droop Control Interrupt Mask (USBVDCIM), offset 0x438	809
Register 99:	USB VBUS Droop Control Interrupt Status and Clear (USBVDCISC), offset 0x43C	810
Register 100:	USB ID Valid Detect Raw Interrupt Status (USBIDVRIS), offset 0x444	811
Register 101:	USB ID Valid Detect Interrupt Mask (USBIDVIM), offset 0x448	812
Register 102:	USB ID Valid Detect Interrupt Status and Clear (USBIDVISC), offset 0x44C	813
Register 103:	USB End-Point Select (USBEPS), offset 0x450	814
<b>Analog Comparators</b>		<b>816</b>
Register 1:	Analog Comparator Masked Interrupt Status (ACMIS), offset 0x000	821
Register 2:	Analog Comparator Raw Interrupt Status (ACRIS), offset 0x004	822
Register 3:	Analog Comparator Interrupt Enable (ACINTEN), offset 0x008	823
Register 4:	Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x010	824
Register 5:	Analog Comparator Status 0 (ACSTAT0), offset 0x020	825
Register 6:	Analog Comparator Status 1 (ACSTAT1), offset 0x040	825
Register 7:	Analog Comparator Status 2 (ACSTAT2), offset 0x060	825
Register 8:	Analog Comparator Control 0 (ACCTL0), offset 0x024	826
Register 9:	Analog Comparator Control 1 (ACCTL1), offset 0x044	826
Register 10:	Analog Comparator Control 2 (ACCTL2), offset 0x064	826
<b>Pulse Width Modulator (PWM)</b>		<b>828</b>
Register 1:	PWM Master Control (PWMCTL), offset 0x000	839
Register 2:	PWM Time Base Sync (PWMSYNC), offset 0x004	840
Register 3:	PWM Output Enable (PWMENABLE), offset 0x008	841
Register 4:	PWM Output Inversion (PWMINVERT), offset 0x00C	843
Register 5:	PWM Output Fault (PWMFAULT), offset 0x010	844
Register 6:	PWM Interrupt Enable (PWMINTEN), offset 0x014	846
Register 7:	PWM Raw Interrupt Status (PWMRIS), offset 0x018	848
Register 8:	PWM Interrupt Status and Clear (PWMISC), offset 0x01C	850
Register 9:	PWM Status (PWMSTATUS), offset 0x020	852
Register 10:	PWM Fault Condition Value (PWMFAULTVAL), offset 0x024	853
Register 11:	PWM0 Control (PWM0CTL), offset 0x040	855
Register 12:	PWM1 Control (PWM1CTL), offset 0x080	855
Register 13:	PWM2 Control (PWM2CTL), offset 0x0C0	855
Register 14:	PWM3 Control (PWM3CTL), offset 0x100	855
Register 15:	PWM0 Interrupt and Trigger Enable (PWM0INTEN), offset 0x044	860
Register 16:	PWM1 Interrupt and Trigger Enable (PWM1INTEN), offset 0x084	860
Register 17:	PWM2 Interrupt and Trigger Enable (PWM2INTEN), offset 0x0C4	860
Register 18:	PWM3 Interrupt and Trigger Enable (PWM3INTEN), offset 0x104	860
Register 19:	PWM0 Raw Interrupt Status (PWM0RIS), offset 0x048	862
Register 20:	PWM1 Raw Interrupt Status (PWM1RIS), offset 0x088	862
Register 21:	PWM2 Raw Interrupt Status (PWM2RIS), offset 0x0C8	862
Register 22:	PWM3 Raw Interrupt Status (PWM3RIS), offset 0x108	862

Register 23:	PWM0 Interrupt Status and Clear (PWM0ISC), offset 0x04C	863
Register 24:	PWM1 Interrupt Status and Clear (PWM1ISC), offset 0x08C	863
Register 25:	PWM2 Interrupt Status and Clear (PWM2ISC), offset 0x0CC	863
Register 26:	PWM3 Interrupt Status and Clear (PWM3ISC), offset 0x10C	863
Register 27:	PWM0 Load (PWM0LOAD), offset 0x050	864
Register 28:	PWM1 Load (PWM1LOAD), offset 0x090	864
Register 29:	PWM2 Load (PWM2LOAD), offset 0x0D0	864
Register 30:	PWM3 Load (PWM3LOAD), offset 0x110	864
Register 31:	PWM0 Counter (PWM0COUNT), offset 0x054	865
Register 32:	PWM1 Counter (PWM1COUNT), offset 0x094	865
Register 33:	PWM2 Counter (PWM2COUNT), offset 0x0D4	865
Register 34:	PWM3 Counter (PWM3COUNT), offset 0x114	865
Register 35:	PWM0 Compare A (PWM0CMPA), offset 0x058	866
Register 36:	PWM1 Compare A (PWM1CMPA), offset 0x098	866
Register 37:	PWM2 Compare A (PWM2CMPA), offset 0x0D8	866
Register 38:	PWM3 Compare A (PWM3CMPA), offset 0x118	866
Register 39:	PWM0 Compare B (PWM0CMPB), offset 0x05C	867
Register 40:	PWM1 Compare B (PWM1CMPB), offset 0x09C	867
Register 41:	PWM2 Compare B (PWM2CMPB), offset 0x0DC	867
Register 42:	PWM3 Compare B (PWM3CMPB), offset 0x11C	867
Register 43:	PWM0 Generator A Control (PWM0GENA), offset 0x060	868
Register 44:	PWM1 Generator A Control (PWM1GENA), offset 0x0A0	868
Register 45:	PWM2 Generator A Control (PWM2GENA), offset 0x0E0	868
Register 46:	PWM3 Generator A Control (PWM3GENA), offset 0x120	868
Register 47:	PWM0 Generator B Control (PWM0GENB), offset 0x064	871
Register 48:	PWM1 Generator B Control (PWM1GENB), offset 0x0A4	871
Register 49:	PWM2 Generator B Control (PWM2GENB), offset 0x0E4	871
Register 50:	PWM3 Generator B Control (PWM3GENB), offset 0x124	871
Register 51:	PWM0 Dead-Band Control (PWM0DBCTL), offset 0x068	874
Register 52:	PWM1 Dead-Band Control (PWM1DBCTL), offset 0x0A8	874
Register 53:	PWM2 Dead-Band Control (PWM2DBCTL), offset 0x0E8	874
Register 54:	PWM3 Dead-Band Control (PWM3DBCTL), offset 0x128	874
Register 55:	PWM0 Dead-Band Rising-Edge Delay (PWM0DBRISE), offset 0x06C	875
Register 56:	PWM1 Dead-Band Rising-Edge Delay (PWM1DBRISE), offset 0x0AC	875
Register 57:	PWM2 Dead-Band Rising-Edge Delay (PWM2DBRISE), offset 0x0EC	875
Register 58:	PWM3 Dead-Band Rising-Edge Delay (PWM3DBRISE), offset 0x12C	875
Register 59:	PWM0 Dead-Band Falling-Edge-Delay (PWM0DBFALL), offset 0x070	876
Register 60:	PWM1 Dead-Band Falling-Edge-Delay (PWM1DBFALL), offset 0x0B0	876
Register 61:	PWM2 Dead-Band Falling-Edge-Delay (PWM2DBFALL), offset 0x0F0	876
Register 62:	PWM3 Dead-Band Falling-Edge-Delay (PWM3DBFALL), offset 0x130	876
Register 63:	PWM0 Fault Source 0 (PWM0FLTSRC0), offset 0x074	877
Register 64:	PWM1 Fault Source 0 (PWM1FLTSRC0), offset 0x0B4	877
Register 65:	PWM2 Fault Source 0 (PWM2FLTSRC0), offset 0x0F4	877
Register 66:	PWM3 Fault Source 0 (PWM3FLTSRC0), offset 0x134	877
Register 67:	PWM0 Fault Source 1 (PWM0FLTSRC1), offset 0x078	879
Register 68:	PWM1 Fault Source 1 (PWM1FLTSRC1), offset 0x0B8	879
Register 69:	PWM2 Fault Source 1 (PWM2FLTSRC1), offset 0x0F8	879
Register 70:	PWM3 Fault Source 1 (PWM3FLTSRC1), offset 0x138	879

Register 71:	PWM0 Minimum Fault Period (PWM0MINFLTPER), offset 0x07C .....	881
Register 72:	PWM1 Minimum Fault Period (PWM1MINFLTPER), offset 0x0BC .....	881
Register 73:	PWM2 Minimum Fault Period (PWM2MINFLTPER), offset 0x0FC .....	881
Register 74:	PWM3 Minimum Fault Period (PWM3MINFLTPER), offset 0x13C .....	881
Register 75:	PWM0 Fault Pin Logic Sense (PWM0FLTSEN), offset 0x800 .....	882
Register 76:	PWM1 Fault Pin Logic Sense (PWM1FLTSEN), offset 0x880 .....	882
Register 77:	PWM2 Fault Pin Logic Sense (PWM2FLTSEN), offset 0x900 .....	882
Register 78:	PWM3 Fault Pin Logic Sense (PWM3FLTSEN), offset 0x980 .....	882
Register 79:	PWM0 Fault Status 0 (PWM0FLTSTAT0), offset 0x804 .....	883
Register 80:	PWM1 Fault Status 0 (PWM1FLTSTAT0), offset 0x884 .....	883
Register 81:	PWM2 Fault Status 0 (PWM2FLTSTAT0), offset 0x904 .....	883
Register 82:	PWM3 Fault Status 0 (PWM3FLTSTAT0), offset 0x984 .....	883
Register 83:	PWM0 Fault Status 1 (PWM0FLTSTAT1), offset 0x808 .....	885
Register 84:	PWM1 Fault Status 1 (PWM1FLTSTAT1), offset 0x888 .....	885
Register 85:	PWM2 Fault Status 1 (PWM2FLTSTAT1), offset 0x908 .....	885
Register 86:	PWM3 Fault Status 1 (PWM3FLTSTAT1), offset 0x988 .....	885
<b>Quadrature Encoder Interface (QEI) .....</b>	<b>887</b>	
Register 1:	QEI Control (QEICTL), offset 0x000 .....	892
Register 2:	QEI Status (QEISTAT), offset 0x004 .....	894
Register 3:	QEI Position (QEIPPOS), offset 0x008 .....	895
Register 4:	QEI Maximum Position (QEIMAXPOS), offset 0x00C .....	896
Register 5:	QEI Timer Load (QEILOAD), offset 0x010 .....	897
Register 6:	QEI Timer (QEITIME), offset 0x014 .....	898
Register 7:	QEI Velocity Counter (QEICOUNT), offset 0x018 .....	899
Register 8:	QEI Velocity (QEISPEED), offset 0x01C .....	900
Register 9:	QEI Interrupt Enable (QEIINTEN), offset 0x020 .....	901
Register 10:	QEI Raw Interrupt Status (QEIRIS), offset 0x024 .....	902
Register 11:	QEI Interrupt Status and Clear (QEIISC), offset 0x028 .....	903



## About This Document

This data sheet provides reference information for the LM3S5791 microcontroller, describing the functional blocks of the system-on-chip (SoC) device designed around the ARM® Cortex™-M3 core.

### Audience

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

### About This Manual

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

### Related Documents

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

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

The following related documents are also referenced:

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

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

### Documentation Conventions

This document uses the conventions shown in Table 1 on page 29.

**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 68.

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

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

# 1 Architectural Overview

Luminary Micro is the industry leader in bringing 32-bit capabilities and the full benefits of ARM® Cortex-M3™-based microcontrollers to the broadest reach of the microcontroller market. For current users of 8- and 16-bit MCUs, Stellaris® with Cortex-M3 offers a direct path to the strongest ecosystem of development tools, software and knowledge in the industry. Designers who migrate to Stellaris® benefit from great tools, small code footprint and outstanding performance. Even more important, designers can enter the ARM ecosystem with full confidence in a compatible roadmap from \$1 to 1 GHz. For users of current 32-bit MCUs, the Stellaris® family offers the industry's first implementation of Cortex-M3 and the Thumb-2 instruction set. With blazingly-fast responsiveness, Thumb-2 technology combines both 16-bit and 32-bit instructions to deliver the best balance of code density and performance. Thumb-2 uses 26 percent less memory than pure 32-bit code to reduce system cost while delivering 25 percent better performance. 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 LM3S5791 microcontroller has the following features:

- ARM® Cortex™-M3 Processor Core
  - 80-MHz operation; 100 DMIPS performance
  - ARM Cortex SysTick Timer
  - Nested Vectored Interrupt Controller (NVIC)
- On-Chip Memory
  - 128 KB single-cycle Flash
  - 64 KB single-cycle SRAM
  - Internal ROM loaded with StellarisWare software:
    - Stellaris® Peripheral Driver Library
    - Stellaris® Boot Loader
    - Advanced Encryption Standard (AES) cryptography tables
    - Cyclic Redundancy Check (CRC) error detection functionality
- External Peripheral Interface (EPI)
  - 8/16/32-bit dedicated parallel bus for external peripherals
  - Supports SDRAM, SRAM/Flash, FPGAs, CPLDs
- Advanced Serial Integration
  - Two CAN 2.0 A/B controllers
  - USB 2.0 OTG/Host/Device

- Three UARTs with IrDA and ISO 7816 support (one UART with full modem controls)
- Two I<sup>2</sup>C modules
- Two Synchronous Serial Interface modules (SSI)
- Integrated Interchip Sound (I<sup>2</sup>S) module
- System Integration
  - Direct Memory Access Controller (DMA)
  - System control and clocks including on-chip precision 16-MHz oscillator
  - Four 32-bit timers (up to eight 16-bit)
  - Eight Capture Compare PWM pins (CCP)
  - Real-Time Clock
  - Two Watchdog Timers
    - One timer runs off the main oscillator
    - One timer runs off the precision internal oscillator
  - 0-72 GPIOs, depending on configuration
    - Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
    - Independently configurable to 2, 4 or 8 mA drive capability
    - Up to 4 GPIOs can have 18 mA drive capability
- Advanced Motion Control
  - Eight advanced PWM outputs for motion and energy applications
  - Four fault inputs to promote low-latency shutdown
  - Two Quadrature Encoder Inputs (QEI)
- Analog
  - Two 10-bit Analog-to-Digital Converters (ADC) with sixteen analog input channels and sample rate of one million samples/second
  - Three Analog Comparators
  - 16 Digital Comparators
  - On-chip voltage regulator
- JTAG and ARM Serial Wire Debug (SWD)
- 100-pin LQFP package

- Industrial (-40°C to 85°C) Temperature Range

The LM3S5791 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.

In addition, the LM3S5791 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 LM3S5791 microcontroller is code-compatible to all members of the extensive Stellaris® family; providing flexibility to fit our customers' precise needs.

Luminary Micro offers a complete solution to get to market quickly, with evaluation and development boards, white papers and application notes, an easy-to-use peripheral driver library, and a strong support, sales, and distributor network. See "Ordering and Contact Information" on page 1028 for ordering information for Stellaris® family devices.

## 1.1 Functional Overview

The following sections provide an overview of the features of the LM3S5791 microcontroller. The page number in parentheses indicates where that feature is discussed in detail. Ordering and support information can be found in "Ordering and Contact Information" on page 1028.

### 1.1.1 ARM Cortex™-M3

The following sections provide an overview of the ARM Cortex™-M3 processor core and instruction set, the integrated System Timer (SysTick) and the Nested Vectored Interrupt Controller.

#### 1.1.1.1 Processor Core (see page 55)

All members of the Stellaris® product family, including the LM3S5791 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.

- 32-bit ARM® Cortex™-M3 v7M architecture optimized for small-footprint embedded applications
- Thumb-2 mixed 16-/32-bit instruction set, delivers the high performance expected of from a 32-bit ARM core in a compact memory size usually associated with 8- and 16-bit devices; typically in the range of a few kilobytes of memory for microcontroller-class applications
  - Single-cycle multiply instruction and hardware divide
  - Atomic bit manipulation (bit-banding), delivering maximum memory utilization and streamlined peripheral control
  - Unaligned data access, enabling data to be efficiently packed into memory
- Harvard architecture characterized by separate buses for instruction and data
- Memory protection unit (MPU) to provide a privileged mode for protected operating system functionality

- Migration from the ARM7™ processor family for better performance and power efficiency
- Optimized for single-cycle Flash usage
- 80-MHz operation
- 1.25 DMIPS/MHz

“ARM Cortex-M3 Processor Core” on page 55 provides an overview of the ARM core; the core is detailed in the *ARM® Cortex™-M3 Technical Reference Manual*.

#### 1.1.1.2 System Timer (SysTick) (see page 65)

ARM 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 that fires at a programmable rate (for example, 100 Hz) and invokes a SysTick routine
- A high-speed alarm timer using the system clock
- A variable rate alarm or signal timer—the duration is range-dependent on the reference clock used and the dynamic range of the counter
- A simple counter used to measure time to completion and time used
- An internal clock-source control based on missing/meeting durations. The COUNTFLAG field in the SysTick 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

#### 1.1.1.3 Nested Vectored Interrupt Controller (NVIC) (see page 71)

The LM3S5791 controller includes the ARM Nested Vectored Interrupt Controller (NVIC). The NVIC and Cortex-M3 prioritize and handle all exceptions in Handler Mode. The processor state is automatically stored to the stack on an exception and automatically restored from the stack at the end of the Interrupt Service Routine (ISR). The interrupt vector is fetched in parallel to the state saving, enabling efficient interrupt entry. The processor supports tail-chaining, meaning that back-to-back interrupts can be performed without the overhead of state saving and restoration. Software can set eight priority levels on 7 exceptions (system handlers) and 52 interrupts.

- Deterministic, fast interrupt processing: always 12 cycles, or just 6 cycles with tail-chaining
- External non-maskable interrupt signal (NMI) available for immediate execution of NMI handler for safety critical applications
- Dynamically reprioritizable interrupts
- Exceptional interrupt handling via hardware implementation of required register manipulations

“Interrupts” on page 71 provides an overview of the NVIC controller and the interrupt map. Exceptions and interrupts are detailed in the *ARM® Cortex™-M3 Technical Reference Manual*.

### 1.1.2 On-Chip Memory

The following sections describe the on-chip memory modules.

### 1.1.2.1 SRAM (see page 189)

The LM3S5791 microcontroller provides 64 KB of single-cycle on-chip SRAM. The internal SRAM of the Stellaris® devices is located at offset 0x2000.0000 of the device memory map.

Because read-modify-write (RMW) operations are very time consuming, 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.

Data can be transferred to and from the SRAM using the Micro Direct Memory Access Controller (μDMA).

### 1.1.2.2 Flash (see page 189)

The LM3S5791 microcontroller provides 128 KB of single-cycle on-chip Flash memory. The Flash is organized as a set of 2-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.

### 1.1.2.3 ROM (see page 972)

Preprogrammed in the LM3S5791 microcontroller's on-chip read-only memory (ROM) is the Stellaris® Peripheral Driver Library, a royalty-free software library for controlling on-chip peripherals with a boot-loader capability. The library performs both peripheral initialization and control functions, with a choice of polled or interrupt-driven peripheral support. In addition, the library is designed to take full advantage of the stellar interrupt performance of the ARM® Cortex™-M3 core. No special pragmas or custom assembly code prologue/epilogue functions are required. For applications that require in-field programmability, the royalty-free Stellaris® Boot Loader can act as an application loader and support in-field firmware updates.

The Advanced Encryption Standard (AES) is a publicly defined encryption standard used by the U.S. Government. AES is a strong encryption method with reasonable performance and size. In addition, it is fast in both hardware and software, is fairly easy to implement, and requires little memory. The Luminary Micro encryption package is available with full source code, and is based on lesser general public license (LGPL) source. An LGPL means that the code can be used within an application without any copyleft implications for the application (the code does not automatically become open source). Modifications to the package source, however, must be open source.

CRC (Cyclic Redundancy Check) is a technique to validate a span of data has the same contents as when previously checked. This technique can be used to validate correct receipt of messages (nothing lost or modified in transit), to validate data after decompression, to validate that Flash contents have not been changed, and for other cases where the data needs to be validated. A CRC is preferred over a simple checksum (e.g. XOR all bits) because it catches changes more readily.

The LM3S5791 ROM is preprogrammed with the following software and programs:

- Stellaris® Peripheral Driver Library
- Stellaris® Boot Loader
- Advanced Encryption Standard (AES) cryptography tables



- Cyclic Redundancy Check (CRC) error-detection functionality

### 1.1.3 External Peripheral Interface (see page 335)

The External Peripheral Interface (EPI) provides access to external devices using a parallel path. Unlike communications peripherals such as SSI, UART, and I<sup>2</sup>C, the EPI is designed to act like a bus to external peripherals and memory. The EPI has the following features:

- 16-bit dedicated parallel bus for external peripherals and memory
- Memory interface supports contiguous memory access independent of data bus width, thus enabling code execution directly from SDRAM, SRAM and Flash memory
- Blocking and non-blocking reads
- Processor from timing details through use of an internal write FIFO
- Efficient transfers using Micro Direct Memory Access Controller ( $\mu$ DMA)
  - Separate channels for read and write
  - Read channel request asserted by programmable levels on the internal non-blocking read FIFO (NBRFIFO)
  - Write channel request asserted by empty on the internal write FIFO (WFIFO)

The EPI supports three primary functional modes: Synchronous Dynamic Random Access Memory (SDRAM) mode, Traditional Host-Bus mode, and General-Purpose mode. The EPI module also provides custom GPIOs; however, unlike regular GPIOs, the EPI module uses a FIFO in the same way as a communication mechanism and is speed-controlled using clocking.

- Synchronous Dynamic Random Access Memory (SDRAM)
  - Supports x16 (single data rate) SDRAM at up to 50 MHz
  - Supports low-cost SDRAMs up to 64 MB (512 Mb)
  - Includes automatic refresh and access to all banks/rows
  - Includes a Sleep/Standby mode to keep contents active with minimal power draw
  - Multiplexed address/data interface for reduced pin count
- Host-bus
  - Traditional x8 MCU bus interface capabilities
  - Similar device compatibility options as PIC, ATmega, 8051, and others
  - Access to SRAM, NOR Flash, and other devices, with up to 1 MB of addressing
  - Support of both muxed and de-muxed address and data
  - Access to a range of devices supporting the non-address FIFO x8 interface variant, with support for external FIFO (XFIFO) `EMPTY` and `FULL` signals

- Speed controlled, with read and write data wait-state counters
- Manual chip-enable (or use extra address pins)
- General Purpose
  - Wide parallel interfaces for fast communications with CPLDs and FPGAs
  - Data widths up to 32-bits
  - Data rates up to 150 Mbytes/second
  - Optional “address” sizes from 4-bits to 16-bits
  - Optional clock output, read/write strobes, framing (with counter-based size), and clock-enable input
- General parallel GPIO
  - 1 to 32 bits, FIFOed with speed control
  - Useful for custom peripherals or for digital data acquisition and actuator controls

#### 1.1.4 Serial Communications Peripherals

The LM3S5791 controller supports both asynchronous and synchronous serial communications with:

- Two CAN 2.0 A/B Controllers
- USB 2.0 (full speed and low speed) OTG/Host/Device
- Three UARTs with IrDA and ISO 7816 support (one UART with full modem controls)
- Two I<sup>2</sup>C modules
- Two Synchronous Serial Interface Modules (SSI)
- Integrated Interchip Sound (I<sup>2</sup>S) Module

The following sections provide more detail on each of these communications functions.

##### 1.1.4.1 Controller Area Network (see page 670)

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 twisted-pair wire. Originally created for automotive purposes, it is now used in many embedded control applications (for example, industrial or medical). Bit rates up to 1Mbps are possible at network lengths below 40 meters. Decreased bit rates allow longer network distances (for example, 125 Kbps 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 LM3S5791 microcontroller includes two CAN units with the following features:

- CAN protocol version 2.0 part A/B
- Bit rates up to 1 Mbps
- 32 message objects with individual identifier masks
- Maskable interrupt
- Disable Automatic Retransmission mode for Time-Triggered CAN (TTCAN) applications
- Programmable Loopback mode for self-test operation
- Programmable FIFO mode enables storage of multiple message objects
- Gluelessly attaches to an external CAN transceiver through the CANnTX and CANnRX signals

#### 1.1.4.2 USB (see page 716)

Universal Serial Bus (USB) is a serial bus standard designed to allow peripherals to be connected and disconnected using a standardized interface without rebooting the system.

The LM3S5791 controller supports three configurations in USB 2.0 full and low speed: USB Device, USB Host, and USB On-The-Go (negotiated on-the-go as host or device when connected to other USB-enabled systems). The USB module has the following features:

- Complies with USB-IF certification standards
- USB 2.0 full-speed (12 Mbps) and low-speed (1.5 Mbps) operation
- Integrated PHY
- 4 transfer types: Control, Interrupt, Bulk, and Isochronous
- 16 endpoints
  - 1 dedicated control IN endpoint and 1 dedicated control OUT endpoint
  - 7 configurable IN endpoints and 7 configurable OUT endpoints
- 4 KB dedicated endpoint memory - one endpoint may be defined for double-buffered 1023-byte isochronous packet size
- Efficient transfers using Micro Direct Memory Access Controller ( $\mu$ DMA)
  - Separate channels for transmit and receive for up to 3 IN endpoints and 3 OUT endpoints
  - Burst requests
  - Channel requests asserted when FIFO contains required amount of data

#### 1.1.4.3 UART (see page 513)

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 LM3S5791 controller includes three fully programmable 16C550-type UARTs. Although the functionality is similar to a 16C550 UART, this UART design is not register compatible. The UART

can generate individually masked interrupts from the Rx, Tx, modem status, and error conditions. The module generates a single combined interrupt when any of the interrupts are asserted and are unmasked. The UARTs have the following features:

- Programmable baud-rate generator allowing speeds up to 5 Mbps for regular speed (divide by 16) and 10 Mbps for high speed (divide by 8)
- Separate 16x8 transmit (TX) and receive (RX) FIFOs to reduce CPU interrupt service loading
- LIN protocol support
- 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
- 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
- Support for communication with ISO 7816 smart cards
- Full modem handshake support (on UART1)
- Standard FIFO-level and End-of-Transmission interrupts
- Efficient transfers using Micro Direct Memory Access Controller ( $\mu$ DMA)
  - Separate channels for transmit and receive
  - Receive single request asserted when data is in the FIFO; burst request asserted at programmed FIFO level
  - Transmit single request asserted when there is space in the FIFO; burst request asserted at programmed FIFO level

#### 1.1.4.4 I<sup>2</sup>C (see page 603)

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.

Each device on the I<sup>2</sup>C bus can be designated as either a master or a slave. Each I<sup>2</sup>C module supports both sending and receiving data as either a master or a slave and can operate simultaneously as both a master and a slave. Both the I<sup>2</sup>C master and slave can generate interrupts.

The LM3S5791 controller includes two I<sup>2</sup>C modules with the following features:

- Devices on the I<sup>2</sup>C bus can be designated as either a master or a slave
  - Supports both sending and receiving data as either a master or a slave
  - Supports simultaneous master and slave operation
- Four I<sup>2</sup>C modes
  - Master transmit
  - Master receive
  - Slave transmit
  - Slave receive
- Two transmission speeds: Standard (100 Kbps) and Fast (400 Kbps)
- Master and slave interrupt generation
  - Master generates interrupts when a transmit or receive operation completes (or aborts due to an error)
  - Slave generates interrupts when data has been sent or requested by a master or when a START or STOP condition is detected
- Master with arbitration and clock synchronization, multimaster support, and 7-bit addressing mode

#### 1.1.4.5 SSI (see page 563)

Synchronous Serial Interface (SSI) is a four-wire bi-directional communications interface that converts data between parallel and serial. Each 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. Each 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 TX and RX paths are buffered with separate internal FIFOs.

Each 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.

Each SSI module provide the following features:

- Programmable interface operation for Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces
- Master or slave operation
- Programmable clock bit rate and prescaler
- Separate transmit and receive FIFOs, 16 bits wide, 8 locations deep
- Programmable data frame size from 4 to 16 bits
- Internal loopback test mode for diagnostic/debug testing
- Standard FIFO-based interrupts and End-of-Transmission interrupt
- Efficient transfers using Micro Direct Memory Access Controller ( $\mu$ DMA)
  - Separate channels for transmit and receive
  - Receive single request asserted when data is in the FIFO; burst request asserted when FIFO contains 4 entries
  - Transmit single request asserted when there is space in the FIFO; burst request asserted when FIFO contains 4 entries

#### 1.1.4.6 Inter-Integrated Circuit Sound (I<sup>2</sup>S) Interface (see page 638)

The I<sup>2</sup>S interface is a configurable serial audio core that contains a transmit module and a receive module. The module is configurable for the I<sup>2</sup>S as well as Left-Justified and Right-Justified serial audio formats. Data can be in one of four modes: Stereo, Mono, Compact 16-bit Stereo and Compact 8-Bit Stereo.

The transmit and receive modules each have an 8-entry audio-sample FIFO. An audio sample can consist of a Left and Right Stereo sample, a Mono sample, or a Left and Right Compact Stereo sample. In Compact 16-Bit Stereo, each FIFO entry contains both the 16-bit left and 16-bit right samples, allowing efficient data transfers and requiring less memory space. In Compact 8-bit Stereo, each FIFO entry contains an 8-bit left and an 8-bit right sample, reducing memory requirements further.

Both the transmitter and receiver are capable of being a master or a slave.

The Stellaris<sup>®</sup> I<sup>2</sup>S interface has the following features:

- Configurable audio format supporting I<sup>2</sup>S, Left-justification, and Right-justification
- Configurable sample size from 8 to 32 bits
- Mono and Stereo support
- 8-, 16-, and 32-bit FIFO interface for packing memory
- Independent transmit and receive 8-entry FIFOs
- Configurable FIFO-level interrupt and  $\mu$ DMA requests
- Independent transmit and receive MCLK direction control

- Transmit and receive internal MCLK sources
- Independent transmit and receive control for serial clock and word select
- MCLK and SCLK can be independently set to master or slave
- Configurable transmit zero or last sample when FIFO empty
- Efficient transfers using Micro Direct Memory Access Controller ( $\mu$ DMA)
  - Separate channels for transmit and receive
  - Burst requests
  - Channel requests asserted when FIFO contains required amount of data

### 1.1.5 System Integration

The LM3S5791 controller provides a variety of standard system functions integrated into the device, including:

- Micro Direct Memory Access Controller ( $\mu$ DMA)
- System control and clocks including on-chip precision 16-MHz oscillator
- ARM Cortex SysTick Timer
- Four 32-bit timers (up to eight 16-bit)
- Eight Capture Compare PWM pins (CCP)
- Real-Time Clock
- Watchdog Timer
- 0-72 GPIOs, depending on configuration
  - Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
  - Independently configurable to 2, 4 or 8 mA drive capability
  - Up to 4 GPIOs can have 18 mA drive capability

The following sections provide more detail on each of these functions.

#### 1.1.5.1 Direct Memory Access (see page 219)

The LM3S5791 microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA ( $\mu$ DMA). The  $\mu$ DMA controller provides a way to offload data transfer tasks from the Cortex-M3 processor, allowing for more efficient use of the processor and the expanded available bus bandwidth. The  $\mu$ DMA controller can perform transfers between memory and peripherals. It has dedicated channels for each supported on-chip module and can be programmed to automatically perform transfers between peripherals and memory as the peripheral is ready to transfer more data. The  $\mu$ DMA controller provides the following features:

- ARM PrimeCell® 32-channel configurable  $\mu$ DMA controller

- Support for multiple transfer modes
  - Memory-to-memory, memory-to-peripheral, peripheral-to-memory
  - Basic for simple transfer scenarios
  - Ping-pong for continuous data flow
  - Scatter-gather for a programmable list of arbitrary transfers initiated from a single request
- Highly flexible and configurable channel operation
  - Independently configured and operated channels
  - Dedicated channels for supported on-chip modules - USB, UART, GP Timer, ADC, EPI, SSI, I<sup>2</sup>S
  - Alternate channel assignments
  - One channel each for receive and transmit path for bidirectional modules
  - Dedicated channel for software-initiated transfers
  - Per-channel configurable bus arbitration scheme
  - Optional software-initiated requests for any channel
- Two levels of priority
- Design optimizations for improved bus access performance between  $\mu$ DMA controller and the processor core
  - $\mu$ DMA controller access is subordinate to core access
  - RAM striping
  - Peripheral bus segmentation
- Data sizes of 8, 16, and 32 bits
- Transfer size is programmable in binary steps from 1 to 1024
- Source and destination address increment size of byte, half-word, word, or no increment
- Maskable device requests
- Interrupt on transfer completion, with a separate interrupt per channel

#### 1.1.5.2 System Control and Clocks (see page 86)

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

- Device identification information - version, part number, SRAM size, Flash size, and so on
- Power control



- On-chip fixed Low Drop-Out (LDO) voltage regulator
- Low-power options for microcontroller: Sleep and Deep-sleep modes with clock gating
- Low-power options for on-chip modules: software controls shutdown of individual peripherals and memory
- 3.3-V supply brown-out detection and reporting via interrupt or reset
- Multiple clock sources for microcontroller system clock
  - Precision Oscillator (PIOSC) - on-chip resource providing a 16 MHz  $\pm$ 1% frequency at room temperature
    - 16 MHz  $\pm$ 3% across temperature
    - Software power down control for low power modes
  - Main Oscillator (MOSC) - 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.
    - External oscillator used with or without on-chip PLL - select supported frequencies from 1 MHz to 16.384 MHz.
    - External crystal - from DC to maximum device speed
  - Internal 30-kHz Oscillator - on chip resource providing a 30 kHz  $\pm$  50% frequency, used during power-saving modes
- Flexible reset sources
  - Power-on reset (POR)
  - Reset pin assertion
  - Brown-out reset (BOR) detector alerts to system power drops
  - Software reset
  - Watchdog timer reset
  - Internal low drop-out (LDO) regulator output goes unregulated

### 1.1.5.3 Four Programmable Timers (see page 381)

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

The Timer Module can be configured independently and has the following functional options:

- Count up or down
- 16- or 32-bit programmable one-shot timer

- 16- or 32-bit programmable periodic timer
- 16-bit general-purpose timer with an 8-bit prescaler
- 32-bit Real-Time Clock (RTC) when using an external 32.768-KHz clock as the input
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events
- ADC event trigger
- User-enabled stalling when the controller asserts CPU Halt flag during debug (excluding RTC mode)
- 16-bit input-edge count- or time-capture modes
- 16-bit PWM mode with software-programmable output inversion of the PWM signal
- Ability to determine the elapsed time between the assertion of the timer interrupt and entry into the interrupt service routine.
- Efficient transfers using Micro Direct Memory Access Controller ( $\mu$ DMA)
  - Dedicated channel for each timer
  - Burst request generated on timer interrupt

#### 1.1.5.4 CCP Pins (see page 386)

The LM3S5791 microcontroller includes eight Capture Compare PWM pins (CCP) which can be used by the General-Purpose Timer Module to time/count external events using the CCP pin as an input. Alternatively, the GPTM can generate a simple PWM output on the CCP pin. Each pin can be programmed to operate in the following modes:

- Capture - The GP Timer is incremented/decremented by programmed events on the CCP input. The GP Timer captures and stores the current timer value when a programmed event occurs.
- Compare - The GP Timer is incremented/decremented by programmed events on the CCP input. The GP Timer compares the current value with a stored value and generates an interrupt when a match occurs.
- PWM - The GP Timer is incremented/decremented by the system clock. A PWM signal is generated based on a match between the counter value and a value stored in a match register and is output on the CCP pin.

#### 1.1.5.5 Watchdog Timers (see page 420)

A watchdog timer is used to regain control when a system has failed due to a software error or to the failure of an external device to respond in the expected way. The Stellaris<sup>®</sup> Watchdog Timer can generate a nonmaskable interrupt (NMI) or a reset when a time-out value is reached. In addition, the Watchdog Timer is ARM FiRM-compliant and 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.

The LM3S5791 microcontroller has two Watchdog Timer modules: Watchdog Timer 0 uses the system clock for its timer clock; Watchdog Timer 1 uses the PIOSC as its timer clock. The Stellaris® Watchdog Timer module has the following features:

- 32-bit down counter with a programmable load register
- Separate watchdog clock with an enable
- Programmable interrupt generation logic with interrupt masking
- Lock register protection from runaway software
- Reset generation logic with an enable/disable
- User-enabled stalling when the controller asserts the CPU Halt flag during debug

#### 1.1.5.6 Programmable GPIOs (see page 284)

General-purpose input/output (GPIO) pins offer flexibility for a variety of connections. The Stellaris® GPIO module is comprised of nine physical GPIO blocks, each corresponding to an individual GPIO port. The GPIO module is FIRM-compliant (compliant to the ARM Foundation IP for Real-Time Microcontrollers specification) and supports 0-72 programmable input/output pins. The number of GPIOs available depends on the peripherals being used (see “Signal Tables” on page 905 for the signals available to each GPIO pin).

- 0-72 GPIOs, depending on configuration
- Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
- 5-V-tolerant input/outputs
- Fast toggle capable of a change every two clock cycles
- Two means of port access: either Advanced Host Bus (AHB) with better back-to-back access performance, or the legacy Advanced Peripheral Bus (APB) for backwards-compatibility with existing code
- Programmable control for GPIO interrupts
  - Interrupt generation masking
  - Edge-triggered on rising, falling, or both
  - Level-sensitive on High or Low values
- Bit masking in both read and write operations through address lines
- Can be used to initiate an ADC sample sequence
- Pins configured as digital inputs are Schmitt-triggered
- Programmable control for GPIO pad configuration
  - Weak pull-up or pull-down resistors

- 2-mA, 4-mA, and 8-mA pad drive for digital communication; up to four pads can be configured with an 18-mA pad drive for high-current applications
- Slew rate control for the 8-mA drive
- Open drain enables
- Digital input enables

### 1.1.6 Advanced Motion Control

The LM3S5791 controller provides motion control functions integrated into the device, including:

- Eight advanced PWM outputs for motion and energy applications
- Four fault inputs to promote low-latency shutdown
- Two Quadrature Encoder Inputs (QEI)

The following provides more detail on these motion control functions.

#### 1.1.6.1 PWM (see page 828)

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 LM3S5791 PWM module consists of four 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. 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. PWM generator block has the following features:

- 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 signal 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
- Can initiate an ADC sample sequence

The control block determines the polarity of the PWM signals and which signals are passed through to the pins. The output of the PWM generation blocks are managed by the output control block before being passed to the device pins. The PWM control block has the following options:

- 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
- Extended fault capabilities with multiple fault signals, programmable polarities, and filtering
- PWM generators can be operated independently or synchronized with other generators

#### 1.1.6.2 Fault Pins ( see page 834))

The LM3S5791 PWM module includes four fault-condition handling inputs to quickly provide low-latency shutdown and prevent damage to the motor being controlled.

#### 1.1.6.3 QEI (see page 887)

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, the position, direction of rotation, and speed can be tracked. In addition, a third channel, or index signal, can be used to reset the position counter. The Stellaris<sup>®</sup> 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 input frequency of the QEI inputs may be as high as 1/4 of the processor frequency (for example, 12.5 MHz for a 50-MHz system). The LM3S5791 microcontroller includes two QEI modules providing control of two motors at the same time. The QEI module has the following features:

- Position integrator that tracks the encoder position
- Programmable noise filter on the inputs
- Velocity capture using built-in timer
- Interrupt generation on:
  - Index pulse
  - Velocity-timer expiration

- Direction change
- Quadrature error detection

## 1.1.7 Analog

The LM3S5791 controller provides analog functions integrated into the device, including:

- Two 10-bit Analog-to-Digital Converters (ADC) with sixteen analog input channels and sample rate of one million samples/second
- Three analog comparators
- Digital Comparator
- On-chip voltage regulator

The following provides more detail on these analog functions.

### 1.1.7.1 ADC (see page 445)

An analog-to-digital converter (ADC) is a peripheral that converts a continuous analog voltage to a discrete digital number. The Stellaris<sup>®</sup> ADC module features 10-bit conversion resolution and supports sixteen input channels plus an internal temperature sensor. Four buffered sample sequencers allow rapid sampling of up to eight analog input sources without controller intervention. Each sample sequencer provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequencer priority. The LM3S5791 microcontroller provides two ADC modules. A digital comparator function is included which allows the conversion value to be diverted to a comparison unit that provides digital comparator. The ADC module has the following features:

- Sixteen analog input channels
- Single-ended and differential-input configurations
- On-chip internal temperature sensor
- Sample rate of one million samples/second
- Flexible, configurable analog-to-digital conversion
- Four programmable sample conversion sequencers 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

- Digital comparison unit providing digital comparator
- Converter uses an internal 3-V reference or an external reference
- Power and ground for the analog circuitry is separate from the digital power and ground
- Efficient transfers using Micro Direct Memory Access Controller ( $\mu$ DMA)
  - Dedicated channel for each sample sequencer
  - Burst request asserted when interrupt is triggered

#### 1.1.7.2 Analog Comparators (see page 816)

An analog comparator is a peripheral that compares two analog voltages and provides a logical output that signals the comparison result. The LM3S5791 microcontroller provides three independent integrated analog comparators that can be configured to drive an output or generate an interrupt or ADC event.

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. Each comparator has the following functions:

- Compare external pin input to external pin input or to internal programmable voltage reference
- Compare a test voltage against any one of these voltages
  - An individual external reference voltage
  - A shared single external reference voltage
  - A shared internal reference voltage

#### 1.1.8 JTAG and ARM Serial Wire Debug (see page 74)

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. Luminary Micro replaces the ARM SW-DP and JTAG-DP with the ARM CoreSight™-compliant Serial Wire JTAG Debug Port (SWJ-DP) interface. The SWJ-DP interface combines the SWD and JTAG debug ports into one module providing all the normal JTAG debug and test functionality plus real-time access to system memory without halting the core or requiring any target resident code. See the *CoreSight™ Design Kit Technical Reference Manual* for details on SWJ-DP. The SWJ-DP interface has the following features:

- IEEE 1149.1-1990 compatible Test Access Port (TAP) controller
- Four-bit Instruction Register (IR) chain for storing JTAG instructions
- IEEE standard instructions: BYPASS, IDCODE, SAMPLE/PRELOAD, EXTEST and INTTEST

- ARM additional instructions: APACC, DPACC and ABORT
- Integrated ARM Serial Wire Debug (SWD)
  - 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

### 1.1.9 Packaging and Temperature

- Industrial-range 100-pin RoHS-compliant LQFP package

## 1.2 Target Applications

The Stellaris<sup>®</sup> family is positioned for cost-conscious applications requiring significant control processing and connectivity capabilities such as:

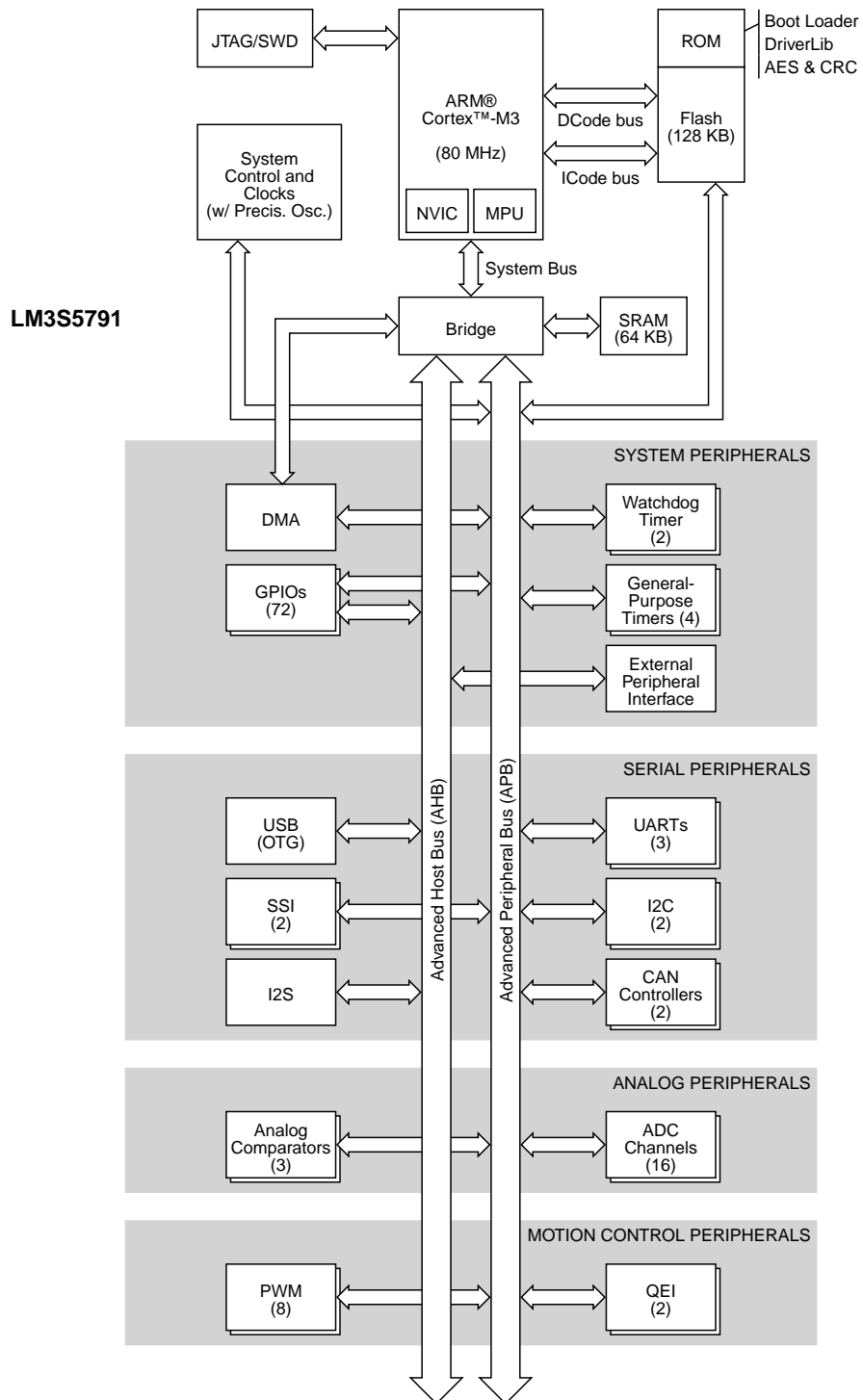
- Remote monitoring
- Electronic point-of-sale (POS) machines
- Test and measurement equipment
- Network appliances and switches
- Factory automation
- HVAC and building control
- Gaming equipment
- Motion control
- Medical instrumentation
- Fire and security
- Power and energy
- Transportation

## 1.3 High-Level Block Diagram

Figure 1-1 depicts the features on the Stellaris<sup>®</sup> LM3S5791 microcontroller. Note that there are two on-chip buses that connect the core to the peripherals. The Advanced Peripheral Bus (APB) bus is the legacy bus. The Advanced Host Bus (AHB) bus provides better back-to-back access performance than the APB bus.



Figure 1-1. Stellaris<sup>®</sup> LM3S5791 Microcontroller High-Level Block Diagram



## 1.4 Additional Features

### 1.4.1 Memory Map (see page 68)

A memory map lists the location of instructions and data in memory. The memory map for the LM3S5791 controller can be found in “Memory Map” on page 68. 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.

### 1.4.2 Hardware Details

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

- “Pin Diagram” on page 904
- “Signal Tables” on page 905
- “Operating Characteristics” on page 940
- “Electrical Characteristics” on page 941
- “Package Information” on page 965

## 2 ARM Cortex-M3 Processor Core

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

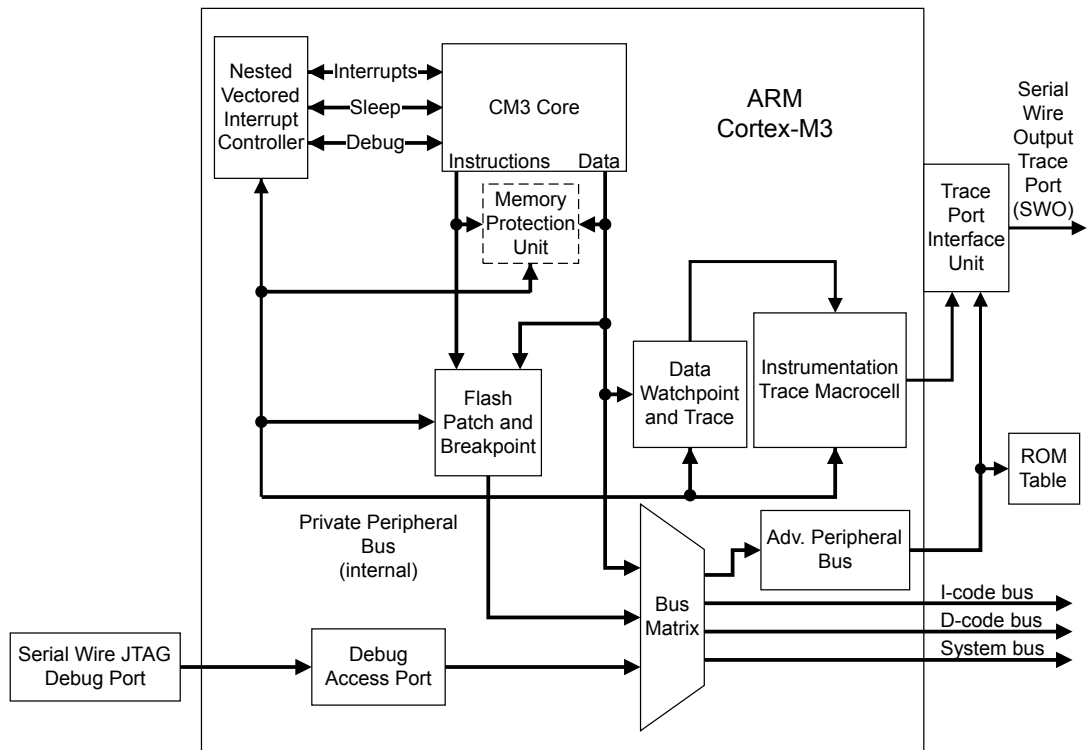
- 32-bit ARM® Cortex™-M3 v7M architecture optimized for small-footprint embedded applications
- Thumb-2 mixed 16-/32-bit instruction set, delivers the high performance expected of from a 32-bit ARM core in a compact memory size usually associated with 8- and 16-bit devices; typically in the range of a few kilobytes of memory for microcontroller-class applications
  - Single-cycle multiply instruction and hardware divide
  - Atomic bit manipulation (bit-banding), delivering maximum memory utilization and streamlined peripheral control
  - Unaligned data access, enabling data to be efficiently packed into memory
- Harvard architecture characterized by separate buses for instruction and data
- Memory protection unit (MPU) to provide a privileged mode for protected operating system functionality
- Migration from the ARM7™ processor family for better performance and power efficiency
- Optimized for single-cycle Flash usage
- 80-MHz operation
- 1.25 DMIPS/MHz

The Stellaris® family of microcontrollers builds on this core to bring high-performance 32-bit computing to cost-sensitive embedded microcontroller applications, such as factory automation and control, industrial control power devices, building and home automation, and stepper 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 implements the ARM Cortex-M3 core as shown in Figure 2-1 on page 56. The Cortex-M3 uses the entire 16-bit Thumb instruction set and the base Thumb-2 32-bit instruction set. In addition, 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 Programming Model

This section provides a brief overview of the programming model for the Cortex-M3 core. More detailed information can be found in the *ARM® Cortex™-M3 Technical Reference Manual*.

- Privileged access and user access - Code can execute as privileged or unprivileged. Unprivileged execution limits or excludes access to some resources. Privileged execution has access to all resources. Handler mode is always privileged. Thread mode can be privileged or unprivileged. Thread mode is privileged out of reset, but you can change it to user or unprivileged by setting the CONTROL[0] bit using the MSR instruction. User access prevents:
  - Use of some instructions such as CPS to set FAULTMASK and PRIMASK

- Access to most registers in System Control Space (SCS)

When Thread mode has been changed from privileged to user, it cannot change itself back to privileged. Only a Handler can change the privilege of Thread mode. Handler mode is always privileged.

- Register set - The processor has the following 32-bit registers:
  - 13 general-purpose registers, r0-r12
  - Stack point alias of banked registers, SP\_process and SP\_main
  - Link register, r14
  - Program counter, r15
  - One program status register, xPSR.
- Data types - The processor supports the following data types:
  - 32-bit words
  - 16-bit halfwords
  - 8-bit bytes
- Memory formats - The processor views memory as a linear collection of bytes numbered in ascending order from 0. For example, bytes 0-3 hold the first stored word and bytes 4-7 hold the second stored word. The processor accesses code and data in little-endian format. In little-endian format, the byte with the lowest address in a word is the least-significant byte of the word. The byte with the highest address in a word is the most significant. The byte at address 0 of the memory system connects to data lines 7-0.
- Instruction set - The Cortex-M3 instruction set contains both 16 and 32-bit instructions. These instructions are summarized in Table 2-1 on page 57 and Table 2-2 on page 59, respectively.

**Table 2-1. 16-Bit Cortex-M3 Instruction Set Summary**

Operation	Assembler
Add register value and C flag to register value	ADC <Rd>, <Rm>
Add immediate 3-bit value to register	ADD <Rd>, <Rn>, #<immed_3>
Add immediate 8-bit value to register	ADD <Rd>, #<immed_8>
Add low register value to low register value	ADD <Rd>, <Rn>, <Rm>
Add high register value to low or high register value	ADD <Rd>, <Rm>
Add 4* (immediate 8-bit value) with PC to register	ADD <Rd>, PC, #<immed_8> * 4
Add 4* (immediate 8-bit value) with SP to register	ADD <Rd>, SP, #<immed_8> * 4
Add 4* (immediate 7-bit value) to SP	ADD SP, #<immed_7> * 4
Bitwise AND register values	AND <Rd>, <Rm>
Arithmetic shift right by immediate number	ASR <Rd>, <Rm>, #<immed_5>
Arithmetic shift right by number in register	ASR <Rd>, <Rs>
Branch conditional	B<cond> <target address>
Branch unconditional	B <target_address>
Bit clear	BIC <Rd>, <Rm>

Operation	Assembler
Software breakpoint	BKPT <immed_8>
Branch with link	BL <Rm>
Branch with link and exchange	BLX <Rm>
Branch and exchange	BX <Rm>
Compare not zero and branch	CBNZ <Rn>, <label>
Compare zero and branch	CBZ <Rn>, <label>
Compare negation of register value with another register value	CMN <Rn>, <Rm>
Compare immediate 8-bit value	CMP <Rn>, #<immed_8>
Compare registers	CMP <Rn>, <Rm>
Compare high register to low or high register	CMP <Rn>, <Rm>
Change processor state	CPS <effect>, <iflags>
Copy high or low register value to another high or low register	CPY <Rd> <Rm>
Bitwise exclusive OR register values	EOR <Rd>, <Rm>
Condition the following instruction	IT <cond>
Condition the following two instructions	IT<x> <cond>
Condition the following three instructions	IT<x><y> <cond>
Condition the following four instructions	IT<x><y><z> <cond>
Multiple sequential memory word loads	LDMIA <Rn>!, <registers>
Load memory word from base register address + 5-bit immediate offset	LDR <Rd>, [<Rn>, #<immed_5> * 4]
Load memory word from base register address + register offset	LDR <Rd>, [<Rn>, <Rm>]
Load memory word from PC address + 8-bit immediate offset	LDR <Rd>, [PC, #<immed_8> * 4]
Load memory word from SP address + 8-bit immediate offset	LDR, <Rd>, [SP, #<immed_8> * 4]
Load memory byte [7:0] from register address + 5-bit immediate offset	LDRB <Rd>, [<Rn>, #<immed_5>]
Load memory byte [7:0] from register address + register offset	LDRB <Rd>, [<Rn>, <Rm>]
Load memory halfword [15:0] from register address + 5-bit immediate offset	LDRH <Rd>, [<Rn>, #<immed_5> * 2]
Load halfword [15:0] from register address + register offset	LDRH <Rd>, [<Rn>, <Rm>]
Load signed byte [7:0] from register address + register offset	LDRSB <Rd>, [<Rn>, <Rm>]
Load signed halfword [15:0] from register address + register offset	LDRSH <Rd>, [<Rn>, <Rm>]
Logical shift left by immediate number	LSL <Rd>, <Rm>, #<immed_5>
Logical shift left by number in register	LSL <Rd>, <Rs>
Logical shift right by immediate number	LSR <Rd>, <Rm>, #<immed_5>
Logical shift right by number in register	LSR <Rd>, <Rs>
Move immediate 8-bit value to register	MOV <Rd>, #<immed_8>
Move low register value to low register	MOV <Rd>, <Rn>
Move high or low register value to high or low register	MOV <Rd>, <Rm>
Multiply register values	MUL <Rd>, <Rm>
Move complement of register value to register	MVN <Rd>, <Rm>
Negate register value and store in register	NEG <Rd>, <Rm>
No operation	NOP <c>
Bitwise logical OR register values	ORR <Rd>, <Rm>
Pop registers from stack	POP <registers>
Pop registers and PC from stack	POP <registers, PC>
Push registers onto stack	PUSH <registers>

Operation	Assembler
Push LR and registers onto stack	PUSH <registers, LR>
Reverse bytes in word and copy to register	REV <Rd>, <Rn>
Reverse bytes in two halfwords and copy to register	REV16 <Rd>, <Rn>
Reverse bytes in low halfword [15:0], sign-extend, and copy to register	REVSH <Rd>, <Rn>
Rotate right by amount in register	ROR <Rd>, <Rs>
Subtract register value and C flag from register value	SBC <Rd>, <Rm>
Send event	SEV <c>
Store multiple register words to sequential memory locations	STMIA <Rn>!, <registers>
Store register word to register address + 5-bit immediate offset	STR <Rd>, [<Rn>, #<immed_5> * 4]
Store register word to register address	STR <Rd>, [<Rn>, <Rm>]
Store register word to SP address + 8-bit immediate offset	STR <Rd>, [SP, #<immed_8> * 4]
Store register byte [7:0] to register address + 5-bit immediate offset	STRB <Rd>, [<Rn>, #<immed_5>]
Store register byte [7:0] to register address	STRB <Rd>, [<Rn>, <Rm>]
Store register halfword [15:0] to register address + 5-bit immediate offset	STRH <Rd>, [<Rn>, #<immed_5> * 2]
Store register halfword [15:0] to register address + register offset	STRH <Rd>, [<Rn>, <Rm>]
Subtract immediate 3-bit value from register	SUB <Rd>, <Rn>, #<immed_3>
Subtract immediate 8-bit value from register value	SUB <Rd>, #<immed_8>
Subtract register values	SUB <Rd>, <Rn>, <Rm>
Subtract 4 (immediate 7-bit value) from SP	SUB SP, #<immed_7> * 4
Operating system service call with 8-bit immediate call code	SVC <immed_8>
Extract byte [7:0] from register, move to register, and sign-extend to 32 bits	SXTB <Rd>, <Rm>
Extract halfword [15:0] from register, move to register, and sign-extend to 32 bits	SXTH <Rd>, <Rm>
Test register value for set bits by ANDing it with another register value	TST <Rn>, <Rm>
Extract byte [7:0] from register, move to register, and zero-extend to 32 bits	UXTB <Rd>, <Rm>10
Extract halfword [15:0] from register, move to register, and zero-extend to 32 bits	UXTH <Rd>, <Rm>
Wait for event	WFE <c>
Wait for interrupt	WFI <c>

Table 2-2. 32-Bit Cortex-M3 Instruction Set Summary

Operation	Assembler
Add register value, immediate 12-bit value, and C bit	ADC{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
Add register value, shifted register value, and C bit	ADC{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
Add register value and immediate 12-bit value	ADD{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
Add register value and shifted register value	ADD{S}.W <Rd>, <Rm>{, <shift>}
Add register value and immediate 12-bit value	ADDW.W <Rd>, <Rn>, #<immed_12>
Bitwise AND register value with immediate 12-bit value	AND{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
Bitwise AND register value with shifted register value	AND{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
Arithmetic shift right by number in register	ASR{S}.W <Rd>, <Rn>, <Rm>
Conditional branch	B{cond}.W <label>
Clear bit field	BFC.W <Rd>, #<lsb>, #<width>
Insert bit field from one register value into another	BFI.W <Rd>, <Rn>, #<lsb>, #<width>

Operation	Assembler
Bitwise AND register value with complement of immediate 12-bit value	BIC{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
Bitwise AND register value with complement of shifted register value	BIC{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
Branch with link	BL <label>
Branch with link (immediate)	BL<c> <label>
Unconditional branch	B.W <label>
Clear exclusive clears the local record of the executing processor that an address has had a request for an exclusive access.	CLREX <c>
Return number of leading zeros in register value	CLZ.W <Rd>, <Rn>
Compare register value with two's complement of immediate 12-bit value	CMN.W <Rn>, #<modify_constant(immed_12)>
Compare register value with two's complement of shifted register value	CMN.W <Rn>, <Rm>{, <shift>}
Compare register value with immediate 12-bit value	CMP.W <Rn>, #<modify_constant(immed_12)>
Compare register value with shifted register value	CMP.W <Rn>, <Rm>{, <shift>}
Data memory barrier	DMB <c>
Data synchronization barrier	DSB <c>
Exclusive OR register value with immediate 12-bit value	EOR{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
Exclusive OR register value with shifted register value	EOR{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
Instruction synchronization barrier	ISB <c>
Load multiple memory registers, increment after or decrement before	LDM{IA DB}.W <Rn>{!}, <registers>
Memory word from base register address + immediate 12-bit offset	LDR.W <Rxf>, [<Rn>, #<offset_12>]
Memory word to PC from register address + immediate 12-bit offset	LDR.W PC, [<Rn>, #<offset_12>]
Memory word to PC from base register address immediate 8-bit offset, postindexed	LDR.W PC, [<Rn>], #<+/-<offset_8>
Memory word from base register address immediate 8-bit offset, postindexed	LDR.W <Rxf>, [<Rn>], #<+/-<offset_8>
Memory word from base register address immediate 8-bit offset, preindexed	LDR.W <Rxf>, [<Rn>, #<+/-<offset_8>!] LDRT.W <Rxf>, [<Rn>, #<offset_8>]
Memory word to PC from base register address immediate 8-bit offset, preindexed	LDR.W PC, [<Rn>, #<+/-<offset_8>!]
Memory word from register address shifted left by 0, 1, 2, or 3 places	LDR.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
Memory word to PC from register address shifted left by 0, 1, 2, or 3 places	LDR.W PC, [<Rn>, <Rm>{, LSL #<shift>}]
Memory word from PC address immediate 12-bit offset	LDR.W <Rxf>, [PC, #<+/-<offset_12>]
Memory word to PC from PC address immediate 12-bit offset	LDR.W PC, [PC, #<+/-<offset_12>]
Memory byte [7:0] from base register address + immediate 12-bit offset	LDRB.W <Rxf>, [<Rn>, #<offset_12>]
Memory byte [7:0] from base register address immediate 8-bit offset, postindexed	LDRB.W <Rxf>, [<Rn>], #<+/-<offset_8>
Memory byte [7:0] from register address shifted left by 0, 1, 2, or 3 places	LDRB.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
Memory byte [7:0] from base register address immediate 8-bit offset, preindexed	LDRB.W <Rxf>, [<Rn>, #<+/-<offset_8>!]
Memory byte from PC address immediate 12-bit offset	LDRB.W <Rxf>, [PC, #<+/-<offset_12>]
Memory doubleword from register address 8-bit offset 4, preindexed	LDRD.W <Rxf>, <Rxf2>, [<Rn>, #<+/-<offset_8> * 4]{!}



Operation	Assembler
Memory doubleword from register address 8-bit offset 4, postindexed	LDRD.W <Rxf>, <Rxf2>, [<Rn>], #+/-<offset_8> * 4
Load register exclusive calculates an address from a base register value and an immediate offset, loads a word from memory, writes it to a register	LDREX<c> <Rt>,[<Rn>{,<#imm>}]
Load register exclusive halfword calculates an address from a base register value and an immediate offset, loads a halfword from memory, writes it to a register	LDREXH<c> <Rt>,[<Rn>{,<#imm>}]
Load register exclusive byte calculates an address from a base register value and an immediate offset, loads a byte from memory, writes it to a register	LDREXB<c> <Rt>,[<Rn>{,<#imm>}]
Memory halfword [15:0] from base register address + immediate 12-bit offset	LDRH.W <Rxf>, [<Rn>], #<offset_12>]
Memory halfword [15:0] from base register address immediate 8-bit offset, preindexed	LDRH.W <Rxf>, [<Rn>], #<+/-<offset_8>]!
Memory halfword [15:0] from base register address immediate 8-bit offset, postindexed	LDRH.W <Rxf>.[<Rn>], #+/-<offset_8>
Memory halfword [15:0] from register address shifted left by 0, 1, 2, or 3 places	LDRH.W <Rxf>, [<Rn>], <Rm>{, LSL #<shift>}]
Memory halfword from PC address immediate 12-bit offset	LDRH.W <Rxf>, [PC, #+/-<offset_12>]
Memory signed byte [7:0] from base register address + immediate 12-bit offset	LDRSB.W <Rxf>, [<Rn>], #<offset_12>]
Memory signed byte [7:0] from base register address immediate 8-bit offset, postindexed	LDRSB.W <Rxf>.[<Rn>], #+/-<offset_8>
Memory signed byte [7:0] from base register address immediate 8-bit offset, preindexed	LDRSB.W <Rxf>, [<Rn>], #<+/-<offset_8>]!
Memory signed byte [7:0] from register address shifted left by 0, 1, 2, or 3 places	LDRSB.W <Rxf>, [<Rn>], <Rm>{, LSL #<shift>}]
Memory signed byte from PC address immediate 12-bit offset	LDRSB.W <Rxf>, [PC, #+/-<offset_12>]
Memory signed halfword [15:0] from base register address + immediate 12-bit offset	LDRSH.W <Rxf>, [<Rn>], #<offset_12>]
Memory signed halfword [15:0] from base register address immediate 8-bit offset, postindexed	LDRSH.W <Rxf>.[<Rn>], #+/-<offset_8>
Memory signed halfword [15:0] from base register address immediate 8-bit offset, preindexed	LDRSH.W <Rxf>, [<Rn>], #<+/-<offset_8>]!
Memory signed halfword [15:0] from register address shifted left by 0, 1, 2, or 3 places	LDRSH.W <Rxf>, [<Rn>], <Rm>{, LSL #<shift>}]
Memory signed halfword from PC address immediate 12-bit offset	LDRSH.W <Rxf>, [PC, #+/-<offset_12>]
Logical shift left register value by number in register	LSL{S}.W <Rd>, <Rn>, <Rm>
Logical shift right register value by number in register	LSR{S}.W <Rd>, <Rn>, <Rm>
Multiply two signed or unsigned register values and add the low 32 bits to a register value	MLA.W <Rd>, <Rn>, <Rm>, <Racc>
Multiply two signed or unsigned register values and subtract the low 32 bits from a register value	MLS.W <Rd>, <Rn>, <Rm>, <Racc>
Move immediate 12-bit value to register	MOV{S}.W <Rd>, #<modify_constant(immed_12)>
Move shifted register value to register	MOV{S}.W <Rd>, <Rm>{, <shift>}
Move immediate 16-bit value to top halfword [31:16] of register	MOVT.W <Rd>, #<immed_16>
Move immediate 16-bit value to bottom halfword [15:0] of register and clear top halfword [31:16]	MOVW.W <Rd>, #<immed_16>
Move to register from status	MRS<c> <Rd>, <psr>

Operation	Assembler
Move to status register	MSR<c> <psr>_<fields>,<Rn>
Multiply two signed or unsigned register values	MUL.W <Rd>, <Rn>, <Rm>
No operation	NOP.W
Logical OR NOT register value with immediate 12-bit value	ORN{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
Logical OR NOT register value with shifted register value	ORN{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
Logical OR register value with immediate 12-bit value	ORR{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
Logical OR register value with shifted register value	ORR{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
Reverse bit order	RBIT.W <Rd>, <Rm>
Reverse bytes in word	REV.W <Rd>, <Rm>
Reverse bytes in each halfword	REV16.W <Rd>, <Rn>
Reverse bytes in bottom halfword and sign-extend	REVSH.W <Rd>, <Rn>
Rotate right by number in register	ROR{S}.W <Rd>, <Rn>, <Rm>
Rotate right with extend	RRX{S}.W <Rd>, <Rm>
Subtract a register value from an immediate 12-bit value	RSB{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
Subtract a register value from a shifted register value	RSB{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
Subtract immediate 12-bit value and C bit from register value	SBC{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
Subtract shifted register value and C bit from register value	SBC{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
Copy selected bits to register and sign-extend	SBFX.W <Rd>, <Rn>, #<lsb>, #<width>
Signed divide	SDIV<c> <Rd>,<Rn>,<Rm>
Send event	SEV<c>
Multiply signed words and add signed-extended value to 2-register value	SMLAL.W <RdLo>, <RdHi>, <Rn>, <Rm>
Multiply two signed register values	SMULL.W <RdLo>, <RdHi>, <Rn>, <Rm>
Signed saturate	SSAT.W <c> <Rd>, #<imm>, <Rn>{, <shift>}
Multiple register words to consecutive memory locations	STM{IA DB}.W <Rn>{!}, <registers>
Register word to register address + immediate 12-bit offset	STR.W <Rxf>, [<Rn>, #<offset_12>]
Register word to register address immediate 8-bit offset, postindexed	STR.W <Rxf>, [<Rn>], #+/-<offset_8>
Register word to register address shifted by 0, 1, 2, or 3 places	STR.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
Register word to register address immediate 8-bit offset, preindexed Store, preindexed	STR.W <Rxf>, [<Rn>, #+/-<offset_8>]{!} STRT.W <Rxf>, [<Rn>, #<offset_8>]
Register byte [7:0] to register address immediate 8-bit offset, preindexed	STRB{T}.W <Rxf>, [<Rn>, #+/-<offset_8>]{!}
Register byte [7:0] to register address + immediate 12-bit offset	STRB.W <Rxf>, [<Rn>, #<offset_12>]
Register byte [7:0] to register address immediate 8-bit offset, postindexed	STRB.W <Rxf>, [<Rn>], #+/-<offset_8>
Register byte [7:0] to register address shifted by 0, 1, 2, or 3 places	STRB.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
Store doubleword, preindexed	STRD.W <Rxf>, <Rxf2>, [<Rn>, #+/-<offset_8> * 4]{!}
Store doubleword, postindexed	STRD.W <Rxf>, <Rxf2>, [<Rn>, #+/-<offset_8> * 4]
Store register exclusive calculates an address from a base register value and an immediate offset, and stores a word from a register to memory if the executing processor has exclusive access to the memory addressed.	STREX <c> <Rd>,<Rt>,<Rn>{,#<imm>}]

Operation	Assembler
Store register exclusive byte derives an address from a base register value, and stores a byte from a register to memory if the executing processor has exclusive access to the memory addressed	STREXB <c> <Rd>, <Rt>, [<Rn>]
Store register exclusive halfword derives an address from a base register value, and stores a halfword from a register to memory if the executing processor has exclusive access to the memory addressed.	STREXH <c> <Rd>, <Rt>, [<Rn>]
Register halfword [15:0] to register address + immediate 12-bit offset	STRH.W <Rxf>, [<Rn>, #<offset_12>]
Register halfword [15:0] to register address shifted by 0, 1, 2, or 3 places	STRH.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
Register halfword [15:0] to register address immediate 8-bit offset, preindexed	STRH{T}.W <Rxf>, [<Rn>, #+/-<offset_8>]{!}
Register halfword [15:0] to register address immediate 8-bit offset, postindexed	STRH.W <Rxf>, [<Rn>], #+/-<offset_8>
Subtract immediate 12-bit value from register value	SUB{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
Subtract shifted register value from register value	SUB{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
Subtract immediate 12-bit value from register value	SUBW.W <Rd>, <Rn>, #<immed_12>
Sign extend byte to 32 bits	SXTB.W <Rd>, <Rm>{, <rotation>}
Sign extend halfword to 32 bits	SXTH.W <Rd>, <Rm>{, <rotation>}
Table branch byte	TBB [<Rn>, <Rm>]
Table branch halfword	TBH [<Rn>, <Rm>, LSL #1]
Exclusive OR register value with immediate 12-bit value	TEQ.W <Rn>, #<modify_constant(immed_12)>
Exclusive OR register value with shifted register value	TEQ.W <Rn>, <Rm>{, <shift>}
Logical AND register value with 12-bit immediate value	TST.W <Rn>, #<modify_constant(immed_12)>
Logical AND register value with shifted register value	TST.W <Rn>, <Rm>{, <shift>}
Copy bit field from register value to register and zero-extend to 32 bits	UBFX.W <Rd>, <Rn>, #<lsb>, #<width>
Unsigned divide	UDIV<c> <Rd>, <Rn>, <Rm>
Multiply two unsigned register values and add to a 2-register value	UMLAL.W <RdLo>, <RdHi>, <Rn>, <Rm>
Multiply two unsigned register values	UMULL.W <RdLo>, <RdHi>, <Rn>, <Rm>
Unsigned saturate	USAT <c> <Rd>, #<imm>, <Rn>{, <shift>}
Copy unsigned byte to register and zero-extend to 32 bits	UXTB.W <Rd>, <Rm>{, <rotation>}
Copy unsigned halfword to register and zero-extend to 32 bits	UXTH.W <Rd>, <Rm>{, <rotation>}
Wait for event	WFE.W
Wait for interrupt	WFI.W

## 2.2.2 Serial Wire and JTAG Debug

Luminary Micro replaces the ARM SW-DP and JTAG-DP with the ARM CoreSight™-compliant Serial Wire JTAG Debug Port (SWJ-DP) interface. As a result, 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.

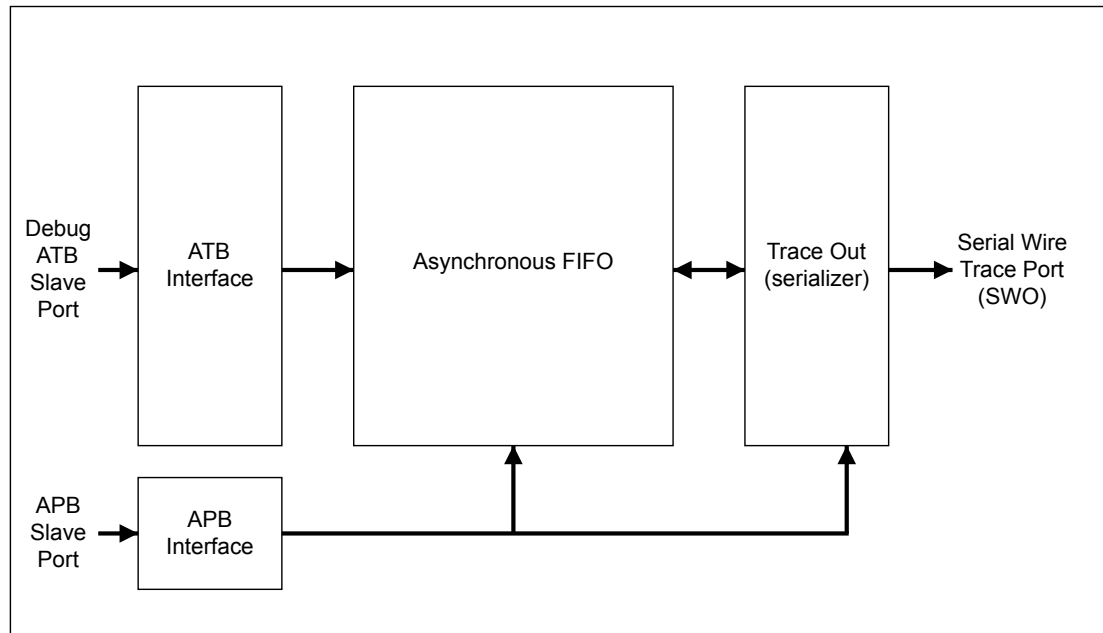
## 2.2.3 Embedded Trace Macrocell (ETM)

ETM is not implemented in the Stellaris® devices. As a result, Chapters 15 and 16 of the *ARM® Cortex™-M3 Technical Reference Manual* can be ignored.

## 2.2.4 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. Stellaris® devices implement the TPIU as shown in Figure 2-2. This implementation is similar to the non-ETM version described in the *ARM® Cortex™-M3 Technical Reference Manual*, however, SWJ-DP only provides the Serial Wire Viewer (SWV) output format for the TPIU.

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



## 2.2.5 ROM Table

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

## 2.2.6 Memory Protection Unit (MPU)

The Memory Protection Unit (MPU) is included on the LM3S5791 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.

## 2.2.7 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 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 by enabling the Configuration Control Register (see the ARM® Cortex™-M3 Technical Reference Manual). Any other user-mode access causes a bus fault.

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

### 2.2.7.1 Interrupts

The *ARM® Cortex™-M3 Technical Reference Manual* describes the maximum number of interrupts and interrupt priorities. The LM3S5791 microcontroller supports 52 interrupts with eight priority levels.

In addition to the peripheral interrupts, the system also provides for a non-maskable interrupt (NMI). The NMI is generally used in safety critical applications where the immediate execution of an interrupt handler is required. The NMI signal is available as an external signal so that it may be generated by external circuitry. The NMI is also used internally as part of the main oscillator verification circuitry. More information on the non-maskable interrupt is located in “Non-Maskable Interrupt” on page 89.

## 2.2.8 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 used 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.

### 2.2.8.1 Functional Description

The timer consists of three registers:

- SysTick Control and Status Register - a control and status counter to configure its clock, enable the counter, enable the SysTick interrupt, and determine counter status
- SysTick Reload Value Register - the reload value for the counter, used to provide the counter's wrap value
- SysTick Current Value Register - 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 on each clock 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. Clearing the SysTick 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 SysTick 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 does not decrement. The timer is clocked with respect to a reference clock, which can be either the core clock or an external clock source.

### 2.2.8.2 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	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	COUNTFLAG	R/W	0	Count Flag When set, this bit indicates that the timer has counted to 0 since the last time this register was read. This bit is cleared by a read of the register. If read by the debugger using the DAP, this bit is cleared only if the MasterType bit in the AHB-AP Control Register is clear. Otherwise, the COUNTFLAG bit is not changed by the debugger read.
15:3	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	CLKSOURCE	R/W	0	Clock Source  Value Description 0 External reference clock. (Not implemented for Stellaris <sup>®</sup> microcontrollers.) 1 Core clock  Because an external reference clock is not supported, this bit must be set in order for SysTick to operate.
1	TICKINT	R/W	0	Tick Interrupt When set, this bit causes an interrupt to be generated to the NVIC when SysTick counts to 0. When clear, interrupt generation is disabled. Software can use the COUNTFLAG to determine if the counter has ever reached 0.
0	ENABLE	R/W	0	Enable When set, this bit enables SysTick to operate in a multi-shot way. That is, the counter loads the Reload value and begins counting down. On reaching 0, the COUNTFLAG bit is set and an interrupt is generated if enabled by TICKINT. The counter then loads the Reload value again and begins counting. When this bit is clear, the counter is disabled.

### 2.2.8.3 SysTick Reload Value Register

The SysTick Reload Value Register specifies the start value to load into the SysTick Current Value Register when the counter reaches 0. The start value can be 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.

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

When configuring SysTick as a single-shot timer, a new value is written on each tick interrupt, and the actual count down value must be written. For example, if a tick is next required after 400 clock pulses, 400 must be written into the RELOAD field.

Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:0	RELOAD	R/W	-	Reload Value Value to load into the SysTick Current Value Register when the counter reaches 0.

#### 2.2.8.4 SysTick Current Value Register

The SysTick Current Value Register contains the current value of the counter.

Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:0	CURRENT	W1C	-	Current Value This field contains the 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.

#### 2.2.8.5 SysTick Calibration Value Register

The SysTick Calibration Value register is not implemented.

### 3 Memory Map

The memory map for the LM3S5791 controller is provided in Table 3-1.

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*.

Note that within the memory map, all reserved space returns a bus fault when read or written.

**Table 3-1. Memory Map**

Start	End	Description	For details, see page ...
<b>Memory</b>			
0x0000.0000	0x0001.FFFF	On-chip Flash	189
0x0002.0000	0x00FF.FFFF	Reserved	-
0x0100.0000	0x0100.4FFF	On-chip ROM	189
0x0100.5000	0x0100.5EFF	AES+CRC software in on-chip ROM	991
0x0100.5F00	0x1FFF.FFFF	Reserved	-
0x2000.0000	0x2000.FFFF	Bit-banded on-chip SRAM	189
0x2001.0000	0x21FF.FFFF	Reserved	-
0x2200.0000	0x221F.FFFF	Bit-band alias of 0x2000.0000 through 0x200F.FFFF	189
0x2220.0000	0x3FFF.FFFF	Reserved	-
<b>FIRM Peripherals</b>			
0x4000.0000	0x4000.0FFF	Watchdog timer 0	423
0x4000.1000	0x4000.1FFF	Watchdog timer 1	423
0x4000.2000	0x4000.3FFF	Reserved	-
0x4000.4000	0x4000.4FFF	GPIO Port A	293
0x4000.5000	0x4000.5FFF	GPIO Port B	293
0x4000.6000	0x4000.6FFF	GPIO Port C	293
0x4000.7000	0x4000.7FFF	GPIO Port D	293
0x4000.8000	0x4000.8FFF	SSI0	576
0x4000.9000	0x4000.9FFF	SSI1	576
0x4000.A000	0x4000.BFFF	Reserved	-
0x4000.C000	0x4000.CFFF	UART0	522
0x4000.D000	0x4000.DFFF	UART1	522
0x4000.E000	0x4000.EFFF	UART2	522
0x4000.F000	0x4001.FFFF	Reserved	-
<b>Peripherals</b>			
0x4002.0000	0x4002.07FF	I <sup>2</sup> C Master 0	617
0x4002.0800	0x4002.0FFF	I <sup>2</sup> C Slave 0	629
0x4002.1000	0x4002.17FF	I <sup>2</sup> C Master 1	617
0x4002.1800	0x4002.1FFF	I <sup>2</sup> C Slave 1	629
0x4002.2000	0x4002.3FFF	Reserved	-
0x4002.4000	0x4002.4FFF	GPIO Port E	293
0x4002.5000	0x4002.5FFF	GPIO Port F	293



Start	End	Description	For details, see page ...
0x4002.6000	0x4002.6FFF	GPIO Port G	293
0x4002.7000	0x4002.7FFF	GPIO Port H	293
0x4002.8000	0x4002.8FFF	PWM	838
0x4002.9000	0x4002.BFFF	Reserved	-
0x4002.C000	0x4002.CFFF	QEI0	891
0x4002.D000	0x4002.DFFF	QEI1	891
0x4002.E000	0x4002.FFFF	Reserved	-
0x4003.0000	0x4003.0FFF	Timer 0	393
0x4003.1000	0x4003.1FFF	Timer 1	393
0x4003.2000	0x4003.2FFF	Timer 2	393
0x4003.3000	0x4003.3FFF	Timer 3	393
0x4003.4000	0x4003.7FFF	Reserved	-
0x4003.8000	0x4003.8FFF	ADC0	461
0x4003.9000	0x4003.9FFF	ADC1	461
0x4003.A000	0x4003.BFFF	Reserved	-
0x4003.C000	0x4003.CFFF	Analog Comparators	816
0x4003.D000	0x4003.DFFF	GPIO Port J	293
0x4003.E000	0x4003.FFFF	Reserved	-
0x4004.0000	0x4004.0FFF	CAN0 Controller	688
0x4004.1000	0x4004.1FFF	CAN1 Controller	688
0x4004.2000	0x4004.FFFF	Reserved	-
0x4005.0000	0x4005.0FFF	USB	734
0x4005.1000	0x4005.3FFF	Reserved	-
0x4005.4000	0x4005.4FFF	I <sup>2</sup> S0	649
0x4005.5000	0x4005.7FFF	Reserved	-
0x4005.8000	0x4005.8FFF	GPIO Port A (AHB aperture)	293
0x4005.9000	0x4005.9FFF	GPIO Port B (AHB aperture)	293
0x4005.A000	0x4005.AFFF	GPIO Port C (AHB aperture)	293
0x4005.B000	0x4005.BFFF	GPIO Port D (AHB aperture)	293
0x4005.C000	0x4005.CFFF	GPIO Port E (AHB aperture)	293
0x4005.D000	0x4005.DFFF	GPIO Port F (AHB aperture)	293
0x4005.E000	0x4005.EFFF	GPIO Port G (AHB aperture)	293
0x4005.F000	0x4005.FFFF	GPIO Port H (AHB aperture)	293
0x4006.0000	0x4006.0FFF	GPIO Port J (AHB aperture)	293
0x4006.1000	0x400C.FFFF	Reserved	-
0x400D.0000	0x400D.FFFF	EPI0	346
0x400E.0000	0x400F.CFFF	Reserved	-
0x400F.D000	0x400F.DFFF	Flash control	193
0x400F.E000	0x400F.EFFF	System control	97
0x400F.F000	0x400F.FFFF	μDMA	240
0x4010.0000	0x41FF.FFFF	Reserved	-
0x4200.0000	0x43FF.FFFF	Bit-banded alias of 0x4000.0000 through 0x400F.FFFF	-

Start	End	Description	For details, see page ...
0x4400.0000	0xDFFF.FFFF	Reserved	-
<b>Private Peripheral Bus</b>			
0xE000.0000	0xE000.0FFF	Instrumentation Trace Macrocell (ITM)	ARM® Cortex™-M3 Technical Reference Manual
0xE000.1000	0xE000.1FFF	Data Watchpoint and Trace (DWT)	ARM® Cortex™-M3 Technical Reference Manual
0xE000.2000	0xE000.2FFF	Flash Patch and Breakpoint (FPB)	ARM® Cortex™-M3 Technical Reference Manual
0xE000.3000	0xE000.DFFF	Reserved	-
0xE000.E000	0xE000.EFFF	Nested Vectored Interrupt Controller (NVIC)	ARM® Cortex™-M3 Technical Reference Manual
0xE000.F000	0xE003.FFFF	Reserved	-
0xE004.0000	0xE004.0FFF	Trace Port Interface Unit (TPIU)	ARM® Cortex™-M3 Technical Reference Manual
0xE004.1000	0xFFFF.FFFF	Reserved	-

## 4 Interrupts

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

Table 4-1 on page 71 lists all exception types. Software can set eight priority levels on seven of these exceptions (system handlers) as well as on 52 interrupts (listed in Table 4-2 on page 72).

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. Priorities can be grouped by splitting priority levels into pre-emption priorities and subpriorities. All of the interrupt registers are described in Chapter 8, “Nested Vectored Interrupt Controller” in the *ARM® Cortex™-M3 Technical Reference Manual*.

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

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

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

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

**Table 4-1. Exception Types**

Exception Type	Vector Number	Priority <sup>a</sup>	Description
-	0	-	Stack top is loaded from the first entry of the vector table on reset.
Reset	1	-3 (highest)	This exception is invoked on power up and warm reset. On the first instruction, Reset drops to the lowest priority (and then is called the base level of activation). This exception is asynchronous.
Non-Maskable Interrupt (NMI)	2	-2	This exception is caused by the assertion of the NMI signal or by using the NVIC Interrupt Control State register and cannot be stopped or preempted by any exception but Reset. This exception is asynchronous.
Hard Fault	3	-1	This exception is caused by all classes of Fault, when the fault cannot activate due to priority or the configurable fault handler has been disabled. This exception is synchronous.
Memory Management	4	programmable	This exception is caused by an MPU mismatch, including access violation and no match. This exception is synchronous.

Exception Type	Vector Number	Priority <sup>a</sup>	Description
Bus Fault	5	programmable	This exception is caused by a pre-fetch fault, memory access fault, and other address/memory related faults. This exception is synchronous when precise and asynchronous when imprecise.  This fault can be enabled or disabled.
Usage Fault	6	programmable	This exception is caused by a usage fault, such as undefined instruction executed or illegal state transition attempt. This exception is synchronous.
-	7-10	-	Reserved.
SVCcall	11	programmable	This exception is caused by a system service call with an SVC instruction. This exception is synchronous.
Debug Monitor	12	programmable	This exception is caused by the debug monitor (when not halting). This exception is synchronous, but only active when enabled. This exception does not activate if it is a lower priority than the current activation.
-	13	-	Reserved.
PendSV	14	programmable	This exception is caused by a penable request for system service. This exception is asynchronous and only pended by software.
SysTick	15	programmable	This exception is caused by the SysTick timer reaching 0, when it is enabled to generate an interrupt. This exception is asynchronous.
Interrupts	16 and above	programmable	This exception is caused by interrupts asserted from outside the ARM Cortex-M3 core and fed through the NVIC (prioritized). These exceptions are all asynchronous. Table 4-2 on page 72 lists the interrupts on the LM3S5791 controller.

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

**Table 4-2. Interrupts**

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Description
0-15	-	Processor exceptions
16	0	GPIO Port A
17	1	GPIO Port B
18	2	GPIO Port C
19	3	GPIO Port D
20	4	GPIO Port E
21	5	UART0
22	6	UART1
23	7	SSI0
24	8	I <sup>2</sup> C0
25	9	PWM Fault
26	10	PWM Generator 0
27	11	PWM Generator 1
28	12	PWM Generator 2
29	13	QEIO
30	14	ADC0 Sequence 0
31	15	ADC0 Sequence 1
32	16	ADC0 Sequence 2
33	17	ADC0 Sequence 3
34	18	Watchdog Timers 0 and 1

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Description
35	19	Timer 0A
36	20	Timer 0B
37	21	Timer 1A
38	22	Timer 1B
39	23	Timer 2A
40	24	Timer 2B
41	25	Analog Comparator 0
42	26	Analog Comparator 1
43	27	Analog Comparator 2
44	28	System Control
45	29	Flash Control
46	30	GPIO Port F
47	31	GPIO Port G
48	32	GPIO Port H
49	33	UART2
50	34	SSI1
51	35	Timer 3A
52	36	Timer 3B
53	37	I <sup>2</sup> C1
54	38	QE11
55	39	CAN0
56	40	CAN1
57-59	41-43	Reserved
60	44	USB
61	45	PWM Generator 3
62	46	μDMA Software
63	47	μDMA Error
64	48	ADC1 Sequence 0
65	49	ADC1 Sequence 1
66	50	ADC1 Sequence 2
67	51	ADC1 Sequence 3
68	52	I <sup>2</sup> S0
69	53	EPI
70	54	GPIO Port J
71	55	Reserved

## 5 JTAG Interface

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

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

The Luminary Micro JTAG controller works with the ARM JTAG controller built into the Cortex-M3 core 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 output. The multiplexer is controlled by the Luminary Micro JTAG controller, which has comprehensive programming for the ARM, Luminary Micro, and unimplemented JTAG instructions.

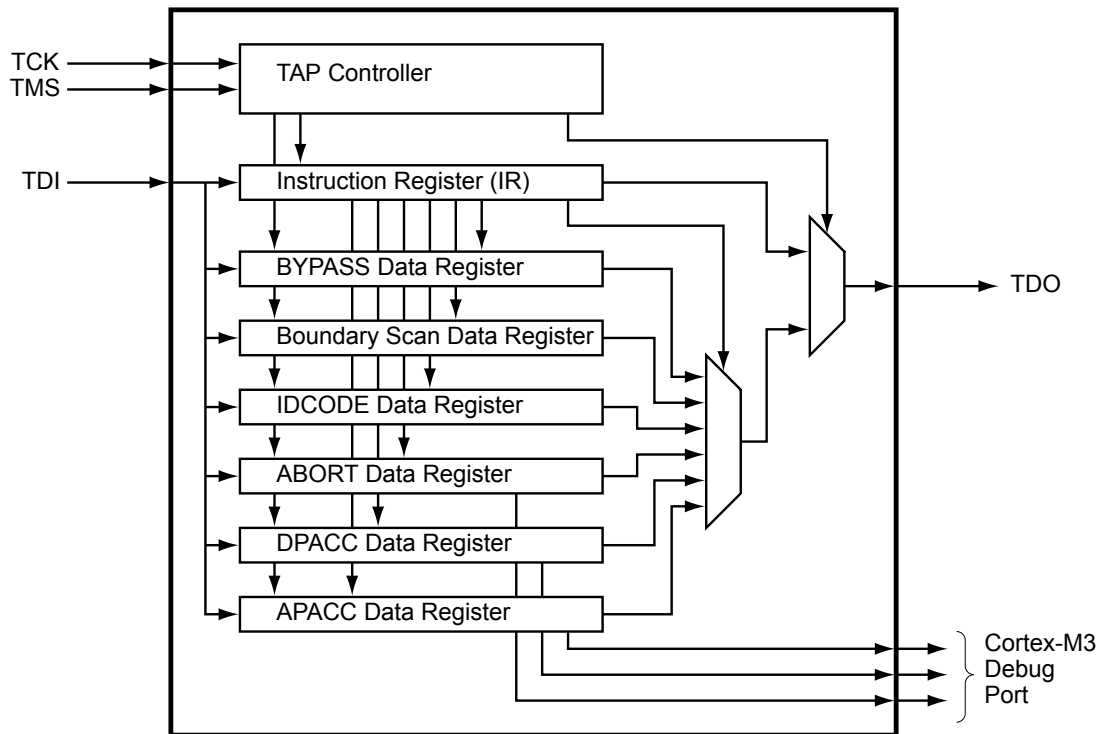
The Stellaris® JTAG module has the following features:

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

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

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

Some instructions, like  $EXTEST$  and  $INTEST$ , operate on data currently in a DR chain and do not capture, shift, or update any of the chains. Instructions that are not implemented decode to the  $BYPASS$  instruction to ensure that the serial path between  $TDI$  and  $TDO$  is always connected (see Table 5-2 on page 81 for a list of implemented instructions).

See “JTAG and Boundary Scan” on page 949 for JTAG timing diagrams.

**Note:** Of all the possible reset sources, only Power-On reset (POR) and the assertion of the  $\overline{RST}$  input have any effect on the JTAG module. The pin configurations are reset by both the

$\overline{\text{RST}}$  input and POR, whereas the internal JTAG logic is only reset with POR. See “Reset Sources” on page 86 for more information on reset.

## 5.2.1 JTAG Interface Pins

The JTAG interface consists of four standard pins: TCK, TMS, TDI, and TDO. These pins and their associated state after a power-on reset or reset caused by the  $\overline{\text{RST}}$  input are given in Table 5-1. Detailed information on each pin follows. Refer to “General-Purpose Input/Outputs (GPIOs)” on page 284 for information on how to reprogram the configuration of these pins.

**Table 5-1. JTAG Port Pins State after Power-On Reset or  $\overline{\text{RST}}$  assertion**

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

### 5.2.1.1 Test Clock Input (TCK)

The TCK pin is the clock for the JTAG module. This clock is provided so the test logic can operate independently of any other system clocks and to ensure 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, assuring 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 (see page 310 and page 312).

### 5.2.1.2 Test Mode Select (TMS)

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

Holding TMS high for five consecutive TCK cycles drives the TAP controller state machine to the Test-Logic-Reset state. When the TAP controller enters the Test-Logic-Reset state, the JTAG module and associated registers are reset to their default values. This procedure should be performed to initialize the JTAG controller. The JTAG Test Access Port state machine can be seen in its entirety in Figure 5-2 on page 78.

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 (see page 310).

### 5.2.1.3 Test Data Input (TDI)

The TDI pin provides a stream of serial information to the IR chain and the DR chains. TDI is sampled on the rising edge of TCK and, depending on the current TAP state and the current instruction, may present 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 (see page 310).

#### 5.2.1.4 Test Data Output (TDO)

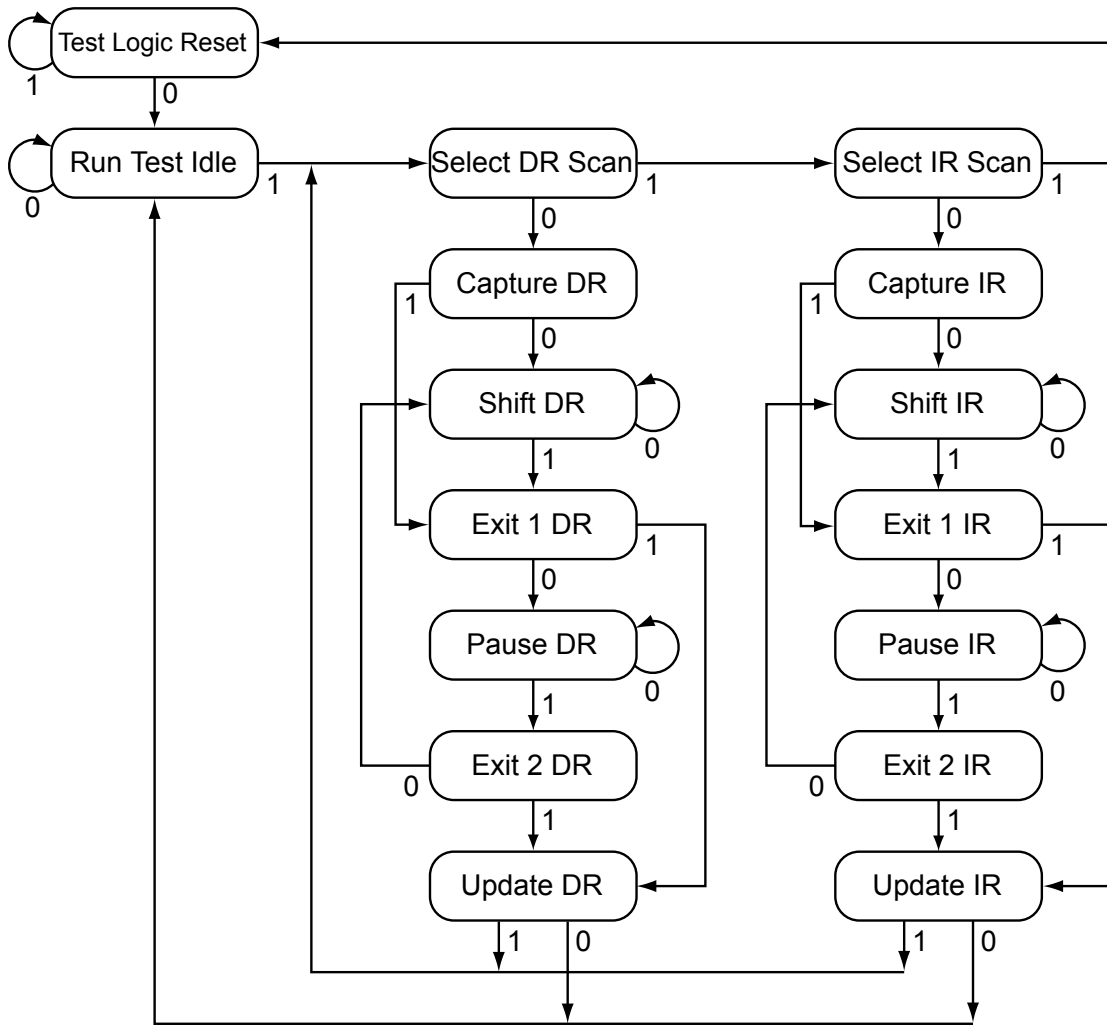
The TDO pin provides an output stream of serial information from the IR chain or the DR chains. The value of TDO depends on the current TAP state, the current instruction, and the data in the chain being accessed. In order to save power when the JTAG port is not being used, the TDO pin is placed in an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

By default, the internal pull-up resistor on the TDO pin is enabled after reset, assuring 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 (see page 310 and page 312).

#### 5.2.2 JTAG TAP Controller

The JTAG TAP controller state machine is shown in Figure 5-2. The TAP controller state machine is reset to the Test-Logic-Reset state on the assertion of a Power-On-Reset (POR). In order to reset the JTAG module after the microcontroller has been powered on, the TMS input must be held HIGH for five TCK clock cycles, resetting the TAP controller and all associated JTAG chains. 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 on 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 81.

### 5.2.4 Operational Considerations

Certain operational parameters must be considered 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 microcontroller is reset with either a POR or  $\overline{\text{RST}}$ , the JTAG/SWD port pins default to their JTAG/SWD configurations. The default configuration includes enabling digital functionality ( $\text{DEN}[3:0]$  set in the **Port C GPIO Digital Enable (GPIODEN)** register), enabling the pull-up resistors ( $\text{PUE}[3:0]$  set in the **Port C GPIO Pull-Up Select (GPIOPUR)** register) and enabling the alternate hardware function ( $\text{AFSEL}[3:0]$  set in the **Port C GPIO Alternate Function Select (GPIOAFSEL)** register) on the JTAG/SWD pins. See page 314, page 310 and page 304.

It is possible for software to configure these pins as GPIOs after reset by clearing  $\text{AFSEL}[3:0]$  in the **Port C GPIOAFSEL** register. If the user does not require the JTAG/SWD port for debugging or board-level testing, this provides four more GPIOs for use in the design.

---

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

---

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the NMI pin ( $\text{PB7}$ ) and the four JTAG/SWD pins ( $\text{PC}[3:0]$ ). Writes to protected bits of the **GPIOAFSEL** register, **GPIOPUR** register, **GPIO Pull-Down Select (GPIOPDR)** register (see page 312), and **GPIODEN** register are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 316) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 317) have been set.

### 5.2.4.2 Recovering a "Locked" Microcontroller

**Note:** Performing the sequence below restores the nonvolatile registers discussed in “Nonvolatile Register Programming” on page 192 to their factory default values. The mass erase of the Flash memory caused by the sequence below occurs prior to the nonvolatile registers being restored.

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

1. Assert and hold the  $\overline{\text{RST}}$  signal.
2. Perform steps 1. and 2. of the JTAG-to-SWD switch sequence on the section called “JTAG-to-SWD Switching” on page 80.
3. Perform steps 1. and 2. of the SWD-to-JTAG switch sequence on the section called “SWD-to-JTAG Switching” on page 80.
4. Perform steps 1. and 2. of the JTAG-to-SWD switch sequence.
5. Perform steps 1. and 2. of the SWD-to-JTAG switch sequence.
6. Perform steps 1. and 2. of the JTAG-to-SWD switch sequence.
7. Perform steps 1. and 2. of the SWD-to-JTAG switch sequence.

8. Perform steps 1. and 2. of the JTAG-to-SWD switch sequence.
9. Perform steps 1. and 2. of the SWD-to-JTAG switch sequence.
10. Perform steps 1. and 2. of the JTAG-to-SWD switch sequence.
11. Perform steps 1. and 2. of the SWD-to-JTAG switch sequence.
12. Release the  $\overline{\text{RST}}$  signal.
13. Wait 400 ms.
14. Power-cycle the microcontroller.

### 5.2.4.3 ARM Serial Wire Debug (SWD)

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

The switching preamble used to enable the SWD interface of the SWJ-DP module starts with the TAP controller in the Test-Logic-Reset state. From here, the preamble sequences the TAP controller through the following states: Run Test Idle, Select DR, Select IR, 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 sequence 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 instance is the only one 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 the switching preamble to the microcontroller. The 16-bit TMS command for switching to SWD mode is defined as b1110.0111.1001.1110, transmitted LSB first. This command can also be represented as 0xE79E 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 High to ensure that both JTAG and SWD are in their reset/idle states.
2. Send the 16-bit JTAG-to-SWD switch command, 0xE79E, on TMS.
3. Send at least 50 TCK/SWCLK cycles with TMS/SWDIO High to ensure that if SWJ-DP was already in SWD mode, the SWD goes into the line reset state before sending the switch sequence.

#### **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 command to the microcontroller. The 16-bit TMS

command for switching to JTAG mode is defined as b1110.0111.0011.1100, transmitted LSB first. This command can also be represented as 0xE73C 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 High to ensure that both JTAG and SWD are in their reset/idle states.
2. Send the 16-bit SWD-to-JTAG switch command, 0xE73C, on TMS.
3. Send at least 50 TCK/SWCLK cycles with TMS/SWDIO High to ensure that if SWJ-DP was already in JTAG mode, the JTAG goes into the Test Logic Reset state before sending the switch sequence.

## 5.3 Initialization and Configuration

After a Power-On-Reset or an external reset ( $\overline{\text{RST}}$ ), 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. To return the pins to their JTAG functions, enable the four JTAG pins (PC[3:0]) for their alternate function using the **GPIOAFSEL** register. In addition to enabling the alternate functions, any other changes to the GPIO pad configurations on the four JTAG pins (PC[3:0]) should be returned to their default settings.

## 5.4 Register Descriptions

The registers in the JTAG TAP Controller or Shift Register chains are not memory mapped and are not accessible through the on-chip Advanced Peripheral Bus (APB). Instead, the registers within the JTAG controller are all accessed serially through the TAP Controller. These registers include the Instruction Register and the six Data Registers.

### 5.4.1 Instruction Register (IR)

The JTAG TAP Instruction Register (IR) is a four-bit serial scan chain connected between the JTAG TDI and TDO pins with a parallel load register. When the TAP Controller is placed in the correct states, bits can be shifted into the IR. Once these bits have been shifted into the chain and updated, they are interpreted as the current instruction. The decode of the IR bits is shown in Table 5-2. 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
0x0	EXTEST	Drives the values preloaded into the Boundary Scan Chain by the SAMPLE/PRELOAD instruction onto the pads.
0x1	INTEST	Drives the values preloaded into the Boundary Scan Chain by the SAMPLE/PRELOAD instruction into the controller.
0x2	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.
0x8	ABORT	Shifts data into the ARM Debug Port Abort Register.
0xA	DPACC	Shifts data into and out of the ARM DP Access Register.
0xB	APACC	Shifts data into and out of the ARM AC Access Register.
0xE	IDCODE	Loads manufacturing information defined by the <i>IEEE Standard 1149.1</i> into the IDCODE chain and shifts it out.
0xF	BYPASS	Connects TDI to TDO through a single Shift Register chain.

IR[3:0]	Instruction	Description
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 is not associated with its own Data Register chain. Instead, 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. With tests that drive known values out of the controller, this instruction can be used to verify connectivity. While the EXTEST instruction is present in the Instruction Register, the Boundary Scan Data Register can be accessed to sample and shift out the current data and load new data into the Boundary Scan Data Register.

#### 5.4.1.2 INTEST Instruction

The INTEST instruction is not associated with its own Data Register chain. Instead, 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. With tests that drive known values into the controller, this instruction can be used for testing. It is important to note that although the RST input pin is on the Boundary Scan Data Register chain, it is only observable. While the INTEST instruction is present in the Instruction Register, the Boundary Scan Data Register can be accessed to sample and shift out the current data and load new data into the Boundary Scan Data Register.

#### 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 on 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. See “Boundary Scan Data Register” on page 84 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. See the “ABORT Data Register” on page 85 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. See “DPACC Data Register” on page 85 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. See “APACC Data Register” on page 84 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 input and output data streams. IDCODE is the default instruction loaded into the JTAG Instruction Register when a Power-On-Reset (POR) is asserted, or the Test-Logic-Reset state is entered. See “IDCODE Data Register” on page 83 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. See “BYPASS Data Register” on page 84 for more information.

### 5.4.2 Data Registers

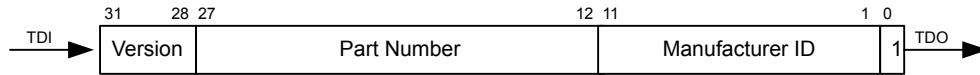
The JTAG module contains six Data Registers. These serial Data Register chains include: IDCODE, BYPASS, Boundary Scan, APACC, DPACC, and ABORT and are discussed in the following sections.

#### 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. The standard requires that every JTAG-compliant microcontroller 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 definition 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 0x412F.C230. This value indicates an ARM Cortex-M3, Version 1 processor and allows the debuggers to automatically configure themselves to work correctly with the Cortex-M3 during debug.

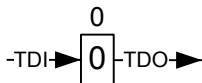
**Figure 5-3. IDCODE Register Format**



**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. The standard requires that every JTAG-compliant microcontroller 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 definition allows auto-configuration test tools to determine which instruction is the default instruction.

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

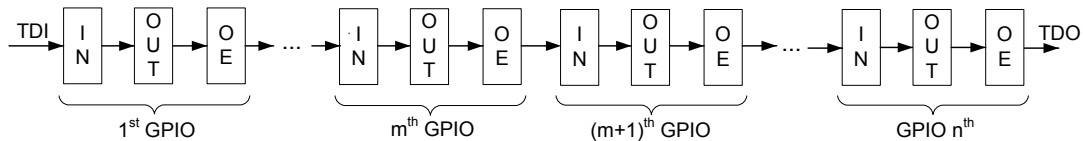


**5.4.2.3 Boundary Scan Data Register**

The format of the Boundary Scan Data Register is shown in Figure 5-5. Each GPIO pin, starting with a GPIO pin next to the JTAG port pins, is included in the Boundary Scan Data Register. Each GPIO pin has three associated digital signals that are included in the chain. These signals are input, output, and output enable, and are arranged in that order as shown in the figure. For detailed information on the order of the input, output, and output enable bits for each of the GPIO ports and any other pins included on the Boundary Scan Data Chain, please refer to the Stellaris® Family Boundary Scan Description Language (BSDL) files, downloadable from [www.luminarymicro.com](http://www.luminarymicro.com).

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. The EXTEST instruction forces data out of the controller, and the INTEST instruction forces data into the controller.

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



**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 configures the overall operation of the device and provides information about the device. Configurable features include reset control, NMI operation, power control, clock control, and low-power modes.

### 6.1 Functional Description

The System Control module provides the following capabilities:

- Device identification, see “Device Identification” on page 86
- Local control, such as reset (see “Reset Control” on page 86), power (see “Power Control” on page 90) and clock control (see “Clock Control” on page 90)
- System control (Run, Sleep, and Deep-Sleep modes), see “System Control” on page 95

#### 6.1.1 Device Identification

Several read-only registers provide software with information on the microcontroller, such as version, part number, SRAM size, Flash size, and other features. See the **DID0** (page 98), **DID1** (page 127), **DC0-DC9** (page 129) and **NVMSTAT** (page 153) registers.

#### 6.1.2 Reset Control

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

##### 6.1.2.1 Reset Sources

The LM3S5791 microcontroller has six sources of reset:

1. External reset input pin ( $\overline{\text{RST}}$ ) assertion (see page 87).
2. Power-on reset (POR) (see page 87).
3. Internal brown-out (BOR) detector (see page 88).
4. Software-initiated reset (with the software reset registers) (see page 88).
5. A watchdog timer reset condition violation (see page 89).
6. MOSC failure (see page 89).

Table 6-1 provides a summary of results of the various reset operations.

**Table 6-1. Reset Sources**

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

Reset Source	Core Reset?	JTAG Reset?	On-Chip Peripherals Reset?
MOSC Failure Reset	Yes	No	Yes

- By using the SYSRESETREQ bit in the ARM Cortex-M3 Application Interrupt and Reset Control register
- Programmable on a module-by-module basis using the Software Reset Control Registers.

After a reset, the **Reset Cause (RESC)** register is set with the reset cause. The bits in this register are sticky and maintain their state across multiple reset sequences, except when an internal POR is the cause, in which case, all the bits in the **RESC** register are cleared except for the POR indicator. A bit in the **RESC** register can be cleared by writing a 0.

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

The external reset pin ( $\overline{\text{RST}}$ ) resets the microcontroller including the core and all the on-chip peripherals except the JTAG TAP controller (see “JTAG Interface” on page 74). The external reset sequence is as follows:

- The external reset pin ( $\overline{\text{RST}}$ ) is asserted for the duration specified by  $T_{\text{MIN}}$  and then de-asserted (see “Reset” on page 950).
- A few clock cycles from  $\overline{\text{RST}}$  de-assertion to the start of the reset sequence is necessary for synchronization.
- The internal reset is released and the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

The external reset timing is shown in Figure 26-6 on page 950.

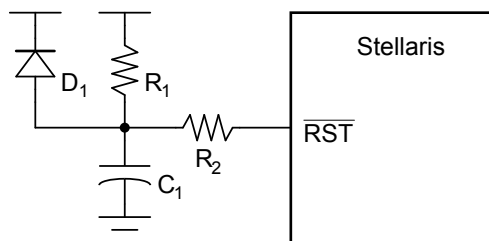
### 6.1.2.3 Power-On Reset (POR)

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

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

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

**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 microcontroller waits for both external reset ( $\overline{RST}$ ) and internal POR to go inactive.
2. The internal reset is released and the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

The internal POR is only active on the initial power-up of the microcontroller. The Power-On Reset timing is shown in Figure 26-7 on page 951.

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

The microcontroller 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 an 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; if  $BORIOR$  is clear, an interrupt is generated. The default condition is to generate an interrupt, so BOR must be enabled. When a Brown-out condition occurs during a Flash PROGRAM or ERASE operation, a full system reset is always triggered without regard to the setting in the **PBORCTL** register.

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

The internal Brown-Out Reset timing is shown in Figure 26-8 on page 951.

#### 6.1.2.5 Software Reset

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

Peripherals can be individually reset by software via three registers that control reset signals to each on-chip peripheral (see the **SRCRn** registers, see page 181). 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 95).

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

1. A software microcontroller reset is initiated by setting 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 microcontroller 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 26-9 on page 951.

### 6.1.2.6 Watchdog Timer Reset

The Watchdog Timer module's function is to prevent system hangs. The LM3S5791 microcontroller has two Watchdog Timer modules in case one watchdog clock source fails. One watchdog is run off the system clock and the other is run off the Precision Internal Oscillator (PIOSC). Each module operates in the same manner except that because the PIOSC watchdog timer module is in a different clock domain, register accesses must have a time delay between them. The watchdog timer can be configured to generate an interrupt to the microcontroller on its first time-out and to generate a reset on its second time-out.

After the watchdog's first time-out event, the 32-bit watchdog counter is reloaded with the value of the **Watchdog Timer Load (WDTLOAD)** register and resumes counting down from that value. If the timer counts down to zero 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 microcontroller. 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 microcontroller loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

For more information on the Watchdog Timer module, see "Watchdog Timer" on page 420.

The watchdog reset timing is shown in Figure 26-10 on page 951.

### 6.1.3 Non-Maskable Interrupt

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

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

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

#### 6.1.3.1 NMI Pin

The alternate function to GPIO port pin B7 is an NMI signal. The alternate function must be enabled in the GPIO for the signal to be used as an interrupt, as described in "General-Purpose Input/Outputs (GPIOs)" on page 284. Note that enabling the NMI alternate function requires the use of the GPIO lock and commit function just like the GPIO port pins associated with JTAG/SWD functionality, see page 317. The active sense of the NMI signal is High; asserting the enabled NMI signal above  $V_{IH}$  initiates the NMI interrupt sequence.

#### 6.1.3.2 Main Oscillator Verification Failure

The LM3S5791 microcontroller provides a main oscillator verification circuit that generates an error condition if the oscillator is running too fast or too slow. The main oscillator verification circuit can be programmed to generate a reset event, at which time a Power-on Reset is generated and control is transferred to the NMI handler. The NMI handler is used to address the main oscillator verification failure because the necessary code can be removed from the general reset handler, speeding up reset processing. The detection circuit is enabled by setting the `CVAL` bit in the **Main Oscillator**

**Control (MOSCCTL)** register. The main oscillator verification error is indicated in the main oscillator fail status (MOSCFAIL) bit in the **Reset Cause (RESC)** register. The main oscillator verification circuit action is described in more detail in “Main Oscillator Verification Circuit” on page 94.

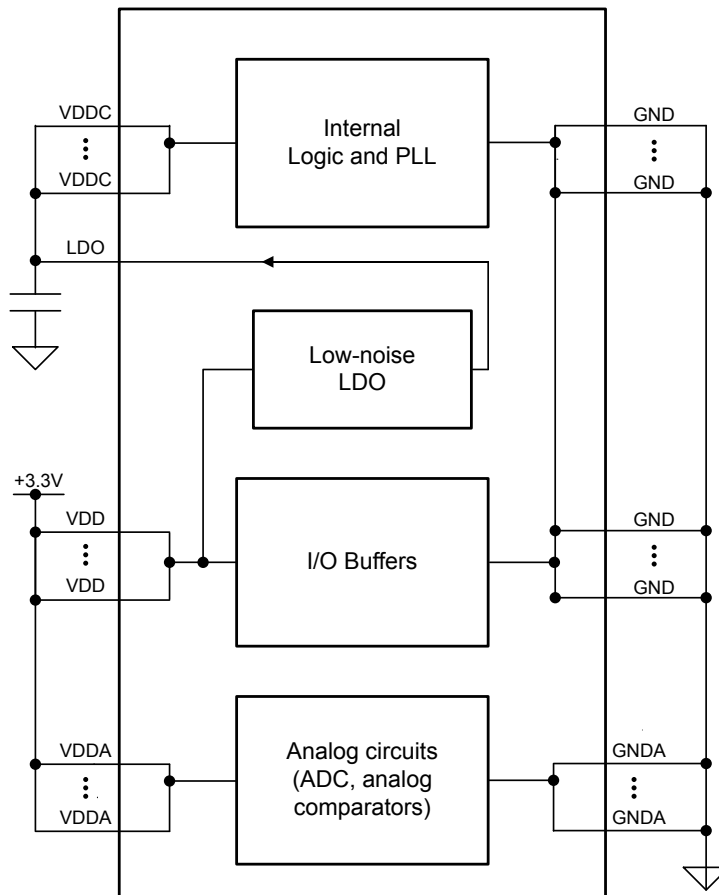
### 6.1.4 Power Control

The Stellaris<sup>®</sup> microcontroller provides an integrated LDO regulator that may be used to provide power to the majority of the microcontroller's internal logic. For power reduction, a non-programmable LDO may be used to scale the microcontroller's 3.3 V input voltage to 1.2V. The voltage output has a minimum voltage of 1.08 V and a maximum of 1.35 V. The LDO delivers up to 60 ma.

Figure 6-2 shows the power architecture.

**Note:** On the printed circuit board, use the LDO output as the source of VDDC input. In addition, the LDO requires decoupling capacitors. See “On-Chip Low Drop-Out (LDO) Regulator Characteristics” on page 942.

**Figure 6-2. Power Architecture**



### 6.1.5 Clock Control

System control determines the control of clocks in this part.

### 6.1.5.1 Fundamental Clock Sources

There are multiple clock sources for use in the microcontroller:

- **Precision Internal Oscillator (PIOSC).** The precision internal oscillator is an on-chip clock source. It does not require the use of any external components. The PIOSC provides a clock that is 16 MHz  $\pm$ 1% at room temperature and  $\pm$ 3% across temperature. Applications that do not depend on highly accurate clock sources may use this clock source to reduce system cost. The precision internal oscillator is the clock source the microcontroller uses during and following POR. If the main oscillator is required, software must enable the main oscillator following reset and allow the main oscillator to stabilize before changing the clock reference.
- **Main Oscillator (MOSC).** The main oscillator provides a frequency-accurate clock source by one of two means: an external single-ended clock source is connected to the OSC0 input pin, or an external crystal is connected across the OSC0 input and OSC1 output pins. If the PLL is being used, the crystal value must be one of the supported frequencies between 3.579545 MHz through 16.384 MHz (inclusive). If the PLL is not being used, the crystal may be any one of the supported frequencies between 1 MHz and 16.384 MHz. The single-ended clock source range is from DC through the specified speed of the microcontroller. The supported crystals are listed in the XTAL bit field in the RCC register (see page 109).
- **Internal 30-kHz Oscillator.** The internal 30-kHz oscillator is similar to the internal oscillator, except that it provides an operational frequency of 30 kHz  $\pm$  50%. It is intended for use during Deep-Sleep power-saving modes. This power-savings mode benefits from reduced internal switching and also allows the main oscillator to be powered down.

The internal system clock (SysClk), is derived from any of the above sources plus two others: the output of the main internal PLL and the precision internal oscillator divided by four (4 MHz  $\pm$  1%). The frequency of the PLL clock reference must be in the range of 3.579545 MHz to 16.384 MHz (inclusive). Table 6-2 on page 91 shows how the various clock sources can be used in a system.

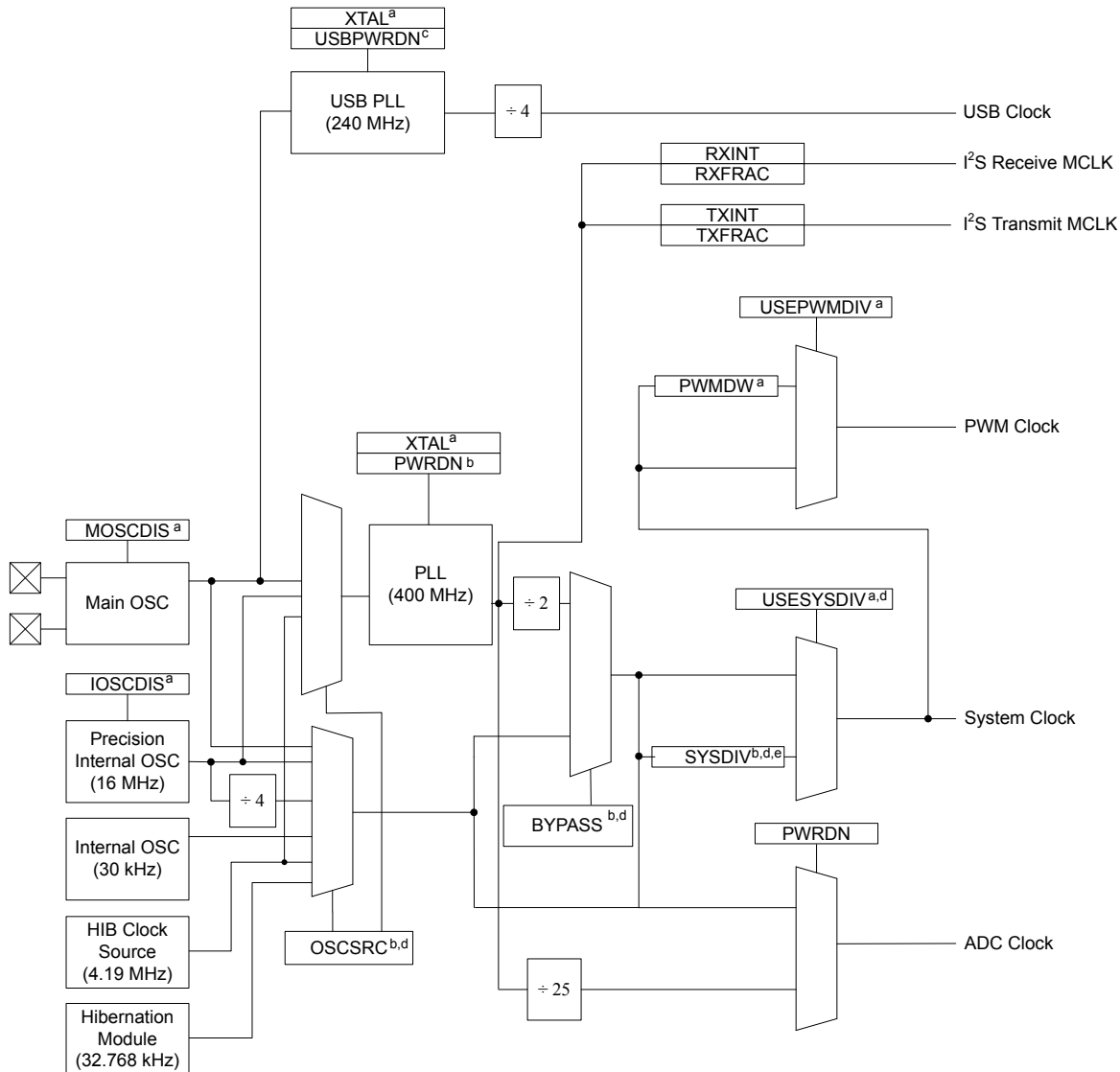
**Table 6-2. Clock Source Options**

Clock Source	Drive PLL?		Used as SysClk?	
	Yes	No	Yes	No
Precision Internal Oscillator	Yes	BYPASS = 0, OSCSRC = 0x1	Yes	BYPASS = 1, OSCSRC = 0x1
Internal Oscillator divide by 4 (4 MHz $\pm$ 1%)	No	BYPASS = 1	Yes	BYPASS = 1, OSCSRC = 0x2
Main Oscillator	Yes	BYPASS = 0, OSCSRC = 0x0	Yes	BYPASS = 1, OSCSRC = 0x0
Internal 30-kHz Oscillator	No	BYPASS = 1	Yes	BYPASS = 1, OSCSRC = 0x3

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

Figure 6-3 shows the logic for the main clock tree. The peripheral blocks are driven by the system clock signal and can be individually enabled/disabled. The ADC clock signal is automatically divided down to 16 MHz for proper ADC operation. The PWM clock signal is a synchronous divide of the system clock to provide the PWM circuit with more range (set with **PWMDIV** in **RCC**).

Figure 6-3. Main Clock Tree



- a. Control provided by RCC register bit/field.  
 b. Control provided by RCC register bit/field or RCC2 register bit/field, if overridden with RCC2 register bit USERCC2.  
 c. Control provided by RCC2 register bit/field.  
 d. Also may be controlled by DSLPCLKCFG when in deep sleep mode.  
 e. The USEFRACT and FRACT bit fields can also be used to influence the system clock for clock frequencies greater than 50 MHz..

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

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

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



The `XTAL` bit in the **RCC** register (see page 109) 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. Table 26-10 on page 947 shows the actual PLL frequency and error for a given crystal choice.

### 6.1.5.3 Main PLL Frequency Configuration

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

If the main oscillator provides the clock reference to the main PLL, the translation provided by hardware and used to program the PLL is available for software in the **XTAL to PLL Translation (PLLCFG)** register (see page 114). The internal translation provides a translation within  $\pm 1\%$  of the targeted PLL VCO frequency.

The Crystal Value field (`XTAL`) in the **Run-Mode Clock Configuration (RCC)** register (see page 109) describes the available crystal choices and default programming of the **PLLCFG** register. Any time the `XTAL` field changes, the new settings are translated and the internal PLL settings are updated.

The microcontroller powers up with the PIOSC running. To configure the PIOSC to be the clock source for the main PLL, program the `OSCR2` field in the **Run-Mode Clock Configuration 2 (RCC2)** register to be 0x1. If another clock source is desired, the PIOSC can be powered down by setting the `IOSCDIS` bit in the **RCC** register.

The PIOSC generates a 16 MHz clock with a  $\pm 1\%$  accuracy at room temperatures. Across the extended temperature range, the accuracy is  $\pm 3\%$ . At the factory, the PIOSC is set to 16 MHz at room temperature, however, the frequency can be trimmed for other voltage or temperature conditions using software in one of two ways:

- Default calibration: clear the `UTEN` bit and set the `UPDATE` bit in the **Precision Internal Oscillator Calibration (PIOCCAL)** register.
- User-defined calibration: The user can program the `UT` value to adjust the PIOSC frequency. As the `UT` value increases, the generated period increases. To commit a new `UT` value, first set the `UTEN` bit, then program the `UT` field, and then set the `UPDATE` bit. The adjustment finishes within a few clock periods and is glitch free.

### 6.1.5.4 USB PLL Frequency Configuration

The USB PLL is disabled by default during power-on reset and is enabled later by software. The USB PLL must be enabled and running for proper USB function. The main oscillator is the only clock reference for the USB PLL. The USB PLL is enabled by clearing the `USBPWRDN` bit of the **RCC2** register. The `XTAL` bit field (Crystal Value) of the **RCC** register describes the available crystal choices. The main oscillator must be connected to one of the following crystal values in order to correctly generate the USB clock: 4, 5, 6, 8, 10, 12, or 16 MHz. Only these crystals provide the necessary USB PLL VCO frequency to conform with the USB timing specifications.

### 6.1.5.5 PLL Modes

Both PLLs have 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 109 and page 117).

#### 6.1.5.6 PLL Operation

If a PLL configuration is changed, the PLL output frequency is unstable until it reconverges (relocks) to the new setting. The time between the configuration change and relock is  $T_{\text{READY}}$  (see Table 26-9 on page 947). During the relock time, the affected PLL is not usable as a clock reference.

Either PLL is changed by one of the following:

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

A counter is defined to measure the  $T_{\text{READY}}$  requirement. The counter is clocked by the main oscillator. The range of the main oscillator has been taken into account and the down counter is set to 0x1200 (that is, ~600  $\mu\text{s}$  at an 8.192 MHz external oscillator clock). When the  $XTAL$  value is greater than 0x0F, the down counter is set to 0x2400 to maintain the required lock time on higher frequency crystal inputs. Hardware is provided to keep the PLL from being used as a system clock until the  $T_{\text{READY}}$  condition is met after one of the two changes above. It is the user's responsibility to have a stable clock source (like the main oscillator) before the **RCC/RCC2** register is switched to use the PLL.

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

The USB PLL is not protected during the lock time ( $T_{\text{READY}}$ ), and software should ensure that the USB PLL has locked before using the interface. Software can use many methods to ensure the  $T_{\text{READY}}$  period has passed, including periodically polling the  $USBPLLRIS$  bit in the **Raw Interrupt Status (RIS)** register, and enabling the USB PLL Lock interrupt.

#### 6.1.5.7 Main Oscillator Verification Circuit

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

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

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

## 6.1.6 System Control

For power-savings purposes, the **RCGCn**, **SCGCn**, and **DCGCn** registers control the clock gating logic for each peripheral or block in the system while the microcontroller is in Run, Sleep, and Deep-Sleep mode, respectively. The **DC1**, **DC2** and **DC4** registers act as a write mask for the **RCGCn**, **SCGCn**, and **DCGCn** registers.

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

- **Run Mode.** In Run mode, the microcontroller actively executes code. Run mode provides normal operation of the processor and all of the peripherals that are currently enabled by the **RCGCn** registers. The system clock can be any of the available clock sources including the PLL.
- **Sleep Mode.** In Sleep mode, the clock frequency of the active peripherals is unchanged, but the processor and the memory subsystem are not clocked and therefore no longer execute code. Sleep mode is entered by the Cortex-M3 core executing a WFI (Wait for Interrupt) instruction. Any properly configured interrupt event in the system brings the processor back into Run mode. See the system control NVIC section of the *ARM® Cortex™-M3 Technical Reference Manual* for more details.

Peripherals are clocked that are enabled in the **SCGCn** register when auto-clock gating is enabled (see the **RCC** register) or the **RCGCn** register when the auto-clock gating is disabled. The system clock has the same source and frequency as that during Run mode.

- **Deep-Sleep Mode.** In Deep-Sleep mode, the clock frequency of the active peripherals may change (depending on the Run mode clock configuration) in addition to the processor clock being stopped. An interrupt returns the microcontroller to Run mode from one of the sleep modes; the sleep modes are entered on request from the code. 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 brings the processor back into Run mode. See the system control NVIC section of the *ARM® Cortex™-M3 Technical Reference Manual* for more details.

The Cortex-M3 processor core and the memory subsystem are not clocked. Peripherals are clocked that are enabled in the **DCGCn** register when auto-clock gating is enabled (see the **RCC** register) or the **RCGCn** register when auto-clock gating is disabled. The system clock source is the main oscillator by default or the internal oscillator specified in the **DSLPCCLKCFG** register if one is enabled. When the **DSLPCCLKCFG** register is used, the internal oscillator is powered up, if necessary, and the main oscillator is powered down. If the PLL is running at the time of the WFI instruction, hardware powers the PLL down and overrides the **SYSDIV** field of the active **RCC/RCC2** register, to be determined by the **DSDIVORIDE** setting in the **DSLPCCLKCFG** register, up to /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. If the PIOSC is used as the PLL reference clock source, it may continue to provide the clock during Deep-Sleep. See page 121.

## 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, thereby configuring the microcontroller to run off a “raw” clock source

and allowing 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 6-3 on page 96 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.

Additional Flash and ROM registers defined in the System Control register space are described in the "Internal Memory" on page 188.

**Table 6-3. System Control Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	DID0	RO	-	Device Identification 0	98
0x004	DID1	RO	-	Device Identification 1	127
0x008	DC0	RO	0x00FF.003F	Device Capabilities 0	129
0x010	DC1	RO	-	Device Capabilities 1	130
0x014	DC2	RO	0x570F.5337	Device Capabilities 2	133
0x018	DC3	RO	0xBFFF.B6FF	Device Capabilities 3	136
0x01C	DC4	RO	0x0000.F1FF	Device Capabilities 4	139
0x020	DC5	RO	0x0F30.00FF	Device Capabilities 5	141
0x024	DC6	RO	0x0000.0013	Device Capabilities 6	143
0x028	DC7	RO	0xFFFF.FFFF	Device Capabilities 7	144
0x02C	DC8	RO	0xFFFF.FFFF	Device Capabilities 8 ADC Channels	148
0x030	PBORCTL	R/W	0x0000.7FFD	Brown-Out Reset Control	100
0x040	SRCR0	R/W	0x00000000	Software Reset Control 0	181
0x044	SRCR1	R/W	0x00000000	Software Reset Control 1	183
0x048	SRCR2	R/W	0x00000000	Software Reset Control 2	186

Offset	Name	Type	Reset	Description	See page
0x050	RIS	RO	0x0000.0000	Raw Interrupt Status	101
0x054	IMC	R/W	0x0000.0000	Interrupt Mask Control	103
0x058	MISC	R/W1C	0x0000.0000	Masked Interrupt Status and Clear	105
0x05C	RESC	R/W	-	Reset Cause	107
0x060	RCC	R/W	0x078E.3AD1	Run-Mode Clock Configuration	109
0x064	PLLCFG	RO	-	XTAL to PLL Translation	114
0x06C	GPIOHBCTL	R/W	0x0000.0000	GPIO Host-Bus Control	115
0x070	RCC2	R/W	0x0780.6810	Run-Mode Clock Configuration 2	117
0x07C	MOSCCTL	R/W	0x0000.0000	Main Oscillator Control	120
0x100	RCGC0	R/W	0x00000040	Run Mode Clock Gating Control Register 0	154
0x104	RCGC1	R/W	0x00000000	Run Mode Clock Gating Control Register 1	163
0x108	RCGC2	R/W	0x00000000	Run Mode Clock Gating Control Register 2	175
0x110	SCGC0	R/W	0x00000040	Sleep Mode Clock Gating Control Register 0	157
0x114	SCGC1	R/W	0x00000000	Sleep Mode Clock Gating Control Register 1	167
0x118	SCGC2	R/W	0x00000000	Sleep Mode Clock Gating Control Register 2	177
0x120	DCGC0	R/W	0x00000040	Deep Sleep Mode Clock Gating Control Register 0	160
0x124	DCGC1	R/W	0x00000000	Deep-Sleep Mode Clock Gating Control Register 1	171
0x128	DCGC2	R/W	0x00000000	Deep Sleep Mode Clock Gating Control Register 2	179
0x144	DSLPCCLKCFG	R/W	0x0780.0000	Deep Sleep Clock Configuration	121
0x14C	DSFLASHCFG	R/W	0x0000.0000	Deep Sleep Flash Configuration	123
0x150	PIOSCCAL	R/W	0x0000.0000	Precision Internal Oscillator Calibration	124
0x170	I2SMCLKCFG	R/W	0x0000.0000	I2S MCLK Configuration	125
0x190	DC9	RO	0x00FF.00FF	Device Capabilities 9 ADC Digital Comparators	151
0x1A0	NVMSTAT	RO	0x0000.0001	Non-Volatile Memory Information	153

## 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

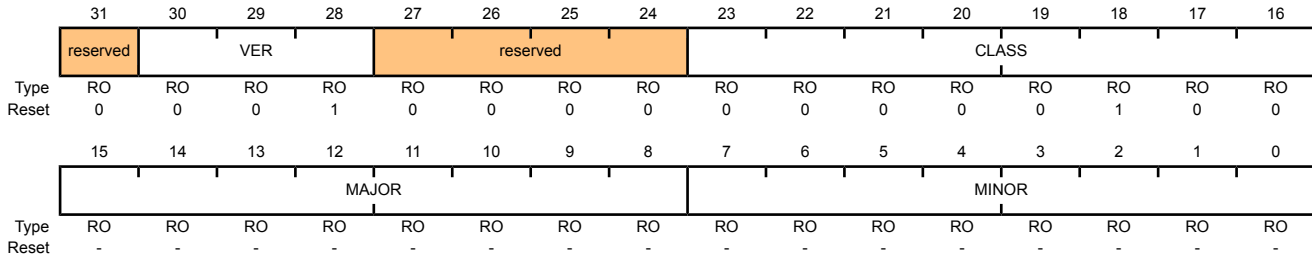
This register identifies the version of the microcontroller.

### 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	<p>DID0 Version</p> <p>This field defines the <b>DID0</b> register format version. The version number is numeric. The value of the <code>VER</code> field is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x1</td> <td>Second version of the <b>DID0</b> register format.</td> </tr> </tbody> </table>	Value	Description	0x1	Second version of the <b>DID0</b> register format.
Value	Description							
0x1	Second version of the <b>DID0</b> register format.							
27:24	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.				
23:16	CLASS	RO	0x04	<p>Device Class</p> <p>The <code>CLASS</code> field value identifies the internal design from which all mask sets are generated for all microcontrollers in a particular product line. The <code>CLASS</code> field value is changed for new product lines, for changes in fab process (for example, a remap or shrink), or any case where the <code>MAJOR</code> or <code>MINOR</code> fields require differentiation from prior microcontrollers. The value of the <code>CLASS</code> field is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x04</td> <td>Stellaris® Tempest-class microcontrollers</td> </tr> </tbody> </table>	Value	Description	0x04	Stellaris® Tempest-class microcontrollers
Value	Description							
0x04	Stellaris® Tempest-class microcontrollers							

Bit/Field	Name	Type	Reset	Description								
15:8	MAJOR	RO	-	<p>Major Revision</p> <p>This field specifies the major revision number of the microcontroller. 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 border="1"> <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 microcontroller. 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 border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Initial device, or a major revision update.</td> </tr> <tr> <td>0x1</td> <td>First metal layer change.</td> </tr> <tr> <td>0x2</td> <td>Second metal layer change.</td> </tr> </tbody> </table> <p>and so on.</p>	Value	Description	0x0	Initial device, or a major revision update.	0x1	First metal layer change.	0x2	Second metal layer change.
Value	Description											
0x0	Initial device, or a major revision update.											
0x1	First metal layer change.											
0x2	Second metal layer change.											

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

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

### Brown-Out Reset Control (PBORCTL)

Base 0x400F.E000

Offset 0x030

Type R/W, reset 0x0000.7FFD

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

Bit/Field	Name	Type	Reset	Description						
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
1	BORIOR	R/W	0	BOR Interrupt or Reset  <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>A Brown Out Event causes an interrupt to be generated to the interrupt controller.</td> </tr> <tr> <td>1</td> <td>A Brown Out Event causes a reset of the microcontroller.</td> </tr> </table>	Value	Description	0	A Brown Out Event causes an interrupt to be generated to the interrupt controller.	1	A Brown Out Event causes a reset of the microcontroller.
Value	Description									
0	A Brown Out Event causes an interrupt to be generated to the interrupt controller.									
1	A Brown Out Event causes a reset of the microcontroller.									
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: Raw Interrupt Status (RIS), offset 0x050

This register indicates the status for system control raw interrupts. An interrupt is sent to the interrupt controller if the corresponding bit in the **Interrupt Mask Control (IMC)** register is set. Writing a 1 to the corresponding bit in the **Masked Interrupt Status and Clear (MISC)** register clears an interrupt status bit.

#### Raw Interrupt Status (RIS)

Base 0x400F.E000  
Offset 0x050  
Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	MOSCPUPRIS	RO	0	<p>MOSC Power Up Raw Interrupt Status</p> <p>Value Description</p> <p>1 Sufficient time has passed for the MOSC to reach the expected frequency. The value for this power-up time is indicated by <math>T_{MOSC\_SETTLE}</math>.</p> <p>0 Sufficient time has not passed for the MOSC to reach the expected frequency.</p> <p>This bit is cleared by writing a 1 to the MOSCPUPMIS bit in the <b>MISC</b> register.</p>
7	USBPLLRRIS	RO	0	<p>USB PLL Lock Raw Interrupt Status</p> <p>Value Description</p> <p>1 The USB PLL timer has reached <math>T_{READY}</math> indicating that sufficient time has passed for the USB PLL to lock.</p> <p>0 The USB PLL timer has not reached <math>T_{READY}</math>.</p> <p>This bit is cleared by writing a 1 to the USBPLLRRMIS bit in the <b>MISC</b> register.</p>
6	PLLLRRIS	RO	0	<p>PLL Lock Raw Interrupt Status</p> <p>Value Description</p> <p>1 The PLL timer has reached <math>T_{READY}</math> indicating that sufficient time has passed for the PLL to lock.</p> <p>0 The PLL timer has not reached <math>T_{READY}</math>.</p> <p>This bit is cleared by writing a 1 to the PLLRRMIS bit in the <b>MISC</b> register.</p>

Bit/Field	Name	Type	Reset	Description
5:2	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	BORRIS	RO	0	<p>Brown-Out Reset Raw Interrupt Status</p> <p>Value Description</p> <p>1 A brown-out condition is currently active.</p> <p>0 A brown-out condition is not currently active.</p> <p>Note the BORIOR bit in the <b>PBORCTL</b> register must be cleared to cause an interrupt due to a Brown Out Event.</p> <p>This bit is cleared by writing a 1 to the BORMIS bit in the <b>MISC</b> register.</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.

## Register 4: Interrupt Mask Control (IMC), offset 0x054

This register contains the mask bits for system control raw interrupts. A raw interrupt, indicated by a bit being set in the **Raw Interrupt Status (RIS)** register, is sent to the interrupt controller if the corresponding bit in this register is set.

### Interrupt Mask Control (IMC)

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

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

Bit/Field	Name	Type	Reset	Description				
31:9	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.				
8	MOSCPUPIM	R/W	0	<p>MOSC Power Up Interrupt Mask</p> <p>This bit controls the reporting of the MOSC power up interrupt status to the interrupt controller.</p> <p>Value Description</p> <table border="0"> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the MOSCPUPRIS bit in the <b>RIS</b> register is set.</td> </tr> <tr> <td>0</td> <td>The MOSCPUPRIS interrupt is suppressed and not sent to the interrupt controller.</td> </tr> </table>	1	An interrupt is sent to the interrupt controller when the MOSCPUPRIS bit in the <b>RIS</b> register is set.	0	The MOSCPUPRIS interrupt is suppressed and not sent to the interrupt controller.
1	An interrupt is sent to the interrupt controller when the MOSCPUPRIS bit in the <b>RIS</b> register is set.							
0	The MOSCPUPRIS interrupt is suppressed and not sent to the interrupt controller.							
7	USBPLLIM	R/W	0	<p>USB PLL Lock Interrupt Mask</p> <p>This bit controls the reporting of the USB PLL Lock interrupt status to the interrupt controller.</p> <p>Value Description</p> <table border="0"> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the USBPLLRRIS bit in the <b>RIS</b> register is set.</td> </tr> <tr> <td>0</td> <td>The USBPLLRRIS interrupt is suppressed and not sent to the interrupt controller.</td> </tr> </table>	1	An interrupt is sent to the interrupt controller when the USBPLLRRIS bit in the <b>RIS</b> register is set.	0	The USBPLLRRIS interrupt is suppressed and not sent to the interrupt controller.
1	An interrupt is sent to the interrupt controller when the USBPLLRRIS bit in the <b>RIS</b> register is set.							
0	The USBPLLRRIS interrupt is suppressed and not sent to the interrupt controller.							

Bit/Field	Name	Type	Reset	Description				
6	PLLLIM	R/W	0	<p>PLL Lock Interrupt Mask</p> <p>This bit controls the reporting of the PLL Lock interrupt status to the interrupt controller.</p> <p>Value Description</p> <table><tbody><tr><td>1</td><td>An interrupt is sent to the interrupt controller when the <code>PLLLRIS</code> bit in the <b>RIS</b> register is set.</td></tr><tr><td>0</td><td>The <code>PLLLRIS</code> interrupt is suppressed and not sent to the interrupt controller.</td></tr></tbody></table>	1	An interrupt is sent to the interrupt controller when the <code>PLLLRIS</code> bit in the <b>RIS</b> register is set.	0	The <code>PLLLRIS</code> interrupt is suppressed and not sent to the interrupt controller.
1	An interrupt is sent to the interrupt controller when the <code>PLLLRIS</code> bit in the <b>RIS</b> register is set.							
0	The <code>PLLLRIS</code> interrupt is suppressed and not sent to the interrupt controller.							
5:2	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	BORIM	R/W	0	<p>Brown-Out Reset Interrupt Mask</p> <p>This bit controls the reporting of the Brown-Out Reset interrupt status to the interrupt controller.</p> <p>Value Description</p> <table><tbody><tr><td>1</td><td>An interrupt is sent to the interrupt controller when the <code>BORRIS</code> bit in the <b>RIS</b> register is set.</td></tr><tr><td>0</td><td>The <code>BORRIS</code> interrupt is suppressed and not sent to the interrupt controller.</td></tr></tbody></table>	1	An interrupt is sent to the interrupt controller when the <code>BORRIS</code> bit in the <b>RIS</b> register is set.	0	The <code>BORRIS</code> interrupt is suppressed and not sent to the interrupt controller.
1	An interrupt is sent to the interrupt controller when the <code>BORRIS</code> bit in the <b>RIS</b> register is set.							
0	The <code>BORRIS</code> interrupt is suppressed and not sent to the interrupt controller.							
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: Masked Interrupt Status and Clear (MISC), offset 0x058

On a read, this register gives the current masked status value of the corresponding interrupt in the **Raw Interrupt Status (RIS)** register. All of the bits are R/W1C, thus writing a 1 to a bit clears the corresponding raw interrupt bit in the **RIS** register (see page 101).

### Masked Interrupt Status and Clear (MISC)

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

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

Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	MOSCPUPMIS	R/W1C	0	MOSC Power Up Masked Interrupt Status
				<p><b>Value Description</b></p> <p>1 When read, a 1 indicates that an unmasked interrupt was signaled because sufficient time has passed for the MOSC PLL to lock.</p> <p>Writing a 1 to this bit clears it and also the <code>MOSCPUPRIS</code> bit in the <b>RIS</b> register.</p> <p>0 When read, a 0 indicates that sufficient time has not passed for the MOSC PLL to lock.</p> <p>A write of 0 has no effect on the state of this bit.</p>
7	USBPLLMISS	R/W1C	0	USB PLL Lock Masked Interrupt Status
				<p><b>Value Description</b></p> <p>1 When read, a 1 indicates that an unmasked interrupt was signaled because sufficient time has passed for the USB PLL to lock.</p> <p>Writing a 1 to this bit clears it and also the <code>USBPLLRRIS</code> bit in the <b>RIS</b> register.</p> <p>0 When read, a 0 indicates that sufficient time has not passed for the USB PLL to lock.</p> <p>A write of 0 has no effect on the state of this bit.</p>

Bit/Field	Name	Type	Reset	Description
6	PLLLMIS	R/W1C	0	PLL Lock Masked Interrupt Status  Value Description 1 When read, a 1 indicates that an unmasked interrupt was signaled because sufficient time has passed for the PLL to lock.  Writing a 1 to this bit clears it and also the PLLLRIS bit in the <b>RIS</b> register. 0 When read, a 0 indicates that sufficient time has not passed for the PLL to lock.  A write of 0 has no effect on the state of this bit.
5:2	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	BORMIS	R/W1C	0	BOR Masked Interrupt Status  Value Description 1 When read, a 1 indicates that an unmasked interrupt was signaled because of a brown-out condition.  Writing a 1 to this bit clears it and also the BORRIS bit in the <b>RIS</b> register. 0 When read, a 0 indicates that a brown-out condition has not occurred.  A write of 0 has no effect on the state of this bit.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 6: 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 power-on reset is the cause, in which case, all bits other than POR in the **RESC** register are cleared.

### Reset Cause (RESC)

Base 0x400F.E000

Offset 0x05C

Type R/W, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															MOSCFAIL
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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											SW	WDT0	BOR	POR	EXT
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

16	MOSCFAIL	R/W	-	MOSC Failure Reset
----	----------	-----	---	--------------------

#### Value Description

- 1 When read, this bit indicates that the MOSC circuit was enabled for clock validation and failed, generating a reset event.
- 0 When read, this bit indicates that a MOSC failure has not generated a reset since the previous power-on reset.

Writing a 0 to this bit clears it.

15:5	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.
------	----------	----	------	---

5	WDT1	R/W	-	Watchdog Timer 1 Reset
---	------	-----	---	------------------------

#### Value Description

- 1 When read, this bit indicates that Watchdog Timer 1 timed out and generated a reset.
- 0 When read, this bit indicates that Watchdog Timer 1 has not generated a reset since the previous power-on reset.

Writing a 0 to this bit clears it.

Bit/Field	Name	Type	Reset	Description
4	SW	R/W	-	<p>Software Reset</p> <p>Value Description</p> <p>1 When read, this bit indicates that a software reset has caused a reset event.</p> <p>0 When read, this bit indicates that a software reset has not generated a reset since the previous power-on reset.</p> <p>Writing a 0 to this bit clears it.</p>
3	WDT0	R/W	-	<p>Watchdog Timer 0 Reset</p> <p>Value Description</p> <p>1 When read, this bit indicates that Watchdog Timer 0 timed out and generated a reset.</p> <p>0 When read, this bit indicates that Watchdog Timer 0 has not generated a reset since the previous power-on reset.</p> <p>Writing a 0 to this bit clears it.</p>
2	BOR	R/W	-	<p>Brown-Out Reset</p> <p>Value Description</p> <p>1 When read, this bit indicates that a brown-out reset has caused a reset event.</p> <p>0 When read, this bit indicates that a brown-out reset has not generated a reset since the previous power-on reset.</p> <p>Writing a 0 to this bit clears it.</p>
1	POR	R/W	-	<p>Power-On Reset</p> <p>Value Description</p> <p>1 When read, this bit indicates that a power-on reset has caused a reset event.</p> <p>0 When read, this bit indicates that a power-on reset has not generated a reset.</p> <p>Writing a 0 to this bit clears it.</p>
0	EXT	R/W	-	<p>External Reset</p> <p>Value Description</p> <p>1 When read, this bit indicates that an external reset (<math>\overline{RST}</math> assertion) has caused a reset event.</p> <p>0 When read, this bit indicates that an external reset (<math>\overline{RST}</math> assertion) has not caused a reset event since the previous power-on reset.</p> <p>Writing a 0 to this bit clears it.</p>



## Register 7: Run-Mode Clock Configuration (RCC), offset 0x060

The bits in this register configure the system clock and oscillators.

### Run-Mode Clock Configuration (RCC)

Base 0x400F.E000

Offset 0x060

Type R/W, reset 0x078E.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	R/W	RO	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	XTAL				OSCSRC		reserved		IOSCDIS	MOSCDIS	
Type	RO	RO	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO	R/W	R/W
Reset	0	0	1	1	1	0	1	0	1	1	0	1	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:28	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
27	ACG	R/W	0	<p>Auto Clock Gating</p> <p>This bit specifies whether the system uses the <b>Sleep-Mode Clock Gating Control (SCGCn)</b> registers and <b>Deep-Sleep-Mode Clock Gating Control (DCGCn)</b> registers if the microcontroller enters a Sleep or Deep-Sleep mode (respectively).</p> <p>Value Description</p> <p>1 The <b>SCGCn</b> or <b>DCGCn</b> registers are used to control the clocks distributed to the peripherals when the microcontroller is in a sleep mode. The <b>SCGCn</b> and <b>DCGCn</b> registers allows unused peripherals to consume less power when the microcontroller is in a sleep mode.</p> <p>0 The <b>Run-Mode Clock Gating Control (RCGCn)</b> registers are used when the microcontroller enters a sleep mode.</p>

The **RCGCn** registers are always used to control the clocks in Run mode.

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>Although the PLL VCO frequency is 400 MHz, it is predivided by 2 before the divisor is applied.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Divisor (BYPASS=1)</th> <th>Frequency (BYPASS=0)</th> </tr> </thead> <tbody> <tr><td>0x0</td><td>reserved</td><td>reserved</td></tr> <tr><td>0x1</td><td>/2</td><td>reserved</td></tr> <tr><td>0x2</td><td>/3</td><td>80 MHz</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>If the SYSDIV value is less than MINSYSDIV (see page 130), and the PLL is being used, then the MINSYSDIV value is used as the divisor.</p> <p>If the PLL is not being used, the SYSDIV value can be less than MINSYSDIV.</p>	Value	Divisor (BYPASS=1)	Frequency (BYPASS=0)	0x0	reserved	reserved	0x1	/2	reserved	0x2	/3	80 MHz	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	80 MHz																																																					
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	USESYS DIV	R/W	0	<p>Enable System Clock Divider</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>The system clock divider is the source for the system clock. The system clock divider is forced to be used when the PLL is selected as the source.</td> </tr> <tr> <td>0</td> <td>The system clock is used undivided.</td> </tr> </tbody> </table>	Value	Description	1	The system clock divider is the source for the system clock. The system clock divider is forced to be used when the PLL is selected as the source.	0	The system clock is used undivided.																																													
Value	Description																																																						
1	The system clock divider is the source for the system clock. The system clock divider is forced to be used when the PLL is selected as the source.																																																						
0	The system clock is used undivided.																																																						
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 Divisor</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>The PWM clock divider is the source for the PWM clock.</td> </tr> <tr> <td>0</td> <td>The system clock is the source for the PWM clock.</td> </tr> </tbody> </table>	Value	Description	1	The PWM clock divider is the source for the PWM clock.	0	The system clock is the source for the PWM clock.																																													
Value	Description																																																						
1	The PWM clock divider is the source for the PWM clock.																																																						
0	The system clock is the source for the PWM clock.																																																						

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. The rising edge of this clock is synchronous with the system clock.</p> <p>Value Divisor</p> <p>0x0 /2</p> <p>0x1 /4</p> <p>0x2 /8</p> <p>0x3 /16</p> <p>0x4 /32</p> <p>0x5 /64</p> <p>0x6 /64</p> <p>0x7 /64 (default)</p>
16:14	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	PWRDN	R/W	1	<p>PLL Power Down</p> <p>Value Description</p> <p>1 The PLL is powered down. Care must be taken to ensure that another clock source is functioning and that the <code>BYPASS</code> bit is set before setting this bit.</p> <p>0 The PLL is operating normally.</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>Value Description</p> <p>1 The system clock is derived from the OSC source.</p> <p>0 The system clock 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>

Bit/Field	Name	Type	Reset	Description
10:6	XTAL	R/W	0x0B	Crystal Value

This field specifies the crystal value attached to the main oscillator. The encoding for this field is provided below. Depending on the crystal used, the PLL frequency may not be exactly 400 MHz. See Table 26-10 on page 947 for more information.

Frequencies that may be used with the USB interface are indicated in the table. To function within the clocking requirements of the USB specification, a crystal of 4, 5, 6, 8, 10, 12, or 16 MHz must be used.

Value	Crystal Frequency (MHz) Not Using the PLL	Crystal Frequency (MHz) Using the PLL
0x00	1.000	reserved
0x01	1.8432	reserved
0x02	2.000	reserved
0x03	2.4576	reserved
0x04		3.579545 MHz
0x05		3.6864 MHz
0x06		4 MHz (USB)
0x07		4.096 MHz
0x08		4.9152 MHz
0x09		5 MHz (USB)
0x0A		5.12 MHz
0x0B		6 MHz (reset value)(USB)
0x0C		6.144 MHz
0x0D		7.3728 MHz
0x0E		8 MHz (USB)
0x0F		8.192 MHz
0x10		10.0 MHz (USB)
0x11		12.0 MHz (USB)
0x12		12.288 MHz
0x13		13.56 MHz
0x14		14.31818 MHz
0x15		16.0 MHz (USB)
0x16		16.384 MHz

Bit/Field	Name	Type	Reset	Description										
5:4	OSCSRC	R/W	0x1	<p>Oscillator Source</p> <p>Selects the input source for the OSC. The values are:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Input Source</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>MOSC Main oscillator</td> </tr> <tr> <td>0x1</td> <td>PIOSC Precision internal oscillator (default)</td> </tr> <tr> <td>0x2</td> <td>PIOSC/4 Precision internal oscillator / 4</td> </tr> <tr> <td>0x3</td> <td>30 kHz 30-kHz internal oscillator</td> </tr> </tbody> </table> <p>For additional oscillator sources, see the <b>RCC2</b> register.</p>	Value	Input Source	0x0	MOSC Main oscillator	0x1	PIOSC Precision internal oscillator (default)	0x2	PIOSC/4 Precision internal oscillator / 4	0x3	30 kHz 30-kHz internal oscillator
Value	Input Source													
0x0	MOSC Main oscillator													
0x1	PIOSC Precision internal oscillator (default)													
0x2	PIOSC/4 Precision internal oscillator / 4													
0x3	30 kHz 30-kHz internal oscillator													
3:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
1	IOSCDIS	R/W	0	<p>Precision Internal Oscillator Disable</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>The precision internal oscillator (PIOSC) is disabled.</td> </tr> <tr> <td>0</td> <td>The precision internal oscillator is enabled.</td> </tr> </tbody> </table>	Value	Description	1	The precision internal oscillator (PIOSC) is disabled.	0	The precision internal oscillator is enabled.				
Value	Description													
1	The precision internal oscillator (PIOSC) is disabled.													
0	The precision internal oscillator is enabled.													
0	MOSCDIS	R/W	1	<p>Main Oscillator Disable</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>The main oscillator is disabled (default).</td> </tr> <tr> <td>0</td> <td>The main oscillator is enabled.</td> </tr> </tbody> </table>	Value	Description	1	The main oscillator is disabled (default).	0	The main oscillator is enabled.				
Value	Description													
1	The main oscillator is disabled (default).													
0	The main oscillator is enabled.													

### Register 8: 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 109).

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

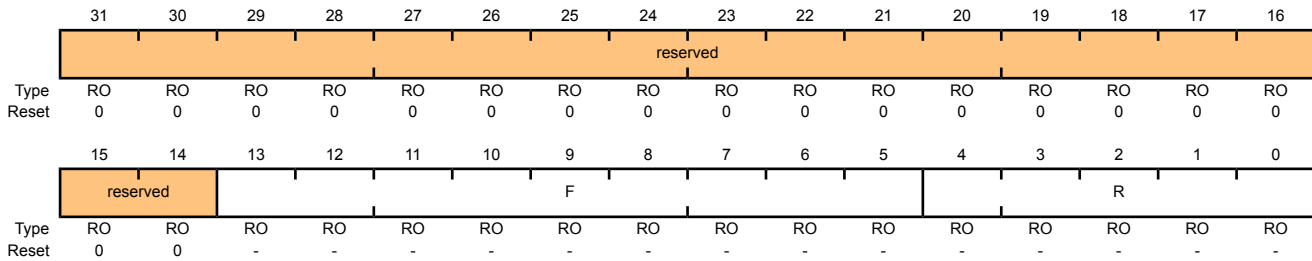
$$PLLFreq = OSCFreq * F / (R + 1)$$

#### XTAL to PLL Translation (PLLCFG)

Base 0x400F.E000

Offset 0x064

Type RO, reset -



Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13:5	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 9: GPIO Host-Bus Control (GPIOHBCTL), offset 0x06C

This register controls which internal bus is used to access each GPIO port. When a bit is clear, the corresponding GPIO port is accessed across the legacy Advanced Peripheral Bus (APB) bus and through the APB memory aperture. When a bit is set, the corresponding port is accessed across the Advanced Host Bus (AHB) bus and through the AHB memory aperture. Each GPIO port can be individually configured to use AHB or APB, but may be accessed only through one aperture. The AHB bus provides better back-to-back access performance than the APB bus. The address aperture in the memory map changes for the ports that are enabled for AHB access (see Table 9-5 on page 292).

#### GPIO Host-Bus Control (GPIOHBCTL)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved							PORTJ	PORTH	PORTG	PORTF	PORTE	PORTD	PORTC	PORTB	PORTA
Type	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	PORTJ	R/W	0	Port J Advanced Host Bus  This bit defines the memory aperture for Port J.  Value Description 1 Advanced Host Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.
7	PORTH	R/W	0	Port H Advanced Host Bus  This bit defines the memory aperture for Port H.  Value Description 1 Advanced Host Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.
6	PORTG	R/W	0	Port G Advanced Host Bus  This bit defines the memory aperture for Port G.  Value Description 1 Advanced Host Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.

Bit/Field	Name	Type	Reset	Description
5	PORTF	R/W	0	Port F Advanced Host Bus This bit defines the memory aperture for Port F.  Value Description 1 Advanced Host Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.
4	PORTE	R/W	0	Port E Advanced Host Bus This bit defines the memory aperture for Port E.  Value Description 1 Advanced Host Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.
3	PORTD	R/W	0	Port D Advanced Host Bus This bit defines the memory aperture for Port D.  Value Description 1 Advanced Host Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.
2	PORTC	R/W	0	Port C Advanced Host Bus This bit defines the memory aperture for Port C.  Value Description 1 Advanced Host Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.
1	PORTB	R/W	0	Port B Advanced Host Bus This bit defines the memory aperture for Port B.  Value Description 1 Advanced Host Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.
0	PORTA	R/W	0	Port A Advanced Host Bus This bit defines the memory aperture for Port A.  Value Description 1 Advanced Host Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.





Bit/Field	Name	Type	Reset	Description
30	USEFRACT	R/W	0	<p>Use FRACT</p> <p>The FRACT bit adds an additional bit as the LSB to the SYSDIV2 field allowing additional frequency choices.</p> <p>Value Description</p> <p>1 The FRACT bit can be set or cleared by the software.</p> <p>0 The FRACT bit is forced to be set.</p>
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	<p>System Clock Divisor</p> <p>Specifies which divisor is used to generate the system clock from the PLL output.</p> <p>Although the PLL VCO frequency is 400 MHz, it is predivided by 2 before the divisor is applied.</p> <p>This field is wider than the <b>RCC</b> register SYSDIV field in order to provide additional divisor values. These additional values permit the system clock to be run at much lower frequencies during Deep Sleep mode. For example, where the <b>RCC</b> register SYSDIV encoding of 1111 provides /16, the <b>RCC2</b> register SYSDIV2 encoding of 111111 provides /64.</p>
22	FRACT	R/W	0	<p>Fractional Divider</p> <p>The FRACT bit adds an additional bit as the LSB to the SYSDIV2 field allowing additional frequency choices.</p> <p>This bit can only be set or cleared when USEFRACT is set.</p>
21:15	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	USBPWRDN	R/W	1	<p>Power-Down USB PLL</p> <p>Value Description</p> <p>1 The USB PLL is powered down.</p> <p>0 The USB PLL operates normally.</p>
13	PWRDN2	R/W	1	<p>Power-Down PLL</p> <p>Value Description</p> <p>1 The PLL is powered down.</p> <p>0 The PLL operates normally.</p>
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	BYPASS2	R/W	1	<p>PLL Bypass</p> <p>Value Description</p> <p>1 The system clock is derived from the OSC source.</p> <p>0 The system clock 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>
10:7	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:4	OSCSRC2	R/W	0x1	<p>Oscillator Source</p> <p>Selects the input source for the OSC. The values are:</p> <p>Value Description</p> <p>0x0 MOSC Main oscillator</p> <p>0x1 PIOSC Precision internal oscillator</p> <p>0x2 PIOSC/4 Precision internal oscillator / 4</p> <p>0x3 30 kHz 30-kHz internal oscillator</p> <p>0x4-0x7 Reserved</p>
3:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

### Register 11: Main Oscillator Control (MOSCCTL), offset 0x07C

This register provides the ability to enable the MOSC clock verification circuit. When enabled, this circuit monitors the frequency of the MOSC to verify that the oscillator is operating within specified limits. If the clock goes invalid after being enabled, the microcontroller issues a power-on reset and reboots to the NMI handler.

#### Main Oscillator Control (MOSCCTL)

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

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	CVAL	R/W	0	Clock Validation for MOSC
				Value Description
				1 The MOSC monitor circuit is enabled.
				0 The MOSC monitor circuit is disabled.

## Register 12: Deep Sleep Clock Configuration (DSLPCCLKCFG), offset 0x144

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

### Deep Sleep Clock Configuration (DSLPCCLKCFG)

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

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

Bit/Field	Name	Type	Reset	Description																																		
31:29	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.																																		
28:23	DSDIVORIDE	R/W	0x0F	<p>Divider Field Override</p> <p>If Deep-Sleep mode is enabled when the PLL is running, the PLL is disabled. This 6-bit field contains a system divider field that overrides the <b>SYSDIV</b> field in the <b>RCC</b> register or the <b>SYSDIV2</b> field in the <b>RCC2</b> register during Deep Sleep. This divider is applied to the source selected by the <b>DSOSCSRC</b> field.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr><td>0x0</td><td>reserved</td></tr> <tr><td>0x1</td><td>/2</td></tr> <tr><td>0x2</td><td>/3</td></tr> <tr><td>0x3</td><td>/4</td></tr> <tr><td>0x4</td><td>/5</td></tr> <tr><td>0x5</td><td>/6</td></tr> <tr><td>0x6</td><td>/7</td></tr> <tr><td>0x7</td><td>/8</td></tr> <tr><td>0x8</td><td>/9</td></tr> <tr><td>0x9</td><td>/10</td></tr> <tr><td>0xA</td><td>/11</td></tr> <tr><td>0xB</td><td>/12</td></tr> <tr><td>0xC</td><td>/13</td></tr> <tr><td>0xD</td><td>/14</td></tr> <tr><td>0xE</td><td>/15</td></tr> <tr><td>0xF</td><td>/16</td></tr> </tbody> </table>	Value	Description	0x0	reserved	0x1	/2	0x2	/3	0x3	/4	0x4	/5	0x5	/6	0x6	/7	0x7	/8	0x8	/9	0x9	/10	0xA	/11	0xB	/12	0xC	/13	0xD	/14	0xE	/15	0xF	/16
Value	Description																																					
0x0	reserved																																					
0x1	/2																																					
0x2	/3																																					
0x3	/4																																					
0x4	/5																																					
0x5	/6																																					
0x6	/7																																					
0x7	/8																																					
0x8	/9																																					
0x9	/10																																					
0xA	/11																																					
0xB	/12																																					
0xC	/13																																					
0xD	/14																																					
0xE	/15																																					
0xF	/16																																					
22:7	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.																																		

Bit/Field	Name	Type	Reset	Description																				
6:4	DSOSCSRC	R/W	0x0	<p>Clock Source</p> <p>Specifies the clock source during Deep-Sleep mode.</p> <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0x0</td><td>MOSC</td></tr><tr><td></td><td>Use the main oscillator as the source.</td></tr><tr><td></td><td><b>Note:</b> If the PIOSC is being used as the clock reference for the PLL, the PIOSC is the clock source instead of MOSC in Deep-Sleep mode.</td></tr><tr><td>0x1</td><td>PIOSC</td></tr><tr><td></td><td>Use the precision internal 16-MHz oscillator as the source.</td></tr><tr><td>0x2</td><td>Reserved</td></tr><tr><td>0x3</td><td>30 kHz</td></tr><tr><td></td><td>Use the 30-kHz internal oscillator as the source.</td></tr><tr><td>0x4-0x7</td><td>Reserved</td></tr></tbody></table>	Value	Description	0x0	MOSC		Use the main oscillator as the source.		<b>Note:</b> If the PIOSC is being used as the clock reference for the PLL, the PIOSC is the clock source instead of MOSC in Deep-Sleep mode.	0x1	PIOSC		Use the precision internal 16-MHz oscillator as the source.	0x2	Reserved	0x3	30 kHz		Use the 30-kHz internal oscillator as the source.	0x4-0x7	Reserved
Value	Description																							
0x0	MOSC																							
	Use the main oscillator as the source.																							
	<b>Note:</b> If the PIOSC is being used as the clock reference for the PLL, the PIOSC is the clock source instead of MOSC in Deep-Sleep mode.																							
0x1	PIOSC																							
	Use the precision internal 16-MHz oscillator as the source.																							
0x2	Reserved																							
0x3	30 kHz																							
	Use the 30-kHz internal oscillator as the source.																							
0x4-0x7	Reserved																							
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 13: Deep Sleep Flash Configuration (DSFLASHCFG), offset 0x14C

This register allows the user to force the shutdown of the Flash subsystem during all Deep-Sleep periods. For deep-sleep periods that do not require a MOSC startup time or a PLL lock time, the microcontroller has a lockout period of 30-120  $\mu$ s for the Flash to start up after the event to exit deep sleep has occurred.

#### Deep Sleep Flash Configuration (DSFLASHCFG)

Base 0x400F.E000

Offset 0x14C

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															SHDWN
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	SHDWN	R/W	0	Flash Shutdown

#### Value Description

- |   |   |
|---|---|
| 1 | The Flash subsystem is shutdown during all deep-sleep operations. |
| 0 | The Flash subsystem is powered up during deep-sleep operations    |

## Register 14: Precision Internal Oscillator Calibration (PIOSCCAL), offset 0x150

This register provides the ability to update or recalibrate the precision internal oscillator.

### Precision Internal Oscillator Calibration (PIOSCCAL)

Base 0x400F.E000  
 Offset 0x150  
 Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31	UTEN	R/W	0	Use User Trim Value  Value Description 1 The trim value in bits[6:0] of this register are used for any update trim operation. 0 The factory calibration value is used for an update trim operation.
30:9	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.
8	UPDATE	R/W	0	Update Trim  Value Description 1 Updates the PIOSC trim value with the UT bit or the DT bit in the <b>PIOSCSTAT</b> register. Used with UTEN. 0 No action.  This bit is auto-cleared after the update.
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.
6:0	UT	R/W	0x0	User Trim Value  User trim value that can be loaded into the PIOSC.  Refer to “Main PLL Frequency Configuration” on page 93 for more information on calibrating the PIOSC.



## Register 15: I<sup>2</sup>S MCLK Configuration (I2SMCLKCFG), offset 0x170

This register configures the receive and transmit fractional clock dividers for the for the I<sup>2</sup>S master transmit and receive clocks (I2S0TXMCLK and I2S0RXMCLK) . Varying the integer and fractional inputs for the clocks allows greater accuracy in hitting the target I<sup>2</sup>S clock frequencies. Refer to “Clock Control” on page 642 for combinations of the TXI and TXF bits and the RXI and RXF bits that provide MCLK frequencies within acceptable error limits.

### I2S MCLK Configuration (I2SMCLKCFG)

Base 0x400F.E000  
Offset 0x170  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RXEN	reserved			RXI								RXF			
Type	R/W	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
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TXEN	reserved			TXI								TXF			
Type	R/W	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	RXEN	R/W	0	RX Clock Enable  <b>Value Description</b> 1 The I <sup>2</sup> S receive clock generator is enabled. 0 The I <sup>2</sup> S receive clock generator is disabled.  If the RXSLV bit in the <b>I<sup>2</sup>S Module Configuration (I2SCFG)</b> register is set, then the I2S0RXMCLK must be externally generated.
30:28	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.
27:20	RXI	R/W	0x0	RX Clock Integer Input  This field contains the integer input for the receive clock generator.
19:16	RXF	R/W	0x0	RX Clock Fractional Input  This field contains the fractional input for the receive clock generator.
15	TXEN	R/W	0	TX Clock Enable  <b>Value Description</b> 1 The I <sup>2</sup> S transmit clock generator is enabled. 0 The I <sup>2</sup> S transmit clock generator is disabled.  If the TXSLV bit in the <b>I<sup>2</sup>S Module Configuration (I2SCFG)</b> register is set, then the I2S0TXMCLK must be externally generated.

Bit/Field	Name	Type	Reset	Description
14:12	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.
11:4	TXI	R/W	0x00	TX Clock Integer Input This field contains the integer input for the transmit clock generator.
3:0	TXF	R/W	0x0	TX Clock Fractional Input This field contains the fractional input for the transmit clock generator.

## Register 16: Device Identification 1 (DID1), offset 0x004

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

### Device Identification 1 (DID1)

Base 0x400F.E000

Offset 0x004

Type RO, reset -

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

Bit/Field	Name	Type	Reset	Description				
31:28	VER	RO	0x1	<p>DID1 Version</p> <p>This field defines the <b>DID1</b> register format version. The version number is numeric. The value of the <code>VER</code> field is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x1</td> <td>Second version of the <b>DID1</b> register format.</td> </tr> </tbody> </table>	Value	Description	0x1	Second version of the <b>DID1</b> register format.
Value	Description							
0x1	Second version of the <b>DID1</b> register format.							
27:24	FAM	RO	0x0	<p>Family</p> <p>This field provides the family identification of the device within the Luminary Micro product portfolio. The value is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Stellaris family of microcontrollers, that is, all devices with external part numbers starting with LM3S.</td> </tr> </tbody> </table>	Value	Description	0x0	Stellaris family of microcontrollers, that is, all devices with external part numbers starting with LM3S.
Value	Description							
0x0	Stellaris family of microcontrollers, that is, all devices with external part numbers starting with LM3S.							
23:16	PARTNO	RO	0x69	<p>Part Number</p> <p>This field provides the part number of the device within the family. The value is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x69</td> <td>LM3S5791</td> </tr> </tbody> </table>	Value	Description	0x69	LM3S5791
Value	Description							
0x69	LM3S5791							
15:13	PINCOUNT	RO	0x2	<p>Package Pin Count</p> <p>This field specifies the number of pins on the device package. The value is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x2</td> <td>100-pin package</td> </tr> </tbody> </table>	Value	Description	0x2	100-pin package
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	-	Temperature Range  This field specifies the temperature rating of the device. The value is encoded as follows (all other encodings are reserved):  Value Description 0x0 Commercial temperature range (0°C to 70°C) 0x1 Industrial temperature range (-40°C to 85°C) 0x2 Extended temperature range (-40°C to 105°C)
4:3	PKG	RO	-	Package Type  This field specifies the package type. The value is encoded as follows (all other encodings are reserved):  Value Description 0x0 SOIC package 0x1 LQFP package 0x2 BGA package
2	ROHS	RO	1	RoHS-Compliance  This bit specifies whether the device is RoHS-compliant. A 1 indicates the part is RoHS-compliant.
1:0	QUAL	RO	-	Qualification Status  This field specifies the qualification status of the device. The value is encoded as follows (all other encodings are reserved):  Value Description 0x0 Engineering Sample (unqualified) 0x1 Pilot Production (unqualified) 0x2 Fully Qualified

## Register 17: 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.003F

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

Bit/Field	Name	Type	Reset	Description
31:16	SRAMSZ	RO	0x00FF	SRAM Size Indicates the size of the on-chip SRAM memory.  Value   Description 0x00FF 64 KB of SRAM
15:0	FLASHSZ	RO	0x003F	Flash Size Indicates the size of the on-chip flash memory.  Value   Description 0x003F 128 KB of Flash

### Register 18: Device Capabilities 1 (DC1), offset 0x010

This register is predefined by the part and can be used to verify features. If any bit is clear in this register, the module is not present. The corresponding bit in the RCGC0, SCGC0, and DCGC0 registers cannot be set.

#### Device Capabilities 1 (DC1)

Base 0x400F.E000

Offset 0x010

Type RO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			WDT1	reserved		CAN1	CAN0	reserved			PWM	reserved		ADC1	ADC0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	1	0	0	1	1	0	0	0	1	0	0	1	1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MINSYSDIV				MAXADC1SPD		MAXADC0SPD		MPU	reserved	TEMPSNS	PLL	WDT0	SWO	SWD	JTAG
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	-	-	-	-	1	1	1	1	1	0	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:29	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	WDT1	RO	1	Watchdog Timer1 Present When set, indicates that watchdog timer 1 is present.
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	CAN1	RO	1	CAN Module 1 Present When set, indicates that CAN unit 1 is present.
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.
19:18	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.
17	ADC1	RO	1	ADC Module 1 Present When set, indicates that ADC module 1 is present.
16	ADC0	RO	1	ADC Module 0 Present When set, indicates that ADC module 0 is present

Bit/Field	Name	Type	Reset	Description
15:12	MINSYSDIV	RO	-	<p>System Clock Divider</p> <p>Minimum 4-bit divider value for system clock. The reset value is hardware-dependent. See the RCC register for how to change the system clock divisor using the SYSDIV bit.</p> <p>Value Description</p> <p>0x1 Divide VCO (400MHZ) by 5 minimum</p> <p>0x2 Divide VCO (400MHZ) by <math>2^2 + 2 = 6</math> minimum</p> <p>0x3 Specifies a 50-MHz CPU clock with a PLL divider of 4.</p> <p>0x7 Specifies a 25-MHz clock with a PLL divider of 8.</p> <p>0x9 Specifies a 20-MHz clock with a PLL divider of 10.</p>
11:10	MAXADC1SPD	RO	0x3	<p>Max ADC1 Speed</p> <p>This field indicates the maximum rate at which the ADC samples data.</p> <p>Value Description</p> <p>0x3 1M samples/second</p>
9:8	MAXADC0SPD	RO	0x3	<p>Max ADC0 Speed</p> <p>This field indicates the maximum rate at which the ADC samples data.</p> <p>Value Description</p> <p>0x3 1M samples/second</p>
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	reserved	RO	0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>
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	WDT0	RO	1	<p>Watchdog Timer 0 Present</p> <p>When set, indicates that watchdog timer 0 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>

Bit/Field	Name	Type	Reset	Description
0	JTAG	RO	1	JTAG Present When set, indicates that the JTAG debugger interface is present.



## Register 19: Device Capabilities 2 (DC2), offset 0x014

This register is predefined by the part and can be used to verify features. If any bit is clear in this register, the module is not present. The corresponding bit in the RCGC0, SCGC0, and DCGC0 registers cannot be set.

### Device Capabilities 2 (DC2)

Base 0x400F.E000

Offset 0x014

Type RO, reset 0x570F.5337

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved	EPI0	reserved	I2S0	reserved	COMP2	COMP1	COMP0	reserved			TIMER3	TIMER2	TIMER1	TIMER0	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	1	0	1	0	1	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
	reserved	I2C1	reserved	I2C0	reserved		QE1	QE0	reserved		SSI1	SSI0	reserved	UART2	UART1	UART0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	1	0	1	0	0	1	1	0	0	1	1	0	1	1	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	EPI0	RO	1	EPI Module 0 Present When set, indicates that EPI 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	I2S0	RO	1	I2S Module 0 Present When set, indicates that I2S module 0 is present.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	COMP2	RO	1	Analog Comparator 2 Present When set, indicates that analog comparator 2 is present.
25	COMP1	RO	1	Analog Comparator 1 Present When set, indicates that analog comparator 1 is present.
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 Module 3 Present When set, indicates that General-Purpose Timer module 3 is present.

Bit/Field	Name	Type	Reset	Description
18	TIMER2	RO	1	Timer Module 2 Present When set, indicates that General-Purpose Timer module 2 is present.
17	TIMER1	RO	1	Timer Module 1 Present When set, indicates that General-Purpose Timer module 1 is present.
16	TIMER0	RO	1	Timer Module 0 Present When set, indicates that General-Purpose Timer module 0 is present.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	RO	1	I2C Module 1 Present When set, indicates that I2C module 1 is present.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	RO	1	I2C Module 0 Present When set, indicates that I2C module 0 is present.
11:10	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	QE11	RO	1	QE1 Module 1 Present When set, indicates that QE1 module 1 is present.
8	QE10	RO	1	QE1 Module 0 Present When set, indicates that QE1 module 0 is present.
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	SSI1	RO	1	SSI Module 1 Present When set, indicates that SSI module 1 is present.
4	SSI0	RO	1	SSI Module 0 Present When set, indicates that SSI module 0 is present.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	RO	1	UART Module 2 Present When set, indicates that UART module 2 is present.
1	UART1	RO	1	UART Module 1 Present When set, indicates that UART module 1 is present.

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

## Register 20: Device Capabilities 3 (DC3), offset 0x018

This register is predefined by the part and can be used to verify features. If any bit is clear in this register, the module is not present. The corresponding bit in the RCGC0, SCGC0, and DCGC0 registers cannot be set.

### Device Capabilities 3 (DC3)

Base 0x400F.E000  
 Offset 0x018  
 Type RO, reset 0xBFFF.B6FF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	32KHZ	reserved	CCP5	CCP4	CCP3	CCP2	CCP1	CCP0	ADC0AIN7	ADC0AIN6	ADC0AIN5	ADC0AIN4	ADC0AIN3	ADC0AIN2	ADC0AIN1	ADC0AIN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PWMFAULT	reserved	C2PLUS	C2MINUS	reserved	C1PLUS	C1MINUS	reserved	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	1	1	0	1	1	0	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31	32KHZ	RO	1	32KHz Input Clock Available When set, indicates an even CCP pin is present and can be used as a 32-KHz input clock.
30	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
29	CCP5	RO	1	CCP5 Pin Present When set, indicates that Capture/Compare/PWM pin 5 is present.
28	CCP4	RO	1	CCP4 Pin Present When set, indicates that Capture/Compare/PWM pin 4 is present.
27	CCP3	RO	1	CCP3 Pin Present When set, indicates that Capture/Compare/PWM pin 3 is present.
26	CCP2	RO	1	CCP2 Pin Present When set, indicates that Capture/Compare/PWM pin 2 is present.
25	CCP1	RO	1	CCP1 Pin Present When set, indicates that Capture/Compare/PWM pin 1 is present.
24	CCP0	RO	1	CCP0 Pin Present When set, indicates that Capture/Compare/PWM pin 0 is present.
23	ADC0AIN7	RO	1	ADC Module 0 AIN7 Pin Present When set, indicates that ADC module 0 input pin 7 is present.
22	ADC0AIN6	RO	1	ADC Module 0 AIN6 Pin Present When set, indicates that ADC module 0 input pin 6 is present.

Bit/Field	Name	Type	Reset	Description
21	ADC0AIN5	RO	1	ADC Module 0 AIN5 Pin Present When set, indicates that ADC module 0 input pin 5 is present.
20	ADC0AIN4	RO	1	ADC Module 0 AIN4 Pin Present When set, indicates that ADC module 0 input pin 4 is present.
19	ADC0AIN3	RO	1	ADC Module 0 AIN3 Pin Present When set, indicates that ADC module 0 input pin 3 is present.
18	ADC0AIN2	RO	1	ADC Module 0 AIN2 Pin Present When set, indicates that ADC module 0 input pin 2 is present.
17	ADC0AIN1	RO	1	ADC Module 0 AIN1 Pin Present When set, indicates that ADC module 0 input pin 1 is present.
16	ADC0AIN0	RO	1	ADC Module 0 AIN0 Pin Present When set, indicates that ADC module 0 input pin 0 is present.
15	PWMFAULT	RO	1	PWM Fault Pin Present When set, indicates that a PWM Fault pin is present. See DC5 for specific Fault pins on this device.
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	C2PLUS	RO	1	C2+ Pin Present When set, indicates that the analog comparator 2 (+) input pin is present.
12	C2MINUS	RO	1	C2- Pin Present When set, indicates that the analog comparator 2 (-) input pin is present.
11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	C1PLUS	RO	1	C1+ Pin Present When set, indicates that the analog comparator 1 (+) input pin is present.
9	C1MINUS	RO	1	C1- Pin Present When set, indicates that the analog comparator 1 (-) input pin is present.
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	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.

Bit/Field	Name	Type	Reset	Description
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 21: Device Capabilities 4 (DC4), offset 0x01C

This register is predefined by the part and can be used to verify features. If any bit is clear in this register, the module is not present. The corresponding bit in the RCGC0, SCGC0, and DCGC0 registers cannot be set.

### Device Capabilities 4 (DC4)

Base 0x400F.E000

Offset 0x01C

Type RO, reset 0x0000.F1FF

	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
	CCP7	CCP6	UDMA	ROM	reserved			GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1

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	CCP7	RO	1	CCP7 Pin Present When set, indicates that Capture/Compare/PWM pin 7 is present.
14	CCP6	RO	1	CCP6 Pin Present When set, indicates that Capture/Compare/PWM pin 6 is present.
13	UDMA	RO	1	Micro-DMA Module Present When set, indicates that the micro-DMA module present.
12	ROM	RO	1	Internal Code ROM Present When set, indicates that internal code ROM is present.
11:9	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	GPIOJ	RO	1	GPIO Port J Present When set, indicates that GPIO Port J is present.
7	GPIOH	RO	1	GPIO Port H Present When set, indicates that GPIO Port H is present.
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.

Bit/Field	Name	Type	Reset	Description
4	GPIOE	RO	1	GPIO Port E Present When set, indicates that GPIO Port E is present.
3	GPIOD	RO	1	GPIO Port D Present When set, indicates that GPIO Port D is present.
2	GPIOC	RO	1	GPIO Port C Present When set, indicates that GPIO Port C is present.
1	GPIOB	RO	1	GPIO Port B Present When set, indicates that GPIO Port B is present.
0	GPIOA	RO	1	GPIO Port A Present When set, indicates that GPIO Port A is present.



## Register 22: Device Capabilities 5 (DC5), offset 0x020

This register is predefined by the part and can be used to verify features. If any bit is clear in this register, the module is not present. The corresponding bit in the RCGC0, SCGC0, and DCGC0 registers cannot be set.

### Device Capabilities 5 (DC5)

Base 0x400F.E000

Offset 0x020

Type RO, reset 0x0F30.00FF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved				PWMFAULT3	PWMFAULT2	PWMFAULT1	PWMFAULT0	reserved		PWMEFLT	PWMSYNC	reserved			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	1	1	1	1	0	0	1	1	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PWM7	PWM6	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:28	reserved	RO	0	Software should not rely on the value of 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	PWMFAULT3	RO	1	PWM Fault 3 Pin Present When set, indicates that the PWM Fault 3 pin is present.
26	PWMFAULT2	RO	1	PWM Fault 2 Pin Present When set, indicates that the PWM Fault 2 pin is present.
25	PWMFAULT1	RO	1	PWM Fault 1 Pin Present When set, indicates that the PWM Fault 1 pin is present.
24	PWMFAULT0	RO	1	PWM Fault 0 Pin Present When set, indicates that the PWM Fault 0 pin is present.
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	PWMEFLT	RO	1	PWM Extended Fault Active When set, indicates that the PWM Extended Fault feature is active.
20	PWMSYNC	RO	1	PWM Extended SYNC Active When set, indicates that the PWM Extended SYNC feature is active.
19:8	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	PWM7	RO	1	PWM7 Pin Present When set, indicates that the PWM pin 7 is present.

Bit/Field	Name	Type	Reset	Description
6	PWM6	RO	1	PWM6 Pin Present When set, indicates that the PWM pin 6 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 23: Device Capabilities 6 (DC6), offset 0x024

This register is predefined by the part and can be used to verify features. If any bit is clear in this register, the module is not present. The corresponding bit in the RCGC0, SCGC0, and DCGC0 registers cannot be set.

### Device Capabilities 6 (DC6)

Base 0x400F.E000

Offset 0x024

Type RO, reset 0x0000.0013

	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												USB0PHY	reserved		USB0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	USB0PHY	RO	1	USB Module 0 PHY Present  When set, indicates that the USB module 0 PHY 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.
1:0	USB0	RO	0x3	USB Module 0 Present  This field indicates that USB module 0 is present and specifies its capability.  Value Description 0x3 USB0 is OTG.

## Register 24: Device Capabilities 7 (DC7), offset 0x028

This register is predefined by the part and can be used to verify uDMA channel features. A 1 indicates the channel is available on this device; a 0 that the channel is only available on other devices in the family. Most channels have primary and alternate assignments. If the primary function is not available on this microcontroller, the alternate function becomes the primary function. If the alternate function is not available, the primary function is the only option.

### Device Capabilities 7 (DC7)

Base 0x400F.E000  
Offset 0x028  
Type RO, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved-31	DMACH30	DMACH29	DMACH28	DMACH27	DMACH26	DMACH25	DMACH24	DMACH23	DMACH22	DMACH21	DMACH20	DMACH19	DMACH18	DMACH17	DMACH16
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DMACH15	DMACH14	DMACH13	DMACH12	DMACH11	DMACH10	DMACH9	DMACH8	DMACH7	DMACH6	DMACH5	DMACH4	DMACH3	DMACH2	DMACH1	DMACH0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31	reserved-31	RO	1	Reserved  Reserved for uDMA channel 31.
30	DMACH30	RO	1	SW  When set, indicates uDMA channel 30 is available for software transfers.
29	DMACH29	RO	1	I2S0_TX / CAN1_TX  When set, indicates uDMA channel 29 is available and connected to the transmit path of I2S module 0. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of CAN module 1 transmit.
28	DMACH28	RO	1	I2S0_RX / CAN1_RX  When set, indicates uDMA channel 28 is available and connected to the receive path of I2S module 0. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of CAN module 1 receive.
27	DMACH27	RO	1	CAN1_TX / ADC1_SS3  When set, indicates uDMA channel 27 is available and connected to the transmit path of CAN module 1. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of ADC module 1 Sample Sequencer 3.
26	DMACH26	RO	1	CAN1_RX / ADC1_SS2  When set, indicates uDMA channel 26 is available and connected to the receive path of CAN module 1. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of ADC module 1 Sample Sequencer 2.

Bit/Field	Name	Type	Reset	Description
25	DMACH25	RO	1	SSI1_TX / ADC1_SS1 When set, indicates uDMA channel 25 is available and connected to the transmit path of SSI module 1. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of ADC module 1 Sample Sequencer 1.
24	DMACH24	RO	1	SSI1_RX / ADC1_SS0 When set, indicates uDMA channel 24 is available and connected to the receive path of SSI module 1. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of ADC module 1 Sample Sequencer 0.
23	DMACH23	RO	1	UART1_TX / CAN2_TX When set, indicates uDMA channel 23 is available and connected to the transmit path of UART module 1. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of CAN module 2 transmit.
22	DMACH22	RO	1	UART1_RX / CAN2_RX When set, indicates uDMA channel 22 is available and connected to the receive path of UART module 1. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of CAN module 2 receive.
21	DMACH21	RO	1	Timer1B / EPI0_TX When set, indicates uDMA channel 21 is available and connected to Timer 1B. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of EPI module 0 transmit.
20	DMACH20	RO	1	Timer1A / EPI0_RX When set, indicates uDMA channel 20 is available and connected to Timer 1A. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of EPI module 0 receive.
19	DMACH19	RO	1	Timer0B / Timer1B When set, indicates uDMA channel 19 is available and connected to Timer 0B. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of Timer 1B.
18	DMACH18	RO	1	Timer0A / Timer1A When set, indicates uDMA channel 18 is available and connected to Timer 0A. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of Timer 1A.
17	DMACH17	RO	1	ADC0_SS3 When set, indicates uDMA channel 17 is available and connected to ADC module 0 Sample Sequencer 3.

Bit/Field	Name	Type	Reset	Description
16	DMACH16	RO	1	ADC0_SS2 When set, indicates uDMA channel 16 is available and connected to ADC module 0 Sample Sequencer 2.
15	DMACH15	RO	1	ADC0_SS1 / Timer2B When set, indicates uDMA channel 15 is available and connected to ADC module 0 Sample Sequencer 1. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of Timer 2B.
14	DMACH14	RO	1	ADC0_SS0 / Timer2A When set, indicates uDMA channel 14 is available and connected to ADC module 0 Sample Sequencer 0. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of Timer 2A.
13	DMACH13	RO	1	CAN0_TX / UART2_TX When set, indicates uDMA channel 13 is available and connected to the transmit path of CAN module 0. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of UART module 2 transmit.
12	DMACH12	RO	1	CAN0_RX / UART2_RX When set, indicates uDMA channel 12 is available and connected to the receive path of CAN module 0. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of UART module 2 receive.
11	DMACH11	RO	1	SSI0_TX / UART1_TX When set, indicates uDMA channel 11 is available and connected to the transmit path of SSI module 0. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of UART module 1 transmit.
10	DMACH10	RO	1	SSI0_RX / UART1_RX When set, indicates uDMA channel 10 is available and connected to the receive path of SSI module 0. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of UART module 1 receive.
9	DMACH9	RO	1	UART0_TX / SSI1_TX When set, indicates uDMA channel 9 is available and connected to the transmit path of UART module 0. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of SSI module 1 transmit.
8	DMACH8	RO	1	UART0_RX / SSI1_RX When set, indicates uDMA channel 8 is available and connected to the receive path of UART module 0. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of SSI module 1 receive.

Bit/Field	Name	Type	Reset	Description
7	DMACH7	RO	1	ETH_TX / Timer2B  When set, indicates uDMA channel 7 is available and connected to the transmit path of the Ethernet module. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of Timer 2B.
6	DMACH6	RO	1	ETH_RX / Timer2A  When set, indicates uDMA channel 6 is available and connected to the receive path of the Ethernet module. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of Timer 2A.
5	DMACH5	RO	1	USB_EP3_TX / Timer2B  When set, indicates uDMA channel 5 is available and connected to the transmit path of USB endpoint 3. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of Timer 2B.
4	DMACH4	RO	1	USB_EP3_RX / Timer2A  When set, indicates uDMA channel 4 is available and connected to the receive path of USB endpoint 3. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of Timer 2A.
3	DMACH3	RO	1	USB_EP2_TX / Timer3B  When set, indicates uDMA channel 3 is available and connected to the transmit path of USB endpoint 2. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of Timer 3B.
2	DMACH2	RO	1	USB_EP2_RX / Timer3A  When set, indicates uDMA channel 2 is available and connected to the receive path of USB endpoint 2. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of Timer 3A.
1	DMACH1	RO	1	USB_EP1_TX / UART2_TX  When set, indicates uDMA channel 1 is available and connected to the transmit path of USB endpoint 1. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of UART module 2 transmit.
0	DMACH0	RO	1	USB_EP1_RX / UART2_RX  When set, indicates uDMA channel 0 is available and connected to the receive path of USB endpoint 1. If the corresponding bit in the <b>DMACHALT</b> register is set, the channel is connected instead to the alternate channel assignment of UART module 2 receive.

## Register 25: Device Capabilities 8 ADC Channels (DC8), offset 0x02C

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

### Device Capabilities 8 ADC Channels (DC8)

Base 0x400F.E000  
 Offset 0x02C  
 Type RO, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ADC1AIN15	ADC1AIN14	ADC1AIN13	ADC1AIN12	ADC1AIN11	ADC1AIN10	ADC1AIN9	ADC1AIN8	ADC1AIN7	ADC1AIN6	ADC1AIN5	ADC1AIN4	ADC1AIN3	ADC1AIN2	ADC1AIN1	ADC1AIN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ADC0AIN15	ADC0AIN14	ADC0AIN13	ADC0AIN12	ADC0AIN11	ADC0AIN10	ADC0AIN9	ADC0AIN8	ADC0AIN7	ADC0AIN6	ADC0AIN5	ADC0AIN4	ADC0AIN3	ADC0AIN2	ADC0AIN1	ADC0AIN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31	ADC1AIN15	RO	1	ADC Module 1 AIN15 Pin Present When set, indicates that ADC module 1 input pin 15 is present.
30	ADC1AIN14	RO	1	ADC Module 1 AIN14 Pin Present When set, indicates that ADC module 1 input pin 14 is present.
29	ADC1AIN13	RO	1	ADC Module 1 AIN13 Pin Present When set, indicates that ADC module 1 input pin 13 is present.
28	ADC1AIN12	RO	1	ADC Module 1 AIN12 Pin Present When set, indicates that ADC module 1 input pin 12 is present.
27	ADC1AIN11	RO	1	ADC Module 1 AIN11 Pin Present When set, indicates that ADC module 1 input pin 11 is present.
26	ADC1AIN10	RO	1	ADC Module 1 AIN10 Pin Present When set, indicates that ADC module 1 input pin 10 is present.
25	ADC1AIN9	RO	1	ADC Module 1 AIN9 Pin Present When set, indicates that ADC module 1 input pin 9 is present.
24	ADC1AIN8	RO	1	ADC Module 1 AIN8 Pin Present When set, indicates that ADC module 1 input pin 8 is present.
23	ADC1AIN7	RO	1	ADC Module 1 AIN7 Pin Present When set, indicates that ADC module 1 input pin 7 is present.
22	ADC1AIN6	RO	1	ADC Module 1 AIN6 Pin Present When set, indicates that ADC module 1 input pin 6 is present.
21	ADC1AIN5	RO	1	ADC Module 1 AIN5 Pin Present When set, indicates that ADC module 1 input pin 5 is present.



Bit/Field	Name	Type	Reset	Description
20	ADC1AIN4	RO	1	ADC Module 1 AIN4 Pin Present When set, indicates that ADC module 1 input pin 4 is present.
19	ADC1AIN3	RO	1	ADC Module 1 AIN3 Pin Present When set, indicates that ADC module 1 input pin 3 is present.
18	ADC1AIN2	RO	1	ADC Module 1 AIN2 Pin Present When set, indicates that ADC module 1 input pin 2 is present.
17	ADC1AIN1	RO	1	ADC Module 1 AIN1 Pin Present When set, indicates that ADC module 1 input pin 1 is present.
16	ADC1AIN0	RO	1	ADC Module 1 AIN0 Pin Present When set, indicates that ADC module 1 input pin 0 is present.
15	ADC0AIN15	RO	1	ADC Module 0 AIN15 Pin Present When set, indicates that ADC module 0 input pin 15 is present.
14	ADC0AIN14	RO	1	ADC Module 0 AIN14 Pin Present When set, indicates that ADC module 0 input pin 14 is present.
13	ADC0AIN13	RO	1	ADC Module 0 AIN13 Pin Present When set, indicates that ADC module 0 input pin 13 is present.
12	ADC0AIN12	RO	1	ADC Module 0 AIN12 Pin Present When set, indicates that ADC module 0 input pin 12 is present.
11	ADC0AIN11	RO	1	ADC Module 0 AIN11 Pin Present When set, indicates that ADC module 0 input pin 11 is present.
10	ADC0AIN10	RO	1	ADC Module 0 AIN10 Pin Present When set, indicates that ADC module 0 input pin 10 is present.
9	ADC0AIN9	RO	1	ADC Module 0 AIN9 Pin Present When set, indicates that ADC module 0 input pin 9 is present.
8	ADC0AIN8	RO	1	ADC Module 0 AIN8 Pin Present When set, indicates that ADC module 0 input pin 8 is present.
7	ADC0AIN7	RO	1	ADC Module 0 AIN7 Pin Present When set, indicates that ADC module 0 input pin 7 is present.
6	ADC0AIN6	RO	1	ADC Module 0 AIN6 Pin Present When set, indicates that ADC module 0 input pin 6 is present.
5	ADC0AIN5	RO	1	ADC Module 0 AIN5 Pin Present When set, indicates that ADC module 0 input pin 5 is present.
4	ADC0AIN4	RO	1	ADC Module 0 AIN4 Pin Present When set, indicates that ADC module 0 input pin 4 is present.

Bit/Field	Name	Type	Reset	Description
3	ADC0AIN3	RO	1	ADC Module 0 AIN3 Pin Present When set, indicates that ADC module 0 input pin 3 is present.
2	ADC0AIN2	RO	1	ADC Module 0 AIN2 Pin Present When set, indicates that ADC module 0 input pin 2 is present.
1	ADC0AIN1	RO	1	ADC Module 0 AIN1 Pin Present When set, indicates that ADC module 0 input pin 1 is present.
0	ADC0AIN0	RO	1	ADC Module 0 AIN0 Pin Present When set, indicates that ADC module 0 input pin 0 is present.

## Register 26: Device Capabilities 9 ADC Digital Comparators (DC9), offset 0x190

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

### Device Capabilities 9 ADC Digital Comparators (DC9)

Base 0x400F.E000

Offset 0x190

Type RO, reset 0x00FF.00FF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved								ADC1DC7	ADC1DC6	ADC1DC5	ADC1DC4	ADC1DC3	ADC1DC2	ADC1DC1	ADC1DC0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								ADC0DC7	ADC0DC6	ADC0DC5	ADC0DC4	ADC0DC3	ADC0DC2	ADC0DC1	ADC0DC0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

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	ADC1DC7	RO	1	ADC1 DC7 Present When set, indicates that ADC module 1 Digital Comparator 7 is present.
22	ADC1DC6	RO	1	ADC1 DC6 Present When set, indicates that ADC module 1 Digital Comparator 6 is present.
21	ADC1DC5	RO	1	ADC1 DC5 Present When set, indicates that ADC module 1 Digital Comparator 5 is present.
20	ADC1DC4	RO	1	ADC1 DC4 Present When set, indicates that ADC module 1 Digital Comparator 4 is present.
19	ADC1DC3	RO	1	ADC1 DC3 Present When set, indicates that ADC module 1 Digital Comparator 3 is present.
18	ADC1DC2	RO	1	ADC1 DC2 Present When set, indicates that ADC module 1 Digital Comparator 2 is present.
17	ADC1DC1	RO	1	ADC1 DC1 Present When set, indicates that ADC module 1 Digital Comparator 1 is present.
16	ADC1DC0	RO	1	ADC1 DC0 Present When set, indicates that ADC module 1 Digital Comparator 0 is present.
15:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	ADC0DC7	RO	1	ADC0 DC7 Present When set, indicates that ADC module 0 Digital Comparator 7 is present.

Bit/Field	Name	Type	Reset	Description
6	ADC0DC6	RO	1	ADC0 DC6 Present When set, indicates that ADC module 0 Digital Comparator 6 is present.
5	ADC0DC5	RO	1	ADC0 DC5 Present When set, indicates that ADC module 0 Digital Comparator 5 is present.
4	ADC0DC4	RO	1	ADC0 DC4 Present When set, indicates that ADC module 0 Digital Comparator 4 is present.
3	ADC0DC3	RO	1	ADC0 DC3 Present When set, indicates that ADC module 0 Digital Comparator 3 is present.
2	ADC0DC2	RO	1	ADC0 DC2 Present When set, indicates that ADC module 0 Digital Comparator 2 is present.
1	ADC0DC1	RO	1	ADC0 DC1 Present When set, indicates that ADC module 0 Digital Comparator 1 is present.
0	ADC0DC0	RO	1	ADC0 DC0 Present When set, indicates that ADC module 0 Digital Comparator 0 is present.

**Register 27: Non-Volatile Memory Information (NVMSTAT), offset 0x1A0**

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

**Non-Volatile Memory Information (NVMSTAT)**

Base 0x400F.E000

Offset 0x1A0

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															FWB
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	FWB	RO	1	32 Word Flash Write Buffer Active  When set, indicates that the 32 word Flash memory write buffer feature is active.

## Register 28: Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100

This register controls the clock gating logic in normal Run mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled (saving power). If the module is unlocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unlocked) unless otherwise noted, so that all functional modules 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 modules to control. This configuration is implemented 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			WDT1	reserved		CAN1	CAN0	reserved			PWM	reserved		ADC1	ADC0
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
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				MAXADC1SPD		MAXADC0SPD		reserved				WDT0	reserved		
Type	RO	RO	RO	RO	R/W	R/W	R/W	R/W	RO	RO	RO	RO	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	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	WDT1	R/W	0	WDT1 Clock Gating Control  This bit controls the clock gating for the Watchdog Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
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	CAN1	R/W	0	CAN1 Clock Gating Control  This bit controls the clock gating for CAN module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
24	CAN0	R/W	0	CAN0 Clock Gating Control  This bit controls the clock gating for CAN module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description				
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	<p>PWM Clock Gating Control</p> <p>This bit controls the clock gating for the PWM module. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>				
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.				
17	ADC1	R/W	0	<p>ADC1 Clock Gating Control</p> <p>This bit controls the clock gating for SAR ADC module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>				
16	ADC0	R/W	0	<p>ADC0 Clock Gating Control</p> <p>This bit controls the clock gating for ADC module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>				
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.				
11:10	MAXADC1SPD	R/W	0	<p>ADC1 Sample Speed</p> <p>This field sets the rate at which ADC module 1 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC1SPD bit as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x3</td> <td>1M samples/second</td> </tr> </tbody> </table>	Value	Description	0x3	1M samples/second
Value	Description							
0x3	1M samples/second							
9:8	MAXADC0SPD	R/W	0	<p>ADC0 Sample Speed</p> <p>This field sets the rate at which ADC0 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC0SPD bit as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x3</td> <td>1M samples/second</td> </tr> </tbody> </table>	Value	Description	0x3	1M samples/second
Value	Description							
0x3	1M samples/second							
7:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.				

Bit/Field	Name	Type	Reset	Description
3	WDT0	R/W	0	WDT0 Clock Gating Control  This bit controls the clock gating for the Watchdog Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.



## Register 29: Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110

This register controls the clock gating logic in Sleep mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled (saving power). If the module is unlocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unlocked) unless otherwise noted, so that all functional modules 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 modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC0** is the clock configuration register for running operation, **SCGC0** for Sleep operation, and **DCGC0** for Deep-Sleep operation. Setting the **ACG** bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

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

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			WDT1	reserved		CAN1	CAN0	reserved			PWM	reserved		ADC1	ADC0
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
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				MAXADC1SPD		MAXADC0SPD		reserved				WDT0	reserved		
Type	RO	RO	RO	RO	R/W	R/W	R/W	R/W	RO	RO	RO	RO	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	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	WDT1	R/W	0	WDT1 Clock Gating Control  This bit controls the clock gating for Watchdog Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
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	CAN1	R/W	0	CAN1 Clock Gating Control  This bit controls the clock gating for CAN module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
24	CAN0	R/W	0	CAN0 Clock Gating Control  This bit controls the clock gating for CAN module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description				
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	<p>PWM Clock Gating Control</p> <p>This bit controls the clock gating for the PWM module. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>				
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.				
17	ADC1	R/W	0	<p>ADC1 Clock Gating Control</p> <p>This bit controls the clock gating for ADC module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>				
16	ADC0	R/W	0	<p>ADC0 Clock Gating Control</p> <p>This bit controls the clock gating for ADC module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>				
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.				
11:10	MAXADC1SPD	R/W	0	<p>ADC1 Sample Speed</p> <p>This field sets the rate at which ADC module 1 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC1SPD bit as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x3</td> <td>1M samples/second</td> </tr> </tbody> </table>	Value	Description	0x3	1M samples/second
Value	Description							
0x3	1M samples/second							
9:8	MAXADC0SPD	R/W	0	<p>ADC0 Sample Speed</p> <p>This field sets the rate at which ADC module 0 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC0SPD bit as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x3</td> <td>1M samples/second</td> </tr> </tbody> </table>	Value	Description	0x3	1M samples/second
Value	Description							
0x3	1M samples/second							
7:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.				

---

Bit/Field	Name	Type	Reset	Description
3	WDT0	R/W	0	WDT0 Clock Gating Control  This bit controls the clock gating for the Watchdog Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 30: Deep Sleep Mode Clock Gating Control Register 0 (DCGC0), offset 0x120

This register controls the clock gating logic in Deep-Sleep mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled (saving power). If the module is unlocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unlocked) unless otherwise noted, so that all functional modules 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 modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC0** is the clock configuration register for running operation, **SCGC0** for Sleep operation, and **DCGC0** for Deep-Sleep operation. Setting the **ACG** bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

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

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			WDT1	reserved		CAN1	CAN0	reserved			PWM	reserved		ADC1	ADC0
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
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				MAXADC1SPD		MAXADC0SPD		reserved				WDT0	reserved		
Type	RO	RO	RO	RO	R/W	R/W	R/W	R/W	RO	RO	RO	RO	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	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	WDT1	R/W	0	WDT1 Clock Gating Control  This bit controls the clock gating for the Watchdog Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
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	CAN1	R/W	0	CAN1 Clock Gating Control  This bit controls the clock gating for CAN module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
24	CAN0	R/W	0	CAN0 Clock Gating Control  This bit controls the clock gating for CAN module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description				
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	<p>PWM Clock Gating Control</p> <p>This bit controls the clock gating for the PWM module. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>				
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.				
17	ADC1	R/W	0	<p>ADC1 Clock Gating Control</p> <p>This bit controls the clock gating for ADC module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>				
16	ADC0	R/W	0	<p>ADC0 Clock Gating Control</p> <p>This bit controls the clock gating for ADC module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>				
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.				
11:10	MAXADC1SPD	R/W	0	<p>ADC1 Sample Speed</p> <p>This field sets the rate at which ADC module 1 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC1SPD bit as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x3</td> <td>1M samples/second</td> </tr> </tbody> </table>	Value	Description	0x3	1M samples/second
Value	Description							
0x3	1M samples/second							
9:8	MAXADC0SPD	R/W	0	<p>ADC0 Sample Speed</p> <p>This field sets the rate at which ADC module 0 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADC0SPD bit as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x3</td> <td>1M samples/second</td> </tr> </tbody> </table>	Value	Description	0x3	1M samples/second
Value	Description							
0x3	1M samples/second							
7:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.				

Bit/Field	Name	Type	Reset	Description
3	WDT0	R/W	0	WDT0 Clock Gating Control  This bit controls the clock gating for the Watchdog Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 31: Run Mode Clock Gating Control Register 1 (RCGC1), offset 0x104

This register controls the clock gating logic in normal Run mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled (saving power). If the module is unclocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional modules 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 modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC1** is the clock configuration register for running operation, **SCGC1** for Sleep operation, and **DCGC1** for Deep-Sleep operation. Setting the **ACG** bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

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

Base 0x400F.E000

Offset 0x104

Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved	EPI0	reserved	I2S0	reserved	COMP2	COMP1	COMP0	reserved			TIMER3	TIMER2	TIMER1	TIMER0	
Type	RO	R/W	RO	R/W	RO	R/W	R/W	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	I2C1	reserved	I2C0	reserved		QE11	QE10	reserved		SSI1	SSI0	reserved	UART2	UART1	UART0
Type	RO	R/W	RO	R/W	RO	RO	R/W	R/W	RO	RO	R/W	R/W	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPI0	R/W	0	EPI0 Clock Gating  This bit controls the clock gating for EPI module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates 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	I2S0	R/W	0	I2S0 Clock Gating  This bit controls the clock gating for I2S module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
26	COMP2	R/W	0	<p>Analog Comparator 2 Clock Gating</p> <p>This bit controls the clock gating for analog comparator 2. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
25	COMP1	R/W	0	<p>Analog Comparator 1 Clock Gating</p> <p>This bit controls the clock gating for analog comparator 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
24	COMP0	R/W	0	<p>Analog Comparator 0 Clock Gating</p> <p>This bit controls the clock gating for analog comparator 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
23:20	reserved	RO	0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>
19	TIMER3	R/W	0	<p>Timer 3 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 3. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
18	TIMER2	R/W	0	<p>Timer 2 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 2. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
17	TIMER1	R/W	0	<p>Timer 1 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
16	TIMER0	R/W	0	<p>Timer 0 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
15	reserved	RO	0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>
14	I2C1	R/W	0	<p>I2C1 Clock Gating Control</p> <p>This bit controls the clock gating for I2C module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>



Bit/Field	Name	Type	Reset	Description
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 module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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	QE11	R/W	0	QE11 Clock Gating Control  This bit controls the clock gating for QE1 module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
8	QE10	R/W	0	QE10 Clock Gating Control  This bit controls the clock gating for QE1 module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
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	SSI1	R/W	0	SSI1 Clock Gating Control  This bit controls the clock gating for SSI module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
4	SSI0	R/W	0	SSI0 Clock Gating Control  This bit controls the clock gating for SSI module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	R/W	0	UART2 Clock Gating Control  This bit controls the clock gating for UART module 2. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
1	UART1	R/W	0	UART1 Clock Gating Control  This bit controls the clock gating for UART module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
0	UART0	R/W	0	UART0 Clock Gating Control  This bit controls the clock gating for UART module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

## Register 32: Sleep Mode Clock Gating Control Register 1 (SCGC1), offset 0x114

This register controls the clock gating logic in Sleep mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled (saving power). If the module is unclocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional modules 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 modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC1** is the clock configuration register for running operation, **SCGC1** for Sleep operation, and **DCGC1** for Deep-Sleep operation. Setting the **ACG** bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

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

Base 0x400F.E000

Offset 0x114

Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved	EPI0	reserved	I2S0	reserved	COMP2	COMP1	COMP0	reserved			TIMER3	TIMER2	TIMER1	TIMER0	
Type	RO	R/W	RO	R/W	RO	R/W	R/W	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	I2C1	reserved	I2C0	reserved		QE1	QE0	reserved		SSI1	SSI0	reserved	UART2	UART1	UART0
Type	RO	R/W	RO	R/W	RO	RO	R/W	R/W	RO	RO	R/W	R/W	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPI0	R/W	0	EPI0 Clock Gating  This bit controls the clock gating for EPI module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates 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	I2S0	R/W	0	I2S0 Clock Gating  This bit controls the clock gating for I2S module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
26	COMP2	R/W	0	<b>Analog Comparator 2 Clock Gating</b>  This bit controls the clock gating for analog comparator 2. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
25	COMP1	R/W	0	<b>Analog Comparator 1 Clock Gating</b>  This bit controls the clock gating for analog comparator 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
24	COMP0	R/W	0	<b>Analog Comparator 0 Clock Gating</b>  This bit controls the clock gating for analog comparator 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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	<b>Timer 3 Clock Gating Control</b>  This bit controls the clock gating for General-Purpose Timer module 3. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
18	TIMER2	R/W	0	<b>Timer 2 Clock Gating Control</b>  This bit controls the clock gating for General-Purpose Timer module 2. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
17	TIMER1	R/W	0	<b>Timer 1 Clock Gating Control</b>  This bit controls the clock gating for General-Purpose Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
16	TIMER0	R/W	0	<b>Timer 0 Clock Gating Control</b>  This bit controls the clock gating for General-Purpose Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	R/W	0	<b>I2C1 Clock Gating Control</b>  This bit controls the clock gating for I2C module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
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 module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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	QE11	R/W	0	QE11 Clock Gating Control  This bit controls the clock gating for QE1 module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
8	QE10	R/W	0	QE10 Clock Gating Control  This bit controls the clock gating for QE1 module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
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	SSI1	R/W	0	SSI1 Clock Gating Control  This bit controls the clock gating for SSI module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
4	SSI0	R/W	0	SSI0 Clock Gating Control  This bit controls the clock gating for SSI module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	R/W	0	UART2 Clock Gating Control  This bit controls the clock gating for UART module 2. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
1	UART1	R/W	0	UART1 Clock Gating Control  This bit controls the clock gating for UART module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
0	UART0	R/W	0	UART0 Clock Gating Control  This bit controls the clock gating for UART module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

## Register 33: Deep-Sleep Mode Clock Gating Control Register 1 (DCGC1), offset 0x124

This register controls the clock gating logic in Deep-Sleep mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled (saving power). If the module is unclocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional modules 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 modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC1** is the clock configuration register for running operation, **SCGC1** for Sleep operation, and **DCGC1** for Deep-Sleep operation. Setting the **ACG** bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

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

Base 0x400F.E000

Offset 0x124

Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved	EPI0	reserved	I2S0	reserved	COMP2	COMP1	COMP0	reserved			TIMER3	TIMER2	TIMER1	TIMER0	
Type	RO	R/W	RO	R/W	RO	R/W	R/W	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	I2C1	reserved	I2C0	reserved		QE1	QE10	reserved		SSI1	SSI0	reserved	UART2	UART1	UART0
Type	RO	R/W	RO	R/W	RO	RO	R/W	R/W	RO	RO	R/W	R/W	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPI0	R/W	0	EPI0 Clock Gating  This bit controls the clock gating for EPI module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates 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	I2S0	R/W	0	I2S0 Clock Gating  This bit controls the clock gating for I2S module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
26	COMP2	R/W	0	<p>Analog Comparator 2 Clock Gating</p> <p>This bit controls the clock gating for analog comparator 2. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
25	COMP1	R/W	0	<p>Analog Comparator 1 Clock Gating</p> <p>This bit controls the clock gating for analog comparator 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
24	COMP0	R/W	0	<p>Analog Comparator 0 Clock Gating</p> <p>This bit controls the clock gating for analog comparator 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
23:20	reserved	RO	0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>
19	TIMER3	R/W	0	<p>Timer 3 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 3. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
18	TIMER2	R/W	0	<p>Timer 2 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 2. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
17	TIMER1	R/W	0	<p>Timer 1 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
16	TIMER0	R/W	0	<p>Timer 0 Clock Gating Control</p> <p>This bit controls the clock gating for General-Purpose Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
15	reserved	RO	0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>
14	I2C1	R/W	0	<p>I2C1 Clock Gating Control</p> <p>This bit controls the clock gating for I2C module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>



Bit/Field	Name	Type	Reset	Description
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 module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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	QE11	R/W	0	QE11 Clock Gating Control  This bit controls the clock gating for QE1 module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
8	QE10	R/W	0	QE10 Clock Gating Control  This bit controls the clock gating for QE1 module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
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	SSI1	R/W	0	SSI1 Clock Gating Control  This bit controls the clock gating for SSI module 1. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
4	SSI0	R/W	0	SSI0 Clock Gating Control  This bit controls the clock gating for SSI module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	R/W	0	UART2 Clock Gating Control  This bit controls the clock gating for UART module 2. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
1	UART1	R/W	0	UART1 Clock Gating Control  This bit controls the clock gating for UART module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
0	UART0	R/W	0	UART0 Clock Gating Control  This bit controls the clock gating for UART module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

## Register 34: Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108

This register controls the clock gating logic in normal Run mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled (saving power). If the module is unlocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unlocked) unless otherwise noted, so that all functional modules 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 modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC2** is the clock configuration register for running operation, **SCGC2** for Sleep operation, and **DCGC2** for Deep-Sleep operation. Setting the **ACG** bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

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

Base 0x400F.E000

Offset 0x108

Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															USB0
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		UDMA	reserved				GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GIPOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:17	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	USB0	R/W	0	<p>USB0 Clock Gating Control</p> <p>This bit controls the clock gating for USB module 0. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
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	UDMA	R/W	0	<p>Micro-DMA Clock Gating Control</p> <p>This bit controls the clock gating for micro-DMA. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
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.

Bit/Field	Name	Type	Reset	Description
8	GPIOJ	R/W	0	<p>Port J Clock Gating Control</p> <p>This bit controls the clock gating for Port J. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
7	GPIOH	R/W	0	<p>Port H Clock Gating Control</p> <p>This bit controls the clock gating for Port H. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
6	GPIOG	R/W	0	<p>Port G Clock Gating Control</p> <p>This bit controls the clock gating for Port G. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
5	GPIOF	R/W	0	<p>Port F Clock Gating Control</p> <p>This bit controls the clock gating for Port F. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
4	GPIOE	R/W	0	<p>Port E Clock Gating Control</p> <p>Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
3	GPIOD	R/W	0	<p>Port D Clock Gating Control</p> <p>Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
2	GPIOC	R/W	0	<p>Port C Clock Gating Control</p> <p>This bit controls the clock gating for Port C. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
1	GPIOB	R/W	0	<p>Port B Clock Gating Control</p> <p>This bit controls the clock gating for Port B. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
0	GPIOA	R/W	0	<p>Port A Clock Gating Control</p> <p>This bit controls the clock gating for Port A. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>

## Register 35: Sleep Mode Clock Gating Control Register 2 (SCGC2), offset 0x118

This register controls the clock gating logic in Sleep mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled (saving power). If the module is unclocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional modules 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 modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC2** is the clock configuration register for running operation, **SCGC2** for Sleep operation, and **DCGC2** for Deep-Sleep operation. Setting the **ACG** bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

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

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															USB0
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		UDMA	reserved				GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:17	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	USB0	R/W	0	<p>USB0 Clock Gating Control</p> <p>This bit controls the clock gating for USB module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.</p>
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	UDMA	R/W	0	<p>Micro-DMA Clock Gating Control</p> <p>This bit controls the clock gating for micro-DMA. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.</p>
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.

Bit/Field	Name	Type	Reset	Description
8	GPIOJ	R/W	0	Port J Clock Gating Control  This bit controls the clock gating for Port J. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
7	GPIOH	R/W	0	Port H Clock Gating Control  This bit controls the clock gating for Port H. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
6	GPIOG	R/W	0	Port G Clock Gating Control  This bit controls the clock gating for Port G. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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 module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
4	GPIOE	R/W	0	Port E Clock Gating Control  Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.
3	GPIOD	R/W	0	Port D Clock Gating Control  Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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 module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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 module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates 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 module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.

## Register 36: Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128

This register controls the clock gating logic in Deep-Sleep mode. Each bit controls a clock enable for a given interface, function, or module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled (saving power). If the module is unclocked, reads or writes to the module generate a bus fault. The reset state of these bits is 0 (unclocked) unless otherwise noted, so that all functional modules 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 modules to control. This configuration is implemented to assure reasonable code compatibility with other family and future parts. **RCGC2** is the clock configuration register for running operation, **SCGC2** for Sleep operation, and **DCGC2** for Deep-Sleep operation. Setting the **ACG** bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

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

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved															USB0	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved		UDMA	reserved				GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GIPOD	GPIOC	GPIOB	GPIOA	
Type	RO	RO	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	USB0	R/W	0	<p>USB0 Clock Gating Control</p> <p>This bit controls the clock gating for USB module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.</p>
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	UDMA	R/W	0	<p>Micro-DMA Clock Gating Control</p> <p>This bit controls the clock gating for micro-DMA. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.</p>
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.

Bit/Field	Name	Type	Reset	Description
8	GPIOJ	R/W	0	<p>Port J Clock Gating Control</p> <p>This bit controls the clock gating for Port J. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
7	GPIOH	R/W	0	<p>Port H Clock Gating Control</p> <p>This bit controls the clock gating for Port H. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
6	GPIOG	R/W	0	<p>Port G Clock Gating Control</p> <p>This bit controls the clock gating for Port G. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
5	GPIOF	R/W	0	<p>Port F Clock Gating Control</p> <p>This bit controls the clock gating for Port F. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
4	GPIOE	R/W	0	<p>Port E Clock Gating Control</p> <p>Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
3	GPIOD	R/W	0	<p>Port D Clock Gating Control</p> <p>Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
2	GPIOC	R/W	0	<p>Port C Clock Gating Control</p> <p>This bit controls the clock gating for Port C. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
1	GPIOB	R/W	0	<p>Port B Clock Gating Control</p> <p>This bit controls the clock gating for Port B. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>
0	GPIOA	R/W	0	<p>Port A Clock Gating Control</p> <p>This bit controls the clock gating for Port A. If set, the module receives a clock and functions. Otherwise, the module is unlocked and disabled. If the module is unlocked, a read or write to the module generates a bus fault.</p>



**Register 37: Software Reset Control 0 (SRCR0), offset 0x040**

This register allows individual modules to be reset. Writes to this register are masked by the bits in the **Device Capabilities 1 (DC1)** register.

## Software Reset Control 0 (SRCR0)

Base 0x400F.E000

Offset 0x040

Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			WDT1	reserved		CAN1	CAN0	reserved			PWM	reserved		ADC1	ADC0
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
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												WDT0	reserved		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	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	WDT1	R/W	0	WDT1 Reset Control  When this bit is set, Watchdog Timer module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
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	CAN1	R/W	0	CAN1 Reset Control  When this bit is set, CAN module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
24	CAN0	R/W	0	CAN0 Reset Control  When this bit is set, CAN module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
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  When this bit is set, PWM module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
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	ADC1	R/W	0	ADC1 Reset Control  When this bit is set, ADC module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
16	ADC0	R/W	0	ADC0 Reset Control  When this bit is set, ADC module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
15:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT0	R/W	0	WDT0 Reset Control  When this bit is set, Watchdog Timer module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
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 38: Software Reset Control 1 (SRCR1), offset 0x044

This register allows individual modules to be reset. 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	EPI0	reserved	I2S0	reserved	COMP2	COMP1	COMP0	reserved			TIMER3	TIMER2	TIMER1	TIMER0	
Type	RO	R/W	RO	R/W	RO	R/W	R/W	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	I2C1	reserved	I2C0	reserved		QE1	QE10	reserved		SSI1	SSI0	reserved	UART2	UART1	UART0
Type	RO	R/W	RO	R/W	RO	RO	R/W	R/W	RO	RO	R/W	R/W	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPI0	R/W	0	EPI0 Reset Control  When this bit is set, EPI module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
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	I2S0	R/W	0	I2S0 Reset Control  When this bit is set, I2S module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	COMP2	R/W	0	Analog Comp 2 Reset Control  When this bit is set, Analog Comparator module 2 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
25	COMP1	R/W	0	Analog Comp 1 Reset Control  When this bit is set, Analog Comparator module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
24	COMP0	R/W	0	Analog Comp 0 Reset Control  When this bit is set, Analog Comparator module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.

Bit/Field	Name	Type	Reset	Description
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  Timer 3 Reset Control. When this bit is set, General-Purpose Timer module 3 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
18	TIMER2	R/W	0	Timer 2 Reset Control  When this bit is set, General-Purpose Timer module 2 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
17	TIMER1	R/W	0	Timer 1 Reset Control  When this bit is set, General-Purpose Timer module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
16	TIMER0	R/W	0	Timer 0 Reset Control  When this bit is set, General-Purpose Timer module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	R/W	0	I2C1 Reset Control  When this bit is set, I2C module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
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  When this bit is set, I2C module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
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	QE11	R/W	0	QE11 Reset Control  When this bit is set, QE1 module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
8	QE10	R/W	0	QE10 Reset Control  When this bit is set, QE1 module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.

Bit/Field	Name	Type	Reset	Description
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	SSI1	R/W	0	SSI1 Reset Control  When this bit is set, SSI module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
4	SSI0	R/W	0	SSI0 Reset Control  When this bit is set, SSI module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	R/W	0	UART2 Reset Control  When this bit is set, UART module 2 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
1	UART1	R/W	0	UART1 Reset Control  When this bit is set, UART module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
0	UART0	R/W	0	UART0 Reset Control  When this bit is set, UART module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.

## Register 39: Software Reset Control 2 (SRCR2), offset 0x048

This register allows individual modules to be reset. 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															USB0
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		UDMA	reserved				GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:17	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	USB0	R/W	0	<p>USB0 Reset Control</p> <p>When this bit is set, USB module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.</p>
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	UDMA	R/W	0	<p>Micro-DMA Reset Control</p> <p>When this bit is set, uDMA module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.</p>
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	GPIOJ	R/W	0	<p>Port J Reset Control</p> <p>When this bit is set, Port J module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.</p>
7	GPIOH	R/W	0	<p>Port H Reset Control</p> <p>When this bit is set, Port H module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.</p>
6	GPIOG	R/W	0	<p>Port G Reset Control</p> <p>When this bit is set, Port G module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.</p>

---

Bit/Field	Name	Type	Reset	Description
5	GPIOF	R/W	0	Port F Reset Control  When this bit is set, Port F module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
4	GPIOE	R/W	0	Port E Reset Control  When this bit is set, Port E module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
3	GPIOD	R/W	0	Port D Reset Control  When this bit is set, Port D module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
2	GPIOC	R/W	0	Port C Reset Control  When this bit is set, Port C module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
1	GPIOB	R/W	0	Port B Reset Control  When this bit is set, Port B module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
0	GPIOA	R/W	0	Port A Reset Control  When this bit is set, Port A module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.

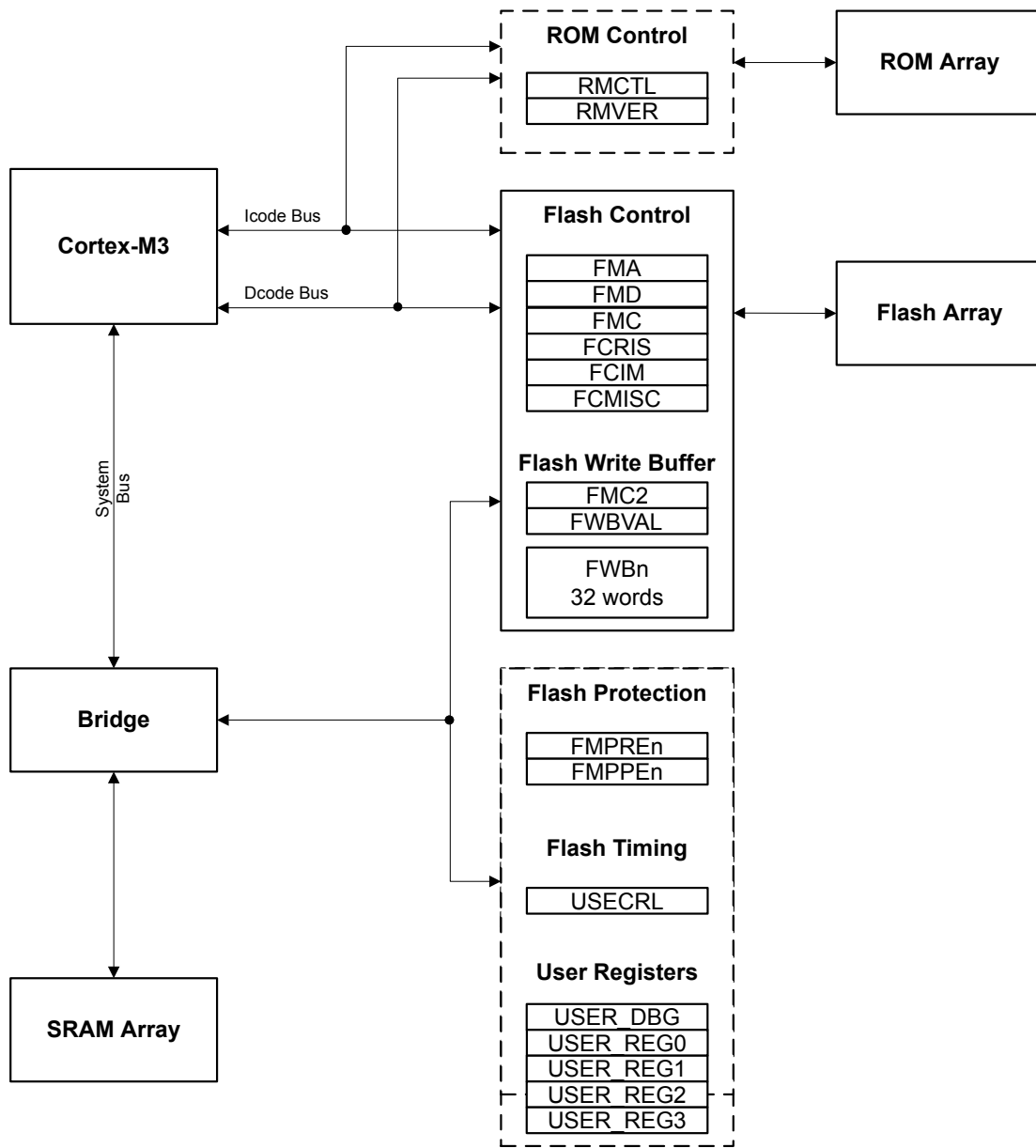
## 7 Internal Memory

The LM3S5791 microcontroller comes with 64 KB of bit-banded SRAM, internal ROM, and 128 KB of Flash memory. The Flash controller provides a user-friendly interface, making Flash programming a simple task. Flash protection can be applied to the Flash memory on a 2-KB block basis.

### 7.1 Block Diagram

Figure 7-1 on page 188 illustrates the Flash functions. The dashed boxes in the figure indicate registers residing in the System Control module rather than the Flash Control module.

Figure 7-1. Flash Block Diagram





## 7.2 Functional Description

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

### 7.2.1 SRAM

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

The internal SRAM of the Stellaris® 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 base is located at address 0x2200.0000.

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

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

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

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

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

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

### 7.2.2 ROM

The internal ROM of the Stellaris® device is located at address 0x0100.0000 of the device memory map. The ROM contains the following components:

- Stellaris® Boot Loader and vector table (see “Boot Loader” on page 967)
- Stellaris® Peripheral Driver Library (DriverLib) release for product-specific peripherals and interfaces (see “ROM DriverLib Functions” on page 972)
- Advanced Encryption Standard (AES) cryptography tables (see “Advance Encryption Standard and Cyclic Redundancy Check Software in ROM” on page 991)
- Cyclic Redundancy Check (CRC) error detection functionality (see “Advance Encryption Standard and Cyclic Redundancy Check Software in ROM” on page 991)

### 7.2.3 Flash Memory

The Flash is organized as a set of 1-KB blocks that can be individually erased. An individual 32-bit word can be programmed to change bits from 1 to 0. In addition, a write buffer provides the ability to concurrently program 32 continuous words in Flash memory. Erasing a block causes the entire contents of the block to be reset to all 1s. The 1-KB blocks are paired into sets 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.

The Flash memory controller has a prefetch buffer that is automatically used when the CPU frequency is greater than 50 MHz. The prefetch buffer fetches two 32-bit words per clock allowing Flash memory to be read with no wait states while code is executing linearly. Branches incur a single wait state.

### 7.2.3.1 Flash Memory Protection

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

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

The policies may be combined as shown in Table 7-1 on page 190.

**Table 7-1. 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 program-protected block is prohibited. An access that attempts to read a read-protected block is prohibited. Such accesses return data of all 0s. A controller interrupt may be optionally generated whenever an attempt is made to improperly access the Flash memory (by setting the **AMASK** bit in the **Flash Controller Interrupt Mask (FCIM)** register) to alert software developers of poorly behaving software during the development and debug phases.

The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. These settings create a policy of open access and programmability. The register bits may be changed by clearing the specific register bit. The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The changes are committed using the **Flash Memory Control (FMC)** register. Details on programming these bits are discussed in “Nonvolatile Register Programming” on page 192.

## 7.3 Flash Memory Initialization and Configuration

### 7.3.1 Flash Programming

The Stellaris<sup>®</sup> devices provide a user-friendly interface for Flash programming. All erase/program operations are handled via three registers: **Flash Memory Address (FMA)**, **Flash Memory Data**

(**FMD**), and **Flash Memory Control (FMC)**. Note that if the debug capabilities of the microcontroller have been deactivated, resulting in a "locked" state, a recovery sequence must be performed in order to reactivate the debug module. See "Recovering a "Locked" Microcontroller" on page 79.

### 7.3.1.1 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.

---

**Important:** To ensure proper operation, two writes to the same word must be separated by an `ERASE`.

---

### 7.3.1.2 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.

### 7.3.1.3 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.

## 7.3.2 32-Word Flash Write Buffer

A 32-word write buffer provides the capability to perform faster write accesses to the Flash memory by concurrently programming 32 words with a single buffered Flash write operation. The buffered Flash write operation takes the same amount of time as the single word write operation controlled by bit 0 in the **FMC** register. The data for the buffered write is written to the **Flash Write Buffer (FWBn)** registers.

The registers are 32-word aligned with Flash memory, and therefore the register **FWB0** corresponds with the address in **FMA** where bits [6:0] of **FMA** are all 0. **FWB1** corresponds with the address in **FMA** + 0x4 and so on. Only the **FWBn** registers that have been updated since the previous buffered Flash write operation are written. The **Flash Write Buffer Valid (FWBVAL)** register shows which registers have been written since the last buffered Flash write operation. This register contains a bit for each of the 32 **FWBn** registers, where bit[n] of **FWBVAL** corresponds to **FWBn**. The **FWBn** register has been updated if the corresponding bit in the **FWBVAL** register is set.

### 7.3.2.1 To program 32 words with a single buffered Flash write operation

1. Write the source data to the **FWBn** registers.
2. Write the target address to the **FMA** register. This must be a 32-word aligned address (that is, bits [6:0] in **FMA** must be 0s).
3. Write the Flash write key and the `WRBUF` bit (a value of 0xA442.0001) to the **FMC2** register.

4. Poll the **FMC2** register until the `WRBUF` bit is cleared.

### 7.3.3 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 using the `COMT` bit in the **FMC** register to activate a write operation. With the exception of the **USER\_DBG** register, the settings in these registers can be tested before committing them to Flash memory.

For the **USER\_DBG** register, the data to be written is loaded into the **FMD** register before it is committed. The **FMD** register is read only and does not allow the **USER\_DBG** operation to be tried before committing it to nonvolatile memory.

**Important:** These registers can only have bits changed from 1 to 0 by user programming, but can be restored to their factory default values only by performing the sequence described in "Recovering a "Locked" Microcontroller" on page 79. The mass erase of the main Flash array caused by the sequence is performed prior to restoring these registers.

In addition, the **USER\_REG0**, **USER\_REG1**, **USER\_REG2**, **USER\_REG3**, and **USER\_DBG** registers each use bit 31 (`NW`) to indicate that they are available for user write. These five registers can only be committed once whereas the Flash protection registers may be committed multiple times. Table 7-2 on page 192 provides the **FMA** address required for commitment of each of the registers and the source of the data to be written when 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 7-2. User-Programmable Flash Resident Registers**

Register to be Committed	FMA Value	Data Source
FMPRE0	0x0000.0000	FMPRE0
FMPRE1	0x0000.0002	FMPRE1
FMPPE0	0x0000.0001	FMPPE0
FMPPE1	0x0000.0003	FMPPE1
USER_REG0	0x8000.0000	USER_REG0
USER_REG1	0x8000.0001	USER_REG1
USER_REG2	0x8000.0002	USER_REG2
USER_REG3	0x8000.0003	USER_REG3
USER_DBG	0x7510.0000	FMD

## 7.4 Register Map

Table 7-3 on page 193 lists the ROM Controller register and the Flash memory and control registers. The offset listed is a hexadecimal increment to the register's address. The **FMA**, **FMD**, **FMC**, **FCRIS**, **FCIM**, **FCMISC**, **FMC2**, **FWBVAL**, and **FWBn** register offsets are relative to the Flash control base address of `0x400F.D000`. The ROM and Flash protection register offsets are relative to the System Control base address of `0x400F.E000`.

Table 7-3. Flash Register Map

Offset	Name	Type	Reset	Description	See page
<b>Flash Registers (Flash Control Offset)</b>					
0x000	FMA	R/W	0x0000.0000	Flash Memory Address	194
0x004	FMD	R/W	0x0000.0000	Flash Memory Data	195
0x008	FMC	R/W	0x0000.0000	Flash Memory Control	196
0x00C	FCRIS	RO	0x0000.0000	Flash Controller Raw Interrupt Status	198
0x010	FCIM	R/W	0x0000.0000	Flash Controller Interrupt Mask	199
0x014	FCMISC	R/W1C	0x0000.0000	Flash Controller Masked Interrupt Status and Clear	200
0x020	FMC2	R/W	0x0000.0000	Flash Memory Control 2	201
0x030	FWBVAL	R/W	0x0000.0000	Flash Write Buffer Valid	202
0x100 - 0x13C	FWBn	R/W	0x0000.0000	Flash Write Buffer n	203
<b>Memory Registers (System Control Offset)</b>					
0x0F0	RMCTL	R/W1C	-	ROM Control	204
0x0F4	RMVER	RO	0x0202.5400	ROM Version Register	205
0x130	FMPRE0	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 0	206
0x200	FMPRE0	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 0	206
0x134	FMPPE0	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 0	207
0x400	FMPPE0	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 0	207
0x1D0	USER_DBG	R/W	0xFFFF.FFFE	User Debug	208
0x1E0	USER_REG0	R/W	0xFFFF.FFFF	User Register 0	209
0x1E4	USER_REG1	R/W	0xFFFF.FFFF	User Register 1	210
0x1E8	USER_REG2	R/W	0xFFFF.FFFF	User Register 2	211
0x1EC	USER_REG3	R/W	0xFFFF.FFFF	User Register 3	212
0x204	FMPRE1	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 1	213
0x208	FMPRE2	R/W	0x0000.0000	Flash Memory Protection Read Enable 2	214
0x20C	FMPRE3	R/W	0x0000.0000	Flash Memory Protection Read Enable 3	215
0x404	FMPPE1	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 1	216
0x408	FMPPE2	R/W	0x0000.0000	Flash Memory Protection Program Enable 2	217
0x40C	FMPPE3	R/W	0x0000.0000	Flash Memory Protection Program Enable 3	218

## 7.5 Flash Register Descriptions (Flash Control Offset)

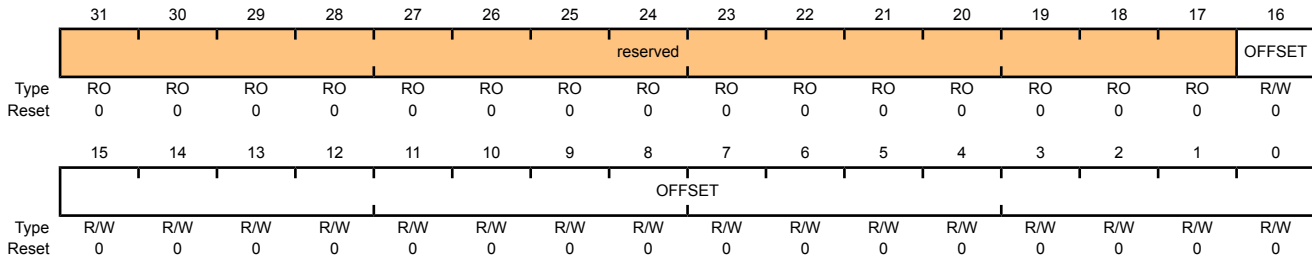
This section lists and describes the Flash Memory registers, in numerical order by address offset. Registers in this section are relative to the Flash control base address of 0x400F.D000.

### Register 1: Flash Memory Address (FMA), offset 0x000

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

#### Flash Memory Address (FMA)

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



Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16:0	OFFSET	R/W	0x0	Address Offset  Address offset in Flash where operation is performed, except for nonvolatile registers (see "Nonvolatile Register Programming" on page 192 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 erase cycles.

### Flash Memory Data (FMD)

Base 0x400F.D000

Offset 0x004

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:0	DATA	R/W	0x0000.0000	Data Value Data value for write operation.

### Register 3: Flash Memory Control (FMC), offset 0x008

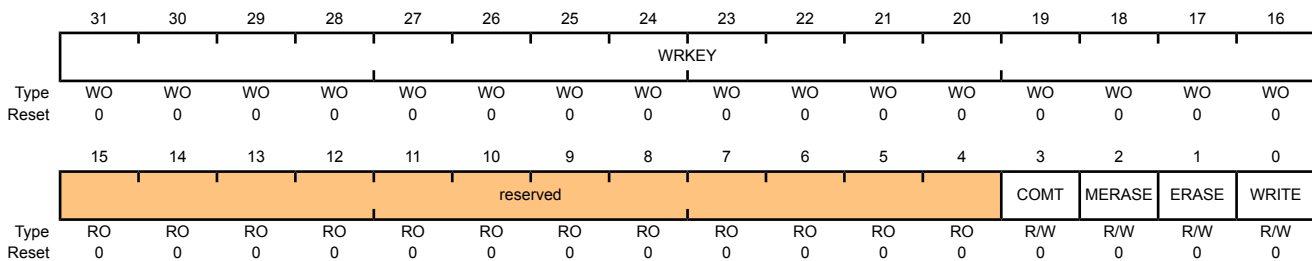
When this register is written, the Flash controller initiates the appropriate access cycle for the location specified by the **Flash Memory Address (FMA)** register (see page 194). If the access is a write access, the data contained in the **Flash Memory Data (FMD)** register (see page 195) is written to the specified address.

This register must be the final register written and initiates the memory operation. The four control bits in the lower byte of this register are used to initiate memory operations.

Care must be taken not to set multiple control bits as the results of such an operation are unpredictable.

#### Flash Memory Control (FMC)

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



Bit/Field	Name	Type	Reset	Description
31:16	WRKEY	WO	0x0000	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 Flash 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	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	COMT	R/W	0	Commit Register Value  This bit is used to commit writes to Flash-resident registers and to monitor the progress of that process.

- Value Description
- 1 Set this bit to commit (write) the register value to a Flash-resident register.  
  
When read, a 1 indicates that the previous commit access is not complete.
  - 0 A write of 0 has no effect on the state of this bit.  
  
When read, a 0 indicates that the previous commit access is complete.

A commit can take up to 50  $\mu$ s.  
 See "Nonvolatile Register Programming" on page 192 for more information on programming Flash-resident registers.



Bit/Field	Name	Type	Reset	Description
2	MERASE	R/W	0	<p>Mass Erase Flash Memory</p> <p>This bit is used to mass erase the Flash main memory and to monitor the progress of that process.</p> <p>Value Description</p> <p>1 Set this bit to erase the Flash main memory.</p> <p>When read, a 1 indicates that the previous mass erase access is not complete.</p> <p>0 A write of 0 has no effect on the state of this bit.</p> <p>When read, a 0 indicates that the previous mass erase access is complete.</p> <p>A mass erase can take up to 250 ms.</p>
1	ERASE	R/W	0	<p>Erase a Page of Flash Memory</p> <p>This bit is used to erase a page of Flash memory and to monitor the progress of that process.</p> <p>Value Description</p> <p>1 Set this bit to erase the Flash memory page specified by the contents of the <b>FMA</b> register.</p> <p>When read, a 1 indicates that the previous page erase access is not complete.</p> <p>0 A write of 0 has no effect on the state of this bit.</p> <p>When read, a 0 indicates that the previous page erase access is complete.</p> <p>A page erase can take up to 25 ms.</p>
0	WRITE	R/W	0	<p>Write a Word into Flash Memory</p> <p>This bit is used to write a word into Flash memory and to monitor the progress of that process.</p> <p>Value Description</p> <p>1 Set this bit to write the data stored in the <b>FMD</b> register into the Flash memory location specified by the contents of the <b>FMA</b> register.</p> <p>When read, a 1 indicates that the write update access is not complete.</p> <p>0 A write of 0 has no effect on the state of this bit.</p> <p>When read, a 0 indicates that the previous write update access is complete.</p> <p>Writing a single word 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 sent to the interrupt controller only if the corresponding **FCIM** register bit is set.

### Flash Controller Raw Interrupt Status (FCRIS)

Base 0x400F.D000

Offset 0x00C

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PRIS	RO	0	<p>Programming Raw Interrupt Status</p> <p>This bit provides status on programming cycles which are write or erase actions generated through the <b>FMC</b> or <b>FMC2</b> register bits (see page 196 and page 201).</p> <p>Value Description</p> <ul style="list-style-type: none"> <li>1 The programming cycle has completed.</li> <li>0 The programming cycle has not completed.</li> </ul> <p>This status is sent to the interrupt controller when the <b>PMASK</b> bit in the <b>FCIM</b> register is set.</p> <p>This bit is cleared by writing a 1 to the <b>PMISC</b> bit in the <b>FCMISC</b> register.</p>
0	ARIS	RO	0	<p>Access Raw Interrupt Status</p> <p>This bit indicates if the Flash was improperly accessed.</p> <p>Value Description</p> <ul style="list-style-type: none"> <li>1 The program tried to access the Flash memory counter to the policy set in the <b>FMPREn</b> and <b>FMPPEn</b> registers.</li> <li>0 No access has tried to improperly access the Flash.</li> </ul> <p>This status is sent to the interrupt controller when the <b>AMASK</b> bit in the <b>FCIM</b> register is set.</p> <p>This bit is cleared by writing a 1 to the <b>AMISC</b> bit in the <b>FCMISC</b> register.</p>

## 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

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

Bit/Field	Name	Type	Reset	Description				
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.				
1	PMASK	R/W	0	<p>Programming Interrupt Mask</p> <p>This bit controls the reporting of the programming raw interrupt status to the interrupt controller.</p> <p>Value Description</p> <table border="0"> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the <code>PRIS</code> bit is set.</td> </tr> <tr> <td>0</td> <td>The <code>PRIS</code> interrupt is suppressed and not sent to the interrupt controller.</td> </tr> </table>	1	An interrupt is sent to the interrupt controller when the <code>PRIS</code> bit is set.	0	The <code>PRIS</code> interrupt is suppressed and not sent to the interrupt controller.
1	An interrupt is sent to the interrupt controller when the <code>PRIS</code> bit is set.							
0	The <code>PRIS</code> interrupt is suppressed and not sent to the interrupt controller.							
0	AMASK	R/W	0	<p>Access Interrupt Mask</p> <p>This bit controls the reporting of the access raw interrupt status to the interrupt controller.</p> <p>Value Description</p> <table border="0"> <tr> <td>1</td> <td>An interrupt is sent to the interrupt controller when the <code>ARIS</code> bit is set.</td> </tr> <tr> <td>0</td> <td>The <code>ARIS</code> interrupt is suppressed and not sent to the interrupt controller.</td> </tr> </table>	1	An interrupt is sent to the interrupt controller when the <code>ARIS</code> bit is set.	0	The <code>ARIS</code> interrupt is suppressed and not sent to the interrupt controller.
1	An interrupt is sent to the interrupt controller when the <code>ARIS</code> bit is set.							
0	The <code>ARIS</code> interrupt is suppressed and not sent to the interrupt controller.							

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

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

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

Base 0x400F.D000

Offset 0x014

Type R/W1C, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PMISC	R/W1C	0	<p>Programming Masked Interrupt Status and Clear</p> <p>Value Description</p> <p>1 When read, a 1 indicates that an unmasked interrupt was signaled because a programming cycle completed.</p> <p>Writing a 1 to this bit clears <code>PMISC</code> and also the <code>PRIS</code> bit in the <code>FCRIS</code> register (see page 198).</p> <p>0 When read, a 0 indicates that a programming cycle complete interrupt has not occurred.</p> <p>A write of 0 has no effect on the state of this bit.</p>
0	AMISC	R/W1C	0	<p>Access Masked Interrupt Status and Clear</p> <p>Value Description</p> <p>1 When read, a 1 indicates that an unmasked interrupt was signaled because an improper access to protected Flash memory was attempted.</p> <p>Writing a 1 to this bit clears <code>AMISC</code> and also the <code>ARIS</code> bit in the <code>FCRIS</code> register (see page 198).</p> <p>0 When read, a 0 indicates that no improper accesses have occurred.</p> <p>A write of 0 has no effect on the state of this bit.</p>

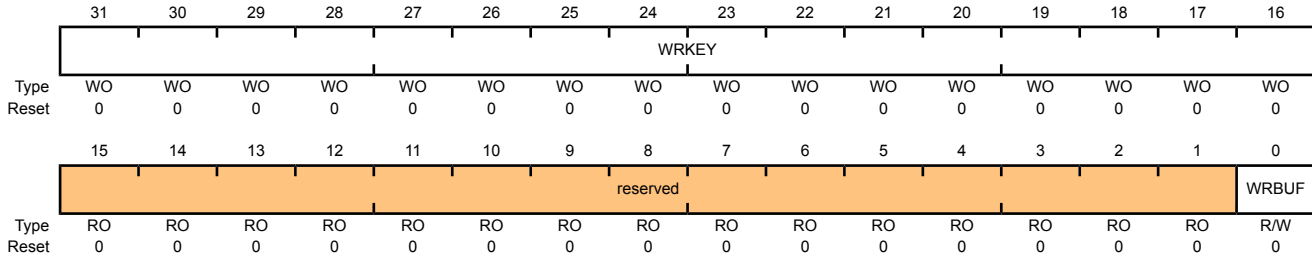
### Register 7: Flash Memory Control 2 (FMC2), offset 0x020

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 194). If the access is a write access, the data contained in the **Flash Write Buffer (FWB)** registers is written.

This register must be the final register written as it initiates the memory operation.

#### Flash Memory Control 2 (FMC2)

Base 0x400F.D000  
 Offset 0x020  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	WRKEY	WO	0x0000	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>FMC2</b> register without this <b>WRKEY</b> value are ignored. A read of this field returns the value 0.
15:1	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.
0	WRBUF	R/W	0	Buffered Flash Write

This bit is used to start a buffered Flash write to Flash Memory.

#### Value Description

1 Set this bit to write the data stored in the **FWBn** registers to the location specified by the contents of the **FMA** register.

When read, a 1 indicates that the previous buffered Flash write access is not complete.

0 A write of 0 has no effect on the state of this bit.

When read, a 0 indicates that the previous buffered Flash write access is complete.

A buffered Flash write can take up to 4 ms.

## Register 8: Flash Write Buffer Valid (FWBVAL), offset 0x030

This register provides a bitwise status of which **FWB<sub>n</sub>** registers have been written by the processor since the last write of the Flash write buffer. The entries with a 1 are written on the next write of the Flash write buffer. This register is cleared after the write operation by hardware. A protection violation on the write operation also clears this status.

Software can program the same 32 words to various Flash memory locations by setting the **FWB<sub>[n]</sub>** bits after they are cleared by the write operation. The next write operation then uses the same data as the previous one. In addition, if a **FWB<sub>n</sub>** register change should not be written to Flash memory, software can clear the corresponding **FWB<sub>[n]</sub>** bit to preserve the existing data when the next write operation occurs.

### Flash Write Buffer Valid (FWBVAL)

Base 0x400F.D000

Offset 0x030

Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	FWB[n]															
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
	FWB[n]															
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	FWB[n]	R/W	0x0	Flash Write Buffer

#### Value Description

- |   |  |
|---|--|
| 1 | The corresponding <b>FWB<sub>n</sub></b> register has been updated since the last buffer write operation and is ready to be written to Flash memory. |
| 0 | The corresponding <b>FWB<sub>n</sub></b> register has no new data to be written.   |

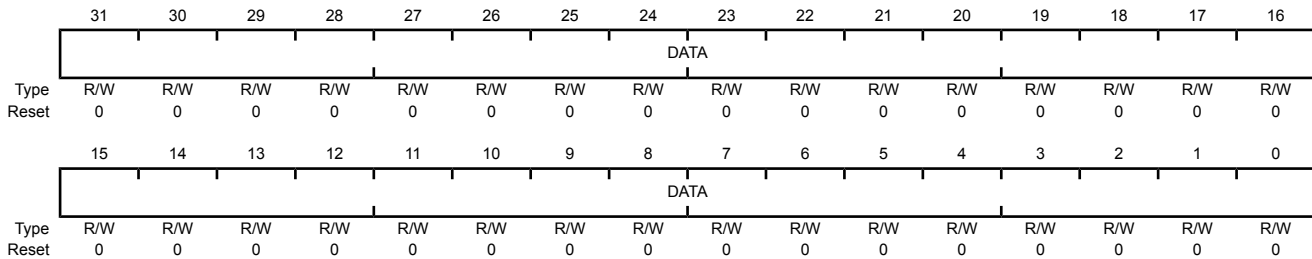
Bit 0 corresponds to **FWB0**, offset 0x100, and bit 31 corresponds to **FWB31**, offset 0x13C.

### Register 9: Flash Write Buffer n (FWBn), offset 0x100 - 0x13C

These 32 registers hold the contents of the data to be written into the Flash on a buffered Flash write operation. The offset selects one of the 32-bit registers. Only **FWBn** registers that have been updated since the preceding buffered Flash write operation are written into the Flash, so it is not necessary to write the entire bank of registers in order to write 1 or 2 words. The **FWBn** registers are written into the Flash with the **FWB0** register corresponding to the address contained in **FMA**. **FWB1** is written to the address **FMA+0x4** etc. Note that only data bits that are 0 result in the Flash memory being modified. A data bit that is 1 leaves the content of the Flash memory bit at its previous value.

#### Flash Write Buffer n (FWBn)

Base 0x400F.D000  
 Offset 0x100 - 0x13C  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	DATA	R/W	0x0000.0000	Data

Data to be written into the Flash.

## 7.6 Memory Register Descriptions (System Control Offset)

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

**Register 10: ROM Control (RMCTL), offset 0x0F0**

This register provides control of the ROM controller state. This register offset is relative to the System Control base address of 0x400F.E000.

## ROM Control (RMCTL)

Base 0x400F.E000

Offset 0x0F0

Type R/W1C, reset -

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	BA	R/W1C	-	Boot Alias

Upon reset, the system control module checks the first two words of the Flash memory to see if it has been programmed. If the first two words of Flash memory contain 0xFFFF.FFFF then it has not yet been programmed, and this bit is then set by hardware so that the on-chip ROM appears at address 0x0.

## Value Description

- 1 The microcontroller's ROM appears at address 0x0. This bit is set automatically if the first two words of the Flash memory contain 0xFFFF.FFFF.
- 0 The Flash memory is at address 0x0.

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



**Register 11: ROM Version Register (RMVER), offset 0x0F4****Note:** Offset is relative to System Control base address of 0x400FE000.

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

## ROM Version Register (RMVER)

Base 0x400F.E000

Offset 0x0F4

Type RO, reset 0x0202.5400

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

Bit/Field	Name	Type	Reset	Description
31:24	CONT	RO	0x02	ROM Contents  Value Description 0x02 Stellaris Boot Loader & DriverLib with AES
23:16	SIZE	RO	0x02	ROM Size  This field encodes the size of the ROM.  Value Description 0x02 Stellaris Boot Loader & DriverLib with AES, ethernet
15:8	VER	RO	0x54	ROM Version
7:0	REV	RO	0x0	ROM Revision

## Register 12: Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x130 and 0x200

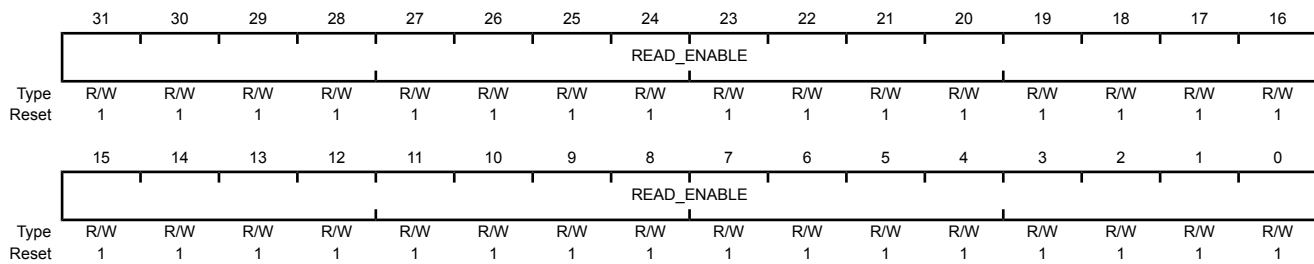
**Note:** This register is aliased for backwards compatibility.

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

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

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

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



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

## Register 13: Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x134 and 0x400

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

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

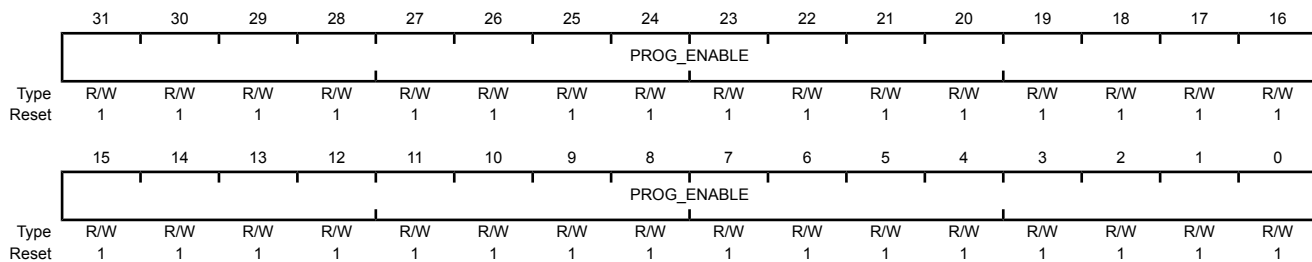
This register stores the execute-only protection bits for each 2-KB flash block (**FMPREN** stores the execute-only bits). This register is loaded during the power-on reset sequence. The factory settings for the **FMPREN** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves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

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

Base 0x400F.E000

Offset 0x134 and 0x400

Type R/W, reset 0xFFFF.FFFF



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

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

Value	Description
-------	-------------

0xFFFFFFFF	Enables 128 KB of Flash memory.
------------	---------------------------------

**Register 14: User Debug (USER\_DBG), offset 0x1D0**

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

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

**User Debug (USER\_DBG)**

Base 0x400F.E000

Offset 0x1D0

Type R/W, reset 0xFFFF.FFFE

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

Bit/Field	Name	Type	Reset	Description
31	NW	R/W	1	User Debug Not Written. When set, this bit specifies that this 32-bit register has not been written. When clear, this bit specifies that this register has been written and may not be written again.
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 15: User Register 0 (USER\_REG0), offset 0x1E0**

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

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

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

Base 0x400F.E000

Offset 0x1E0

Type R/W, reset 0xFFFF.FFFF

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

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

### Register 16: User Register 1 (USER\_REG1), offset 0x1E4

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

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

#### User Register 1 (USER\_REG1)

Base 0x400F.E000

Offset 0x1E4

Type R/W, reset 0xFFFF.FFFF

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

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

**Register 17: User Register 2 (USER\_REG2), offset 0x1E8**

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

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

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

Base 0x400F.E000

Offset 0x1E8

Type R/W, reset 0xFFFF.FFFF

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

Bit/Field	Name	Type	Reset	Description
31	NW	R/W	1	Not Written. When set, this bit specifies that this 32-bit register has not been written. When clear, this bit specifies that this register has been written and may not be written again.
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 18: User Register 3 (USER\_REG3), offset 0x1EC**

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

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

**User Register 3 (USER\_REG3)**

Base 0x400F.E000

Offset 0x1EC

Type R/W, reset 0xFFFF.FFFF

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

Bit/Field	Name	Type	Reset	Description
31	NW	R/W	1	Not Written. When set, this bit specifies that this 32-bit register has not been written. When clear, this bit specifies that this register has been written and may not be written again.
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 19: Flash Memory Protection Read Enable 1 (FMPRE1), offset 0x204

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

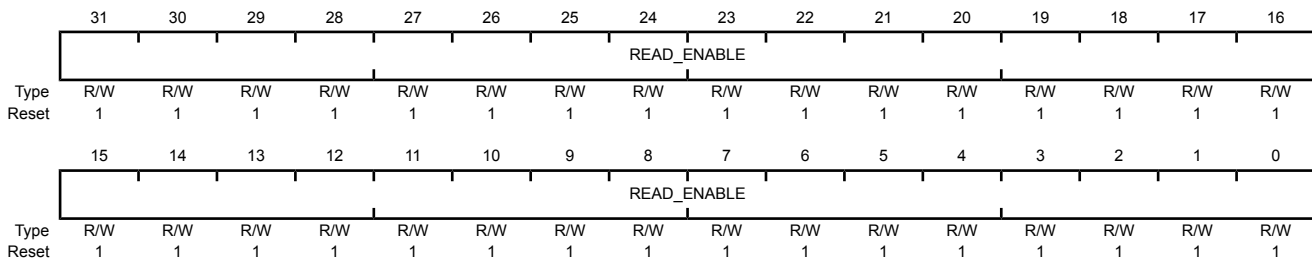
This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). This register is loaded during the power-on reset sequence. The factory settings for the **FMPREN** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves 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 memory blocks to be executed or read. The policies may be combined as shown in the table "Flash Protection Policy Combinations".
	Value	Description		
	0xFFFFFFFF	Enables 128 KB of Flash memory.		

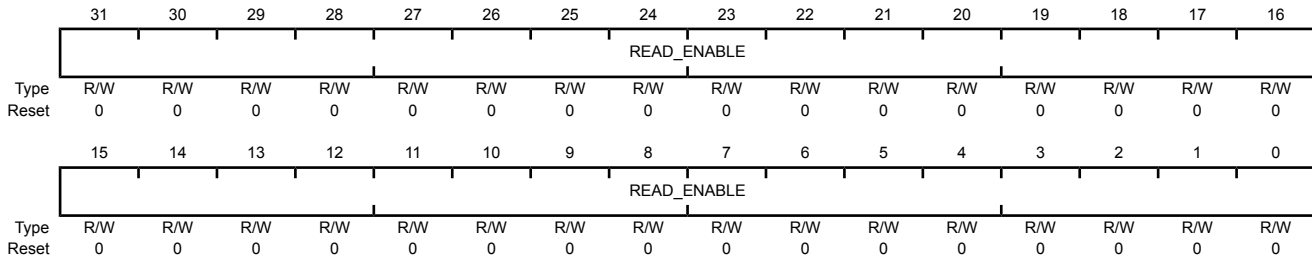
## Register 20: Flash Memory Protection Read Enable 2 (FMPRE2), offset 0x208

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

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

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

Base 0x400F.E000  
 Offset 0x208  
 Type R/W, reset 0x0000.0000



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

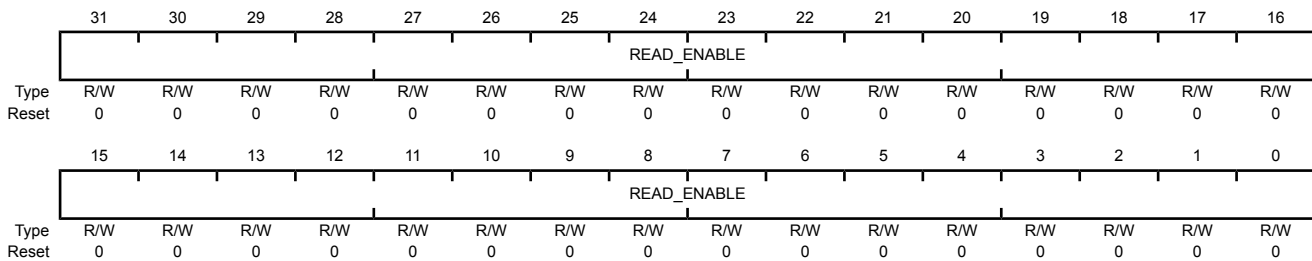
### Register 21: Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C

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

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

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

Base 0x400F.E000  
 Offset 0x20C  
 Type R/W, reset 0x0000.0000



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

## Register 22: Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404

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

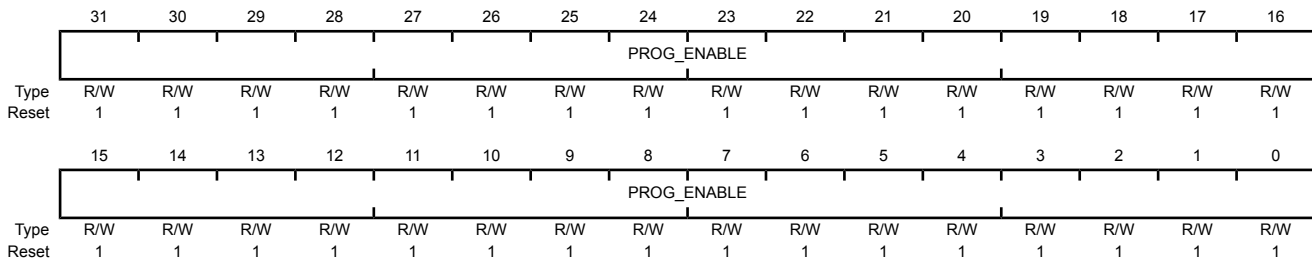
This register stores the execute-only protection bits for each 2-KB flash block (**FMPPE<sub>n</sub>** stores the execute-only bits). This register is loaded during the power-on reset sequence. The factory settings for the **FMPPE<sub>n</sub>** and **FMPPE<sub>n</sub>** registers are a value of 1 for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

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

Base 0x400F.E000

Offset 0x404

Type R/W, reset 0xFFFF.FFFF



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

Value	Description
0xFFFFFFFF	Enables 128 KB of Flash memory.

## Register 23: Flash Memory Protection Program Enable 2 (FMPPE2), offset 0x408

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

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

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

Base 0x400F.E000

Offset 0x408

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0x00000000	Flash Programming Enable

Value	Description
0x00000000	Enables 128 KB of Flash memory.

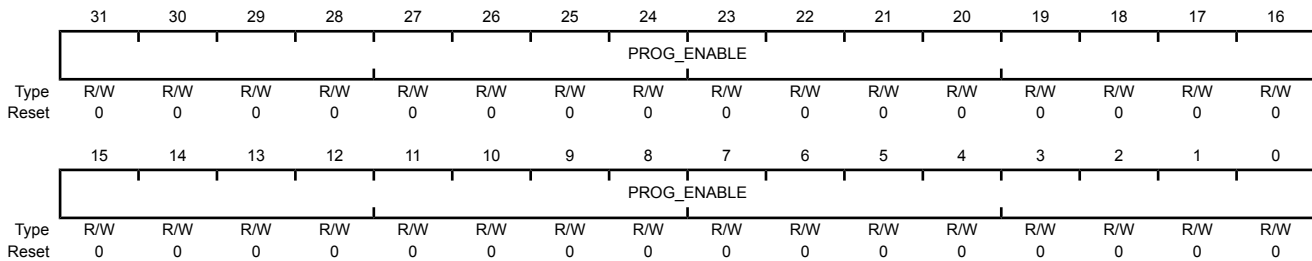
## Register 24: Flash Memory Protection Program Enable 3 (FMPPE3), offset 0x40C

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

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

Base 0x400F.E000  
 Offset 0x40C  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0x00000000	Flash Programming Enable

Value	Description
0x00000000	Enables 128 KB of Flash memory.

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

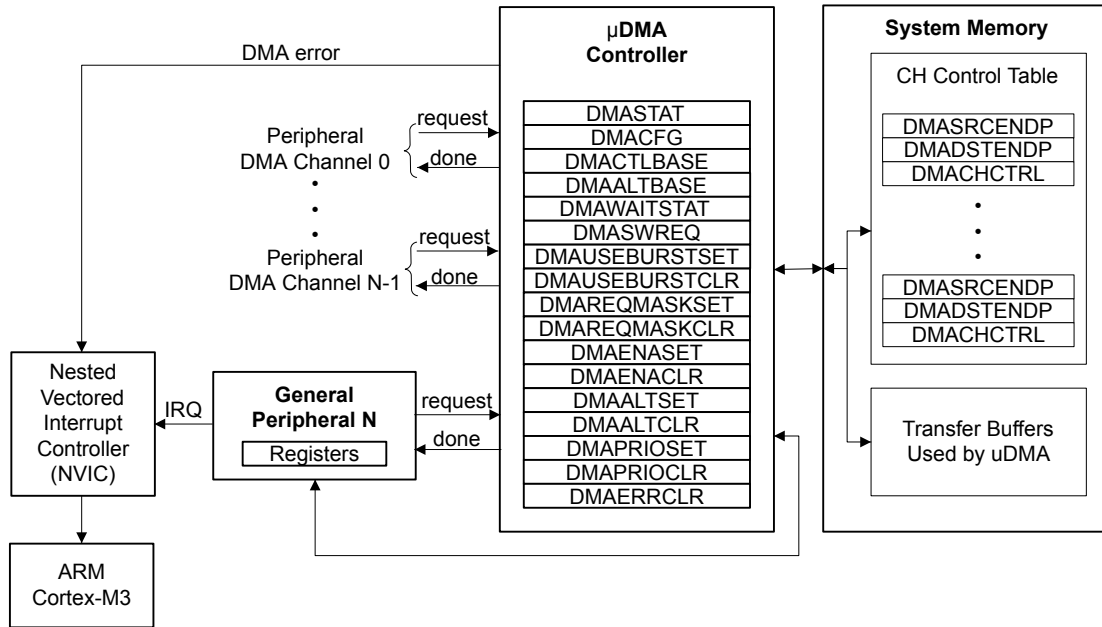
The LM3S5791 microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA ( $\mu$ DMA). The  $\mu$ DMA controller provides a way to offload data transfer tasks from the Cortex-M3 processor, allowing for more efficient use of the processor and the expanded available bus bandwidth. The  $\mu$ DMA controller can perform transfers between memory and peripherals. It has dedicated channels for each supported on-chip module and can be programmed to automatically perform transfers between peripherals and memory as the peripheral is ready to transfer more data. The  $\mu$ DMA controller provides the following features:

- ARM PrimeCell® 32-channel configurable  $\mu$ DMA controller
- Support for multiple transfer modes
  - Memory-to-memory, memory-to-peripheral, peripheral-to-memory
  - Basic for simple transfer scenarios
  - Ping-pong for continuous data flow
  - Scatter-gather for a programmable list of arbitrary transfers initiated from a single request
- Highly flexible and configurable channel operation
  - Independently configured and operated channels
  - Dedicated channels for supported on-chip modules - USB, UART, GP Timer, ADC, EPI, SSI, I<sup>2</sup>S
  - Alternate channel assignments
  - One channel each for receive and transmit path for bidirectional modules
  - Dedicated channel for software-initiated transfers
  - Per-channel configurable bus arbitration scheme
  - Optional software-initiated requests for any channel
- Two levels of priority
- Design optimizations for improved bus access performance between  $\mu$ DMA controller and the processor core
  - $\mu$ DMA controller access is subordinate to core access
  - RAM striping
  - Peripheral bus segmentation
- Data sizes of 8, 16, and 32 bits
- Transfer size is programmable in binary steps from 1 to 1024
- Source and destination address increment size of byte, half-word, word, or no increment

- Maskable device requests
- Interrupt on transfer completion, with a separate interrupt per channel

## 8.1 Block Diagram

Figure 8-1.  $\mu$ DMA Block Diagram



## 8.2 Functional Description

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

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

Each channel also has a configurable arbitration size. The arbitration size is the number of items that are transferred in a burst before the  $\mu$ DMA controller re-arbitrates for channel priority. Using the



arbitration size, it is possible to control exactly how many items are transferred to or from a peripheral each time it makes a DMA service request.

## 8.2.1 Channel Assignments

$\mu$ DMA channels 0-31 are assigned to peripherals according to the following table. The **DMA Channel Alternate Select (DMACHALT)** register (see page 273) can be used to specify the alternate assignment.

**Note:** Channels noted in the table as "Available for software" may be assigned to peripherals in the future. However, they are currently available for software use. Channel 30 is dedicated for software use.

**Table 8-1. DMA Channel Assignments**

DMA Channel	Peripheral Assigned	Alternate Assignment
0	USB Endpoint 1 Receive	UART2 Receive
1	USB Endpoint 1 Transmit	UART2 Transmit
2	USB Endpoint 2 Receive	General-Purpose Timer 3A
3	USB Endpoint 2 Transmit	General-Purpose Timer 3B
4	USB Endpoint 3 Receive	General-Purpose Timer 2A
5	USB Endpoint 3 Transmit	General-Purpose Timer 2B
6	Available for software	General-Purpose Timer 2A
7	Available for software	General-Purpose Timer 2B
8	UART0 Receive	SSI1 Receive
9	UART0 Transmit	SSI1 Transmit
10	SSI0 Receive	UART1 Receive
11	SSI0 Transmit	UART1 Transmit
12	Available for software	UART2 Receive
13	Available for software	UART2 Transmit
14	ADC0 Sample Sequencer 0	General-Purpose Timer 2A
15	ADC0 Sample Sequencer 1	General-Purpose Timer 2B
16	ADC0 Sample Sequencer 2	Available for software
17	ADC0 Sample Sequencer 3	Available for software
18	General-Purpose Timer 0A	General-Purpose Timer 1A
19	General-Purpose Timer 0B	General-Purpose Timer 1B
20	General-Purpose Timer 1A	EPI0 Receive
21	General-Purpose Timer 1B	EPI0 Transmit
22	UART1 Receive	Available for software
23	UART1 Transmit	Available for software
24	SSI1 Receive	ADC1 Sample Sequencer 0
25	SSI1 Transmit	ADC1 Sample Sequencer 1
26	Available for software	ADC1 Sample Sequencer 2
27	Available for software	ADC1 Sample Sequencer 3
28	I <sup>2</sup> S0 Receive	Available for software
29	I <sup>2</sup> S0 Transmit	Available for software
30	Dedicated for software use	
31	Reserved	

## 8.2.2 Priority

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

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

## 8.2.3 Arbitration Size

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

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

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

## 8.2.4 Request Types

The  $\mu$ DMA controller responds to two types of requests from a peripheral: single or burst. Each peripheral may support either or both types of requests. A single request means that the peripheral is ready to transfer one item, while a burst request means that the peripheral is ready to transfer multiple items.

The  $\mu$ DMA controller responds differently depending on whether the peripheral is making a single request or a burst request. If both are asserted, and the  $\mu$ DMA channel has been set up for a burst transfer, then the burst request takes precedence. See Table 8-2, which shows how each peripheral supports the two request types.

**Table 8-2. Request Type Support**

Peripheral	Single Request Signal	Burst Request Signal
USB TX	None	FIFO TXRDY
USB RX	None	FIFO RXRDY
UART TX	TX FIFO Not Full	TX FIFO Level (configurable)
UART RX	RX FIFO Not Empty	RX FIFO Level (configurable)
SSI TX	TX FIFO Not Full	TX FIFO Level (fixed at 4)
SSI RX	RX FIFO Not Empty	RX FIFO Level (fixed at 4)
ADC	None	Sequencer IE bit

Peripheral	Single Request Signal	Burst Request Signal
General-Purpose Timer	Raw interrupt pulse	None
I <sup>2</sup> S TX	None	FIFO service request
I <sup>2</sup> S RX	None	FIFO service request
EPI TX	None	TX FIFO not full
EPI RX	None	RX FIFO not empty

#### 8.2.4.1 Single Request

When a single request is detected, and not a burst request, the  $\mu$ DMA controller transfers one item and then stops to wait for another request.

#### 8.2.4.2 Burst Request

When a burst request is detected, the  $\mu$ DMA controller transfers the number of items that is the lesser of the arbitration size or the number of items remaining in the transfer. Therefore, the arbitration size should be the same as the number of data items that the peripheral can accommodate when making a burst request. For example, the UART generates a burst request based on the FIFO trigger level. In this case, the arbitration size should be set to the amount of data that the FIFO can transfer when the trigger level is reached.

It may be desirable to use only burst transfers and not allow single transfers. For example, perhaps the nature of the data is such that it only makes sense when transferred together as a single unit rather than one piece at a time. The single request can be disabled by using the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register. By setting the bit for a channel in this register, the  $\mu$ DMA controller only responds to burst requests for that channel.

### 8.2.5 Channel Configuration

The  $\mu$ DMA controller uses an area of system memory to store a set of channel control structures in a table. The control table may have one or two entries for each DMA channel. Each entry in the table structure contains source and destination pointers, transfer size, and transfer mode. The control table can be located anywhere in system memory, but it must be contiguous and aligned on a 1024-byte boundary.

Table 8-3 on page 223 shows the layout in memory of the channel control table. Each channel may have one or two control structures in the control table: a primary control structure and an optional alternate control structure. The table is organized so that all of the primary entries are in the first half of the table, and all the alternate structures are in the second half of the table. The primary entry is used for simple transfer modes where transfers can be reconfigured and restarted after each transfer is complete. In this case, the alternate control structures are not used and therefore only the first half of the table must be allocated in memory; the second half of the control table is not necessary, and that memory can be used for something else. If a more complex transfer mode is used such as ping-pong or scatter-gather, then the alternate control structure is also used and memory space should be allocated for the entire table.

Any unused memory in the control table may be used by the application. This includes the control structures for any channels that are unused by the application as well as the unused control word for each channel.

**Table 8-3. Control Structure Memory Map**

Offset	Channel
0x0	0, Primary

Offset	Channel
0x10	1, Primary
...	...
0x1F0	31, Primary
0x200	0, Alternate
0x210	1, Alternate
...	...
0x3F0	31, Alternate

Table 8-4 shows an individual control structure entry in the control table. Each entry has a source and destination end pointer. These pointers point to the ending address of the transfer and are inclusive. If the source or destination is non-incrementing (as for a peripheral register), then the pointer should point to the transfer address.

**Table 8-4. Channel Control Structure**

Offset	Description
0x000	Source End Pointer
0x004	Destination End Pointer
0x008	Control Word
0x00C	Unused

The remaining part of the control structure is the control word. The control word contains the following fields:

- Source and destination data sizes
- Source and destination address increment size
- Number of transfers before bus arbitration
- Total number of items to transfer
- Useburst flag
- Transfer mode

The control word and each field are described in detail in “ $\mu$ DMA Channel Control Structure” on page 241. The  $\mu$ DMA controller updates the transfer size and transfer mode fields as the transfer is performed. At the end of a transfer, the transfer size indicates 0, and the transfer mode will indicate "stopped." Because the control word is modified by the  $\mu$ DMA controller, it must be reconfigured before each new transfer. The source and destination end pointers are not modified, so they can be left unchanged if the source or destination addresses remain the same.

Prior to starting a transfer, a  $\mu$ DMA channel must be enabled by setting the appropriate bit in the **DMA Channel Enable Set (DMAENASET)** register. A channel can be disabled by setting the channel bit in the **DMA Channel Enable Clear (DMAENACLR)** register. At the end of a complete DMA transfer, the controller automatically disables the channel.

## 8.2.6 Transfer Modes

The  $\mu$ DMA controller supports several transfer modes. Two of the modes support simple one-time transfers. Several complex modes support a continuous flow of data.

### 8.2.6.1 Stop Mode

While Stop is not actually a transfer mode, it is a valid value for the mode field of the control word. When the mode field has this value, the  $\mu$ DMA controller does not perform any transfers and disables the channel if it is enabled. At the end of a transfer, the  $\mu$ DMA controller updates the control word to set the mode to Stop.

### 8.2.6.2 Basic Mode

In Basic mode, the  $\mu$ DMA controller performs transfers as long as there are more items to transfer, and a transfer request is present. This mode is used with peripherals that assert a DMA request signal whenever the peripheral is ready for a data transfer. Basic mode should not be used in any situation where the request is momentary even though the entire transfer should be completed. For example, a software-initiated transfer creates a momentary request, and in Basic mode, only one item is transferred on a software request.

When all of the items have been transferred using Basic mode, the  $\mu$ DMA controller sets the mode for that channel to Stop.

### 8.2.6.3 Auto Mode

Auto mode is similar to Basic mode, except that once a transfer request is received, the transfer runs to completion, even if the DMA request is removed. This mode is suitable for software-triggered transfers. Generally, Auto mode is not used with a peripheral.

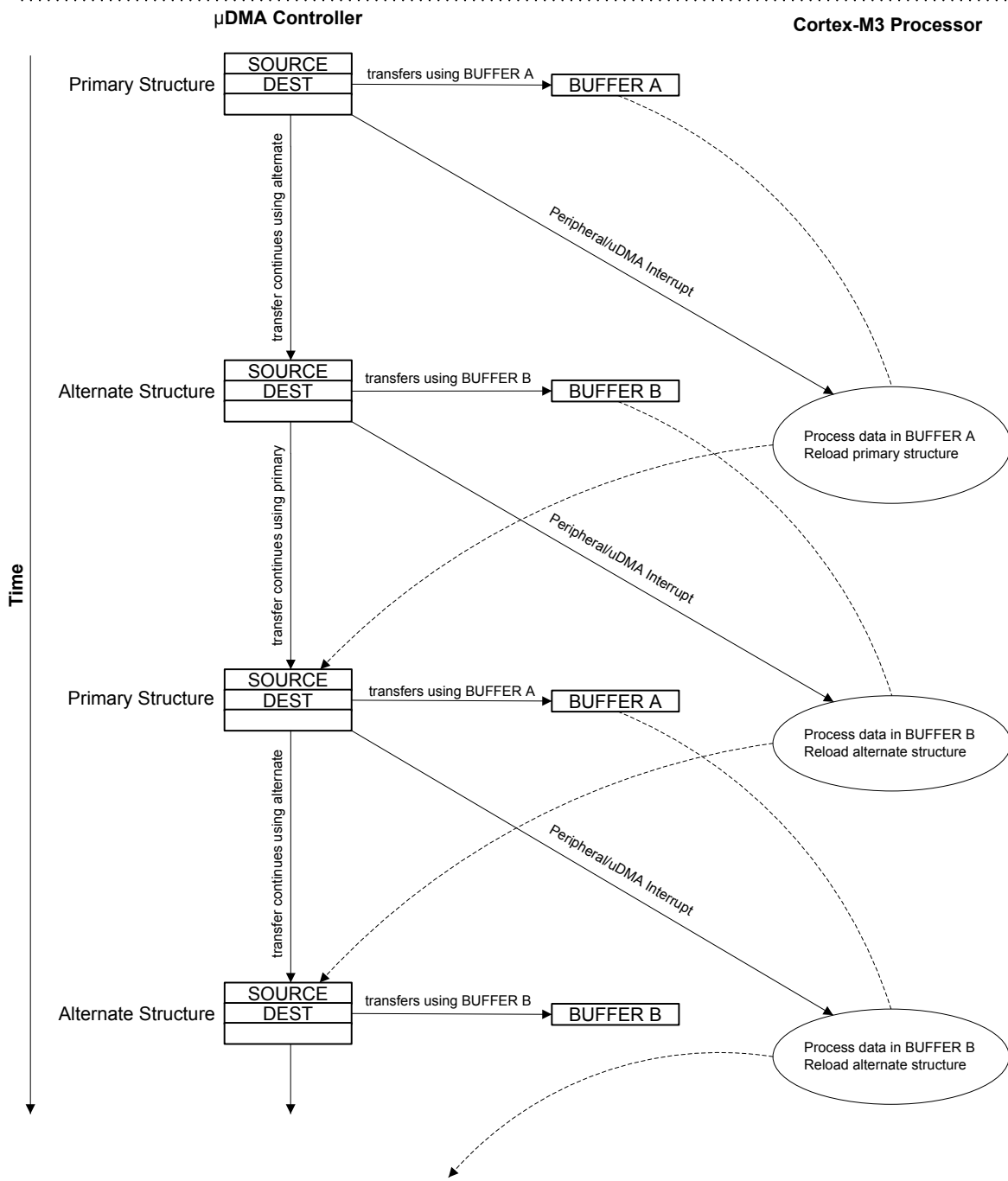
When all the items have been transferred using Auto mode, the  $\mu$ DMA controller sets the mode for that channel to Stop.

### 8.2.6.4 Ping-Pong

Ping-Pong mode is used to support a continuous data flow to or from a peripheral. To use Ping-Pong mode, both the primary and alternate data structures must be implemented. Both structures are set up by the processor for data transfer between memory and a peripheral. The transfer is started using the primary control structure. When the transfer using the primary control structure is complete, the  $\mu$ DMA controller reads the alternate control structure for that channel to continue the transfer. Each time this happens, an interrupt is generated, and the processor can reload the control structure for the just-completed transfer. Data flow can continue indefinitely this way, using the primary and alternate control structures to switch back and forth between buffers as the data flows to or from the peripheral.

Refer to Figure 8-2 for an example showing operation in Ping-Pong mode.

Figure 8-2. Example of Ping-Pong DMA Transaction



### 8.2.6.5 Memory Scatter-Gather

Memory Scatter-Gather mode is a complex mode used when data must be transferred to or from varied locations in memory instead of a set of contiguous locations in a memory buffer. For example, a gather DMA operation could be used to selectively read the payload of several stored packets of a communication protocol and store them together in sequence in a memory buffer.

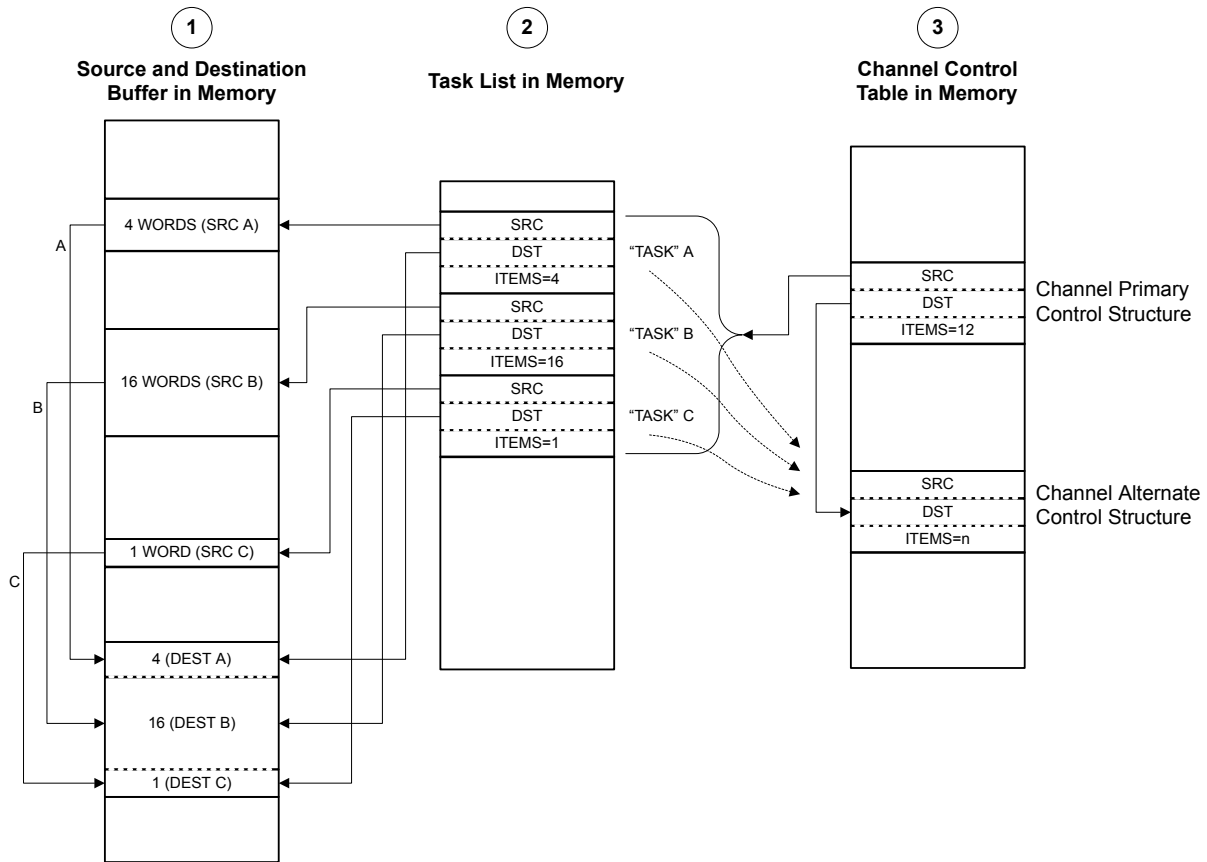
In Memory Scatter-Gather mode, the primary control structure is used to program the alternate control structure from a table in memory. The table is set up by the processor software and contains a list of control structures, each containing the source and destination end pointers, and the control word for a specific transfer. The mode of each control word must be set to Scatter-Gather mode. Each entry in the table is copied in turn to the alternate structure where it is then executed. The  $\mu$ DMA controller alternates between using the primary control structure to copy the next transfer instruction from the list and then executing the new transfer instruction. The end of the list is marked by programming the control word for the last entry to use Basic transfer mode. Once the last transfer is performed using Basic mode, the  $\mu$ DMA controller stops. A completion interrupt is generated only after the last transfer. It is possible to loop the list by having the last entry copy the primary control structure to point back to the beginning of the list (or to a new list). It is also possible to trigger a set of other channels to perform a transfer, either directly, by programming a write to the software trigger for another channel, or indirectly, by causing a peripheral action that results in a  $\mu$ DMA request.

By programming the  $\mu$ DMA controller using this method, a set of arbitrary transfers can be performed based on a single DMA request.

Refer to Figure 8-3 on page 228 and Figure 8-4 on page 229, which show an example of operation in Memory Scatter-Gather mode. This example shows a *gather* operation, where data in three separate buffers in memory is copied together into one buffer. Figure 8-3 on page 228 shows how the application sets up a  $\mu$ DMA task list in memory that is used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel that is used for the operation is configured to copy from the task list to the alternate control structure.

Figure 8-4 on page 229 shows the sequence as the  $\mu$ DMA controller performs the three sets of copy operations. First, using the primary control structure, the  $\mu$ DMA controller loads the alternate control structure with task A. It then performs the copy operation specified by task A, copying the data from the source buffer A to the destination buffer. Next, the  $\mu$ DMA controller again uses the primary control structure to load task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for task C.

Figure 8-3. Memory Scatter-Gather, Setup and Configuration

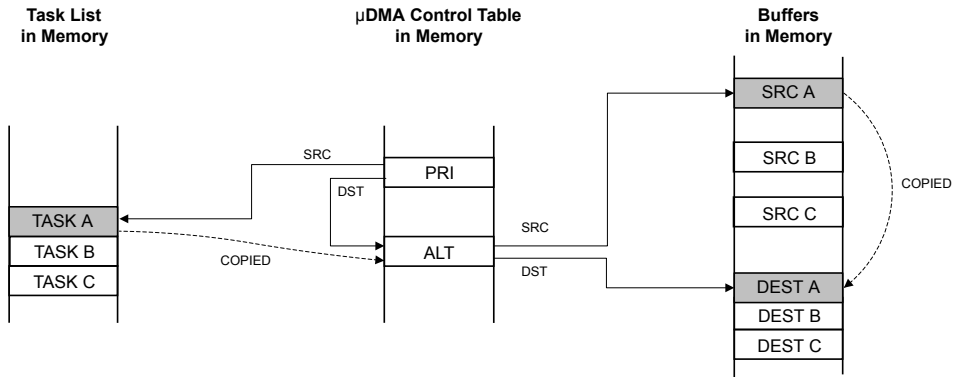


NOTES:

1. Application has a need to copy data items from three separate location in memory into one combined buffer.
2. Application sets up  $\mu$ DMA "task list" in memory, which contains the pointers and control configuration for three  $\mu$ DMA copy "tasks."
3. Application sets up the channel primary control structure to copy each task configuration, one at a time, to the alternate control structure, where it will be executed by the  $\mu$ DMA controller.

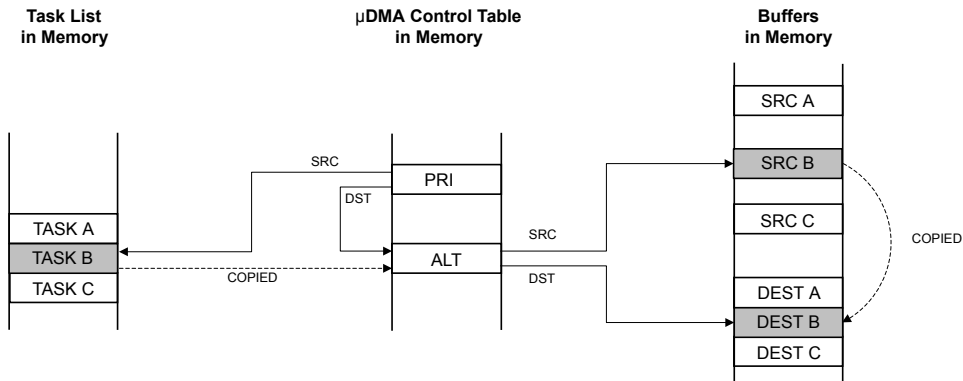


Figure 8-4. Memory Scatter-Gather,  $\mu$ DMA Copy Sequence



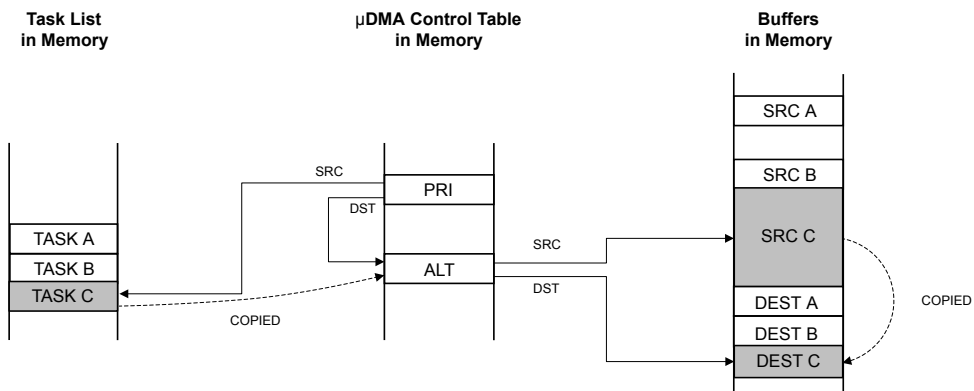
Using the channel's primary control structure, the  $\mu$ DMA controller copies task A configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the  $\mu$ DMA controller copies data from the source buffer A to the destination buffer.



Using the channel's primary control structure, the  $\mu$ DMA controller copies task B configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the  $\mu$ DMA controller copies data from the source buffer B to the destination buffer.



Using the channel's primary control structure, the  $\mu$ DMA controller copies task C configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the  $\mu$ DMA controller copies data from the source buffer C to the destination buffer.

### 8.2.6.6 Peripheral Scatter-Gather

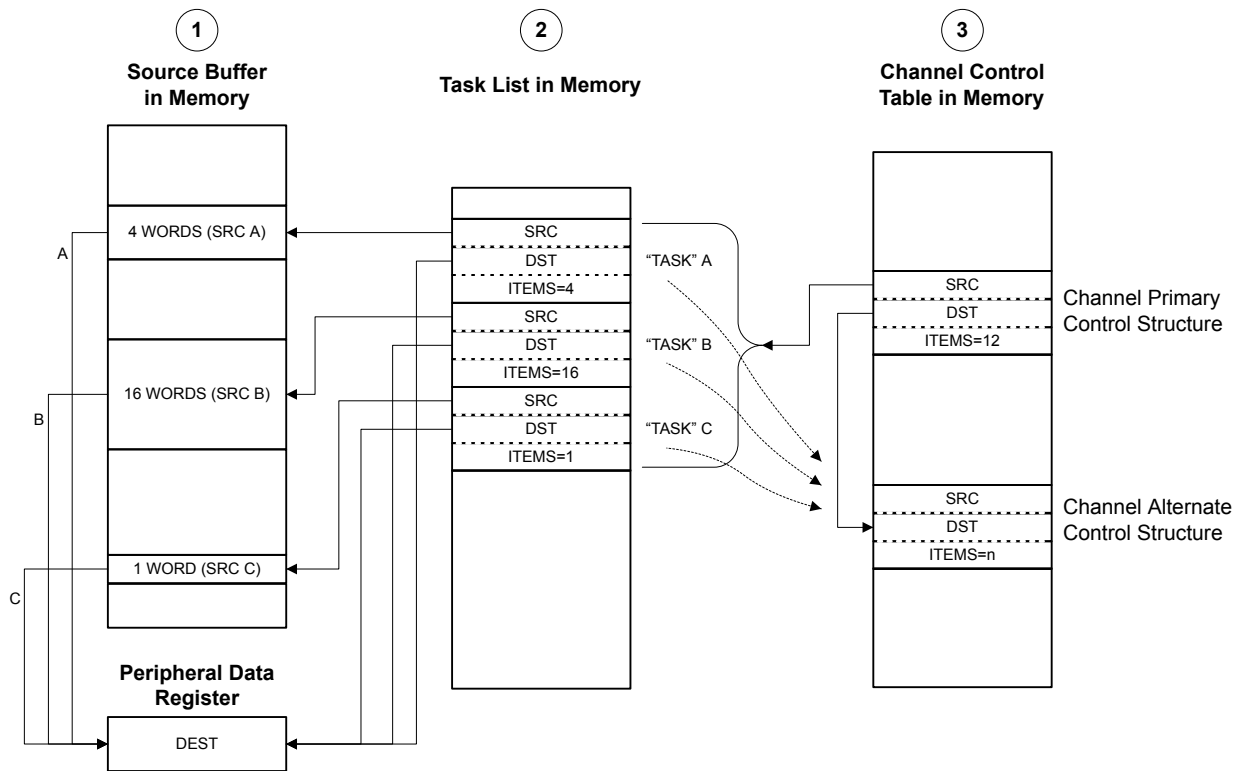
Peripheral Scatter-Gather mode is very similar to Memory Scatter-Gather, except that the transfers are controlled by a peripheral making a DMA request. Upon detecting a DMA request from the peripheral, the  $\mu$ DMA controller uses the primary control structure to copy one entry from the list to the alternate control structure and then performs the transfer. At the end of this transfer, the next transfer is started only if the peripheral again asserts a DMA request. The  $\mu$ DMA controller continues to perform transfers from the list only when the peripheral is making a request, until the last transfer is complete. A completion interrupt is generated only after the last transfer.

By using this method, the  $\mu$ DMA controller can transfer data to or from a peripheral from a set of arbitrary locations whenever the peripheral is ready to transfer data.

Refer to Figure 8-5 on page 231 and Figure 8-6 on page 232, which show an example of operation in Peripheral Scatter-Gather mode. This example shows a gather operation, where data from three separate buffers in memory is copied to a single peripheral data register. Figure 8-5 on page 231 shows how the application sets up a  $\mu$ DMA task list in memory that is used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel that is used for the operation is configured to copy from the task list to the alternate control structure.

Figure 8-6 on page 232 shows the sequence as the  $\mu$ DMA controller performs the three sets of copy operations. First, using the primary control structure, the  $\mu$ DMA controller loads the alternate control structure with task A. It then performs the copy operation specified by task A, copying the data from the source buffer A to the peripheral data register. Next, the  $\mu$ DMA controller again uses the primary control structure to load task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for task C.

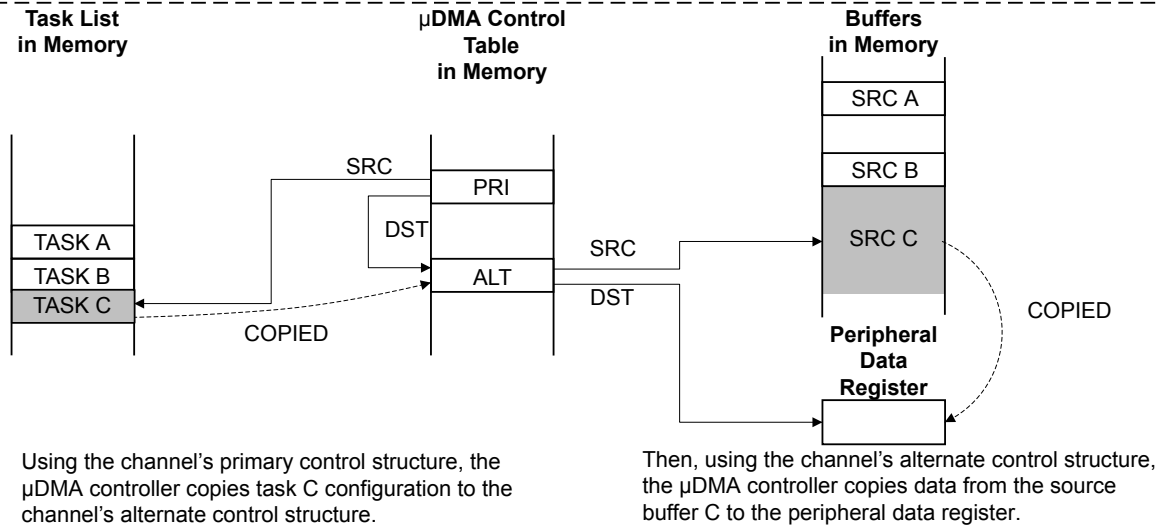
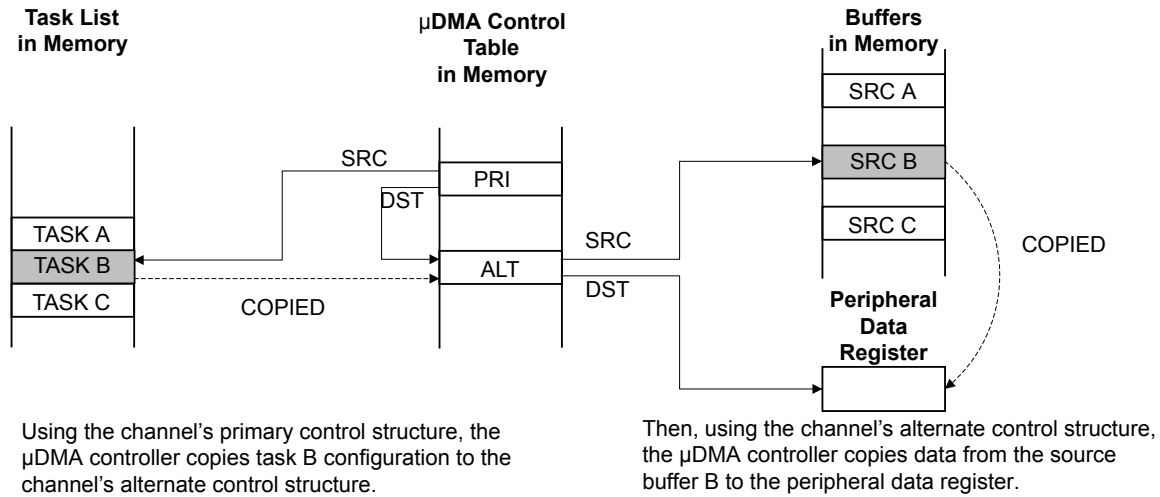
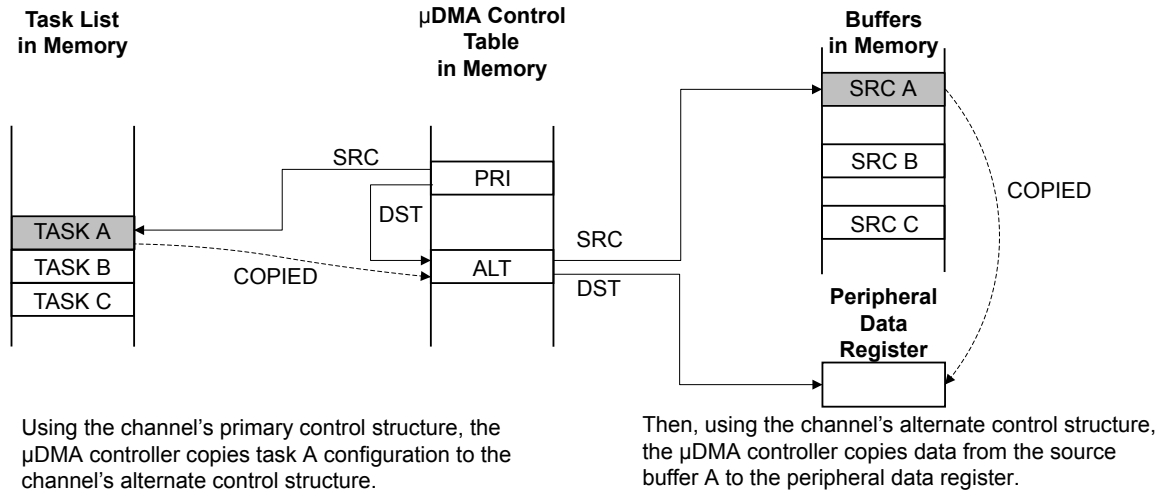
Figure 8-5. Peripheral Scatter-Gather, Setup and Configuration



## NOTES:

1. Application has a need to copy data items from three separate location in memory into a peripheral data register.
2. Application sets up  $\mu$ DMA "task list" in memory, which contains the pointers and control configuration for three  $\mu$ DMA copy "tasks."
3. Application sets up the channel primary control structure to copy each task configuration, one at a time, to the alternate control structure, where it will be executed by the  $\mu$ DMA controller.

Figure 8-6. Peripheral Scatter-Gather,  $\mu$ DMA Copy Sequence



## 8.2.7 Transfer Size and Increment

The  $\mu$ DMA controller supports transfer data sizes of 8, 16, or 32 bits. The source and destination data size must be the same for any given transfer. The source and destination address can be auto-incremented by bytes, half-words, or words, or can be set to no increment. The source and destination address increment values can be set independently, and it is not necessary for the address increment to match the data size as long as the increment is the same or larger than the data size. For example, it is possible to perform a transfer using 8-bit data size, but using an address increment of full words (4 bytes). The data to be transferred must be aligned in memory according to the data size (8, 16, or 32 bits).

Table 8-5 shows the configuration to read from a peripheral that supplies 8-bit data.

**Table 8-5.  $\mu$ DMA Read Example: 8-Bit Peripheral**

Field	Configuration
Source data size	8 bits
Destination data size	8 bits
Source address increment	No increment
Destination address increment	Byte
Source end pointer	Peripheral read FIFO register
Destination end pointer	End of the data buffer in memory

## 8.2.8 Peripheral Interface

Each peripheral that supports  $\mu$ DMA has a DMA single request and/or burst request signal that is asserted when the device is ready to transfer data (see Table 8-2 on page 222). The request signal can be disabled or enabled using the **DMA Channel Request Mask Set (DMAREQMASKSET)** and **DMA Channel Request Mask Clear (DMAREQMASKCLR)** registers. The DMA request signal is disabled, or masked, when the channel request mask bit is set. When the request is not masked, the DMA channel is configured correctly and enabled, and the peripheral asserts the DMA request signal, the  $\mu$ DMA controller begins the transfer.

When a DMA transfer is complete, the  $\mu$ DMA controller asserts a DMA Done signal, which is routed through the interrupt vector of the peripheral. Therefore, if DMA is used to transfer data for a peripheral and interrupts are used, then the interrupt handler for that peripheral must be designed to handle the  $\mu$ DMA transfer completion interrupt. When DMA is enabled for a peripheral, the  $\mu$ DMA controller masks the normal interrupts for a peripheral. Thus, when a large amount of data is transferred using DMA, instead of receiving multiple interrupts from the peripheral as data flows, the processor receives only one interrupt when the transfer is complete.

The interrupt request from the  $\mu$ DMA controller is automatically cleared when the interrupt handler is activated.

## 8.2.9 Software Request

One  $\mu$ DMA channel is dedicated to software-initiated transfers. This channel also has a dedicated interrupt to signal completion of a DMA transfer. A transfer is initiated by software by first configuring and enabling the transfer, and then issuing a software request using the **DMA Channel Software Request (DMASWREQ)** register. For software-based transfers, the Auto transfer mode should be used.

It is possible to initiate a transfer on any channel using the **DMASWREQ** register. If a request is initiated by software using a peripheral DMA channel, then the completion interrupt occurs on the

interrupt vector for the peripheral instead of the software interrupt vector. Any channel may be used for software requests as long as the corresponding peripheral is not using  $\mu$ DMA for data transfer.

## 8.2.10 Interrupts and Errors

When a  $\mu$ DMA channel generates an interrupt, the interrupt status is latched in the **DMA Channel Interrupt Status (DMACHIS)** register (see page 274). This register can be used by the peripheral interrupt handler code to determine if the interrupt was caused by the  $\mu$ DMA channel or something else.

When a DMA transfer is complete, the  $\mu$ DMA controller generates a completion interrupt on the interrupt vector of the peripheral. If the transfer uses the software DMA channel, then the completion interrupt occurs on the dedicated software DMA interrupt vector.

If the  $\mu$ DMA controller encounters a bus or memory protection error as it attempts to perform a data transfer, it disables the DMA channel that caused the error and generates an interrupt on the  $\mu$ DMA Error interrupt vector. The processor can read the **DMA Bus Error Clear (DMAERRCLR)** register to determine if an error is pending. The `ERRCLR` bit is set if an error occurred. The error can be cleared by writing a 1 to the `ERRCLR` bit.

If the peripheral generates an error that causes an interrupt, the interrupt is generated on the interrupt vector for that peripheral. This is the same whether or not  $\mu$ DMA is being used with the peripheral.

Table 8-6 shows the dedicated interrupt assignments for the  $\mu$ DMA controller.

**Table 8-6.  $\mu$ DMA Interrupt Assignments**

Interrupt	Assignment
46	$\mu$ DMA Software Channel Transfer
47	$\mu$ DMA Error

## 8.3 Initialization and Configuration

### 8.3.1 Module Initialization

Before the  $\mu$ DMA controller can be used, it must be enabled in the System Control block and in the peripheral. The location of the channel control structure must also be programmed.

The following steps should be performed one time during system initialization:

1. The  $\mu$ DMA peripheral must be enabled in the System Control block. To do this, set the `UDMA` bit of the System Control **RCGC2** register (page 175).
2. Enable the  $\mu$ DMA controller by setting the `MASTEREN` bit of the **DMA Configuration (DMACFG)** register.
3. Program the location of the channel control table by writing the base address of the table to the **DMA Channel Control Base Pointer (DMACTLBASE)** register. The base address must be aligned on a 1024-byte boundary.

### 8.3.2 Configuring a Memory-to-Memory Transfer

$\mu$ DMA channel 30 is dedicated for software-initiated transfers. However, any channel can be used for software-initiated, memory-to-memory transfer if the associated peripheral is not being used.

### 8.3.2.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Program bit 30 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.
2. Set bit 30 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.
3. Set bit 30 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the  $\mu$ DMA controller to respond to single and burst requests.
4. Set bit 30 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the  $\mu$ DMA controller to recognize requests for this channel.

### 8.3.2.2 Configure the Channel Control Structure

Now the channel control structure must be configured.

This example transfers 256 words from one memory buffer to another. Channel 30 is used for a software transfer, and the control structure for channel 30 is at offset 0x1E0 of the channel control table. The channel control structure for channel 30 is located at the offsets shown in Table 8-7.

**Table 8-7. Channel Control Structure Offsets for Channel 30**

Offset	Description
Control Table Base + 0x1E0	Channel 30 Source End Pointer
Control Table Base + 0x1E4	Channel 30 Destination End Pointer
Control Table Base + 0x1E8	Channel 30 Control Word

#### *Configure the Source and Destination*

The source and destination end pointers must be set to the last address for the transfer (inclusive).

1. Program the source end pointer at offset 0x1E0 to the address of the source buffer + 0x3FC.
2. Program the destination end pointer at offset 0x1E4 to the address of the destination buffer + 0x3FC.

The control word at offset 0x1E8 must be programmed according to Table 8-8.

**Table 8-8. Channel Control Word Configuration for Memory Transfer Example**

Field in DMACHCTL	Bits	Value	Description
DSTINC	31:30	2	32-bit destination address increment
DSTSIZE	29:28	2	32-bit destination data size
SRCINC	27:26	2	32-bit source address increment
SRCSIZE	25:24	2	32-bit source data size
reserved	23:18	0	Reserved
ARBSIZE	17:14	3	Arbitrates after 8 transfers
XFERSIZE	13:4	255	Transfer 256 items
NXTUSEBURST	3	0	N/A for this transfer type
XFERMODE	2:0	2	Use Auto-request transfer mode

### 8.3.2.3 Start the Transfer

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 30 of the **DMA Channel Enable Set (DMAENASET)** register.
2. Issue a transfer request by setting bit 30 of the **DMA Channel Software Request (DMASWREQ)** register.

The DMA transfer begins. If the interrupt is enabled, then the processor is notified by interrupt when the transfer is complete. If needed, the status can be checked by reading bit 30 of the **DMAENASET** register. This bit is automatically cleared when the transfer is complete. The status can also be checked by reading the **XFERMODE** field of the channel control word at offset 0x1E8. This field is automatically cleared at the end of the transfer.

### 8.3.3 Configuring a Peripheral for Simple Transmit

This example configures the  $\mu$ DMA controller to transmit a buffer of data to a peripheral. The peripheral has a transmit FIFO with a trigger level of 4. The example peripheral uses  $\mu$ DMA channel 7.

#### 8.3.3.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Configure bit 7 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.
2. Set bit 7 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.
3. Set bit 7 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the  $\mu$ DMA controller to respond to single and burst requests.
4. Set bit 7 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the  $\mu$ DMA controller to recognize requests for this channel.

#### 8.3.3.2 Configure the Channel Control Structure

This example transfers 64 bytes from a memory buffer to the peripheral's transmit FIFO register using  $\mu$ DMA channel 7. The control structure for channel 7 is at offset 0x070 of the channel control table. The channel control structure for channel 7 is located at the offsets shown in Table 8-9.

**Table 8-9. Channel Control Structure Offsets for Channel 7**

Offset	Description
Control Table Base + 0x070	Channel 7 Source End Pointer
Control Table Base + 0x074	Channel 7 Destination End Pointer
Control Table Base + 0x078	Channel 7 Control Word

#### **Configure the Source and Destination**

The source and destination end pointers must be set to the last address for the transfer (inclusive). Because the peripheral pointer does not change, it simply points to the peripheral's data register.

1. Program the source end pointer at offset 0x070 to the address of the source buffer + 0x3F.



2. Program the destination end pointer at offset 0x074 to the address of the peripheral's transmit FIFO register.

The control word at offset 0x078 must be programmed according to Table 8-10.

**Table 8-10. Channel Control Word Configuration for Peripheral Transmit Example**

Field in DMACHCTL	Bits	Value	Description
DSTINC	31:30	3	Destination address does not increment
DSTSIZE	29:28	0	8-bit destination data size
SRCINC	27:26	0	8-bit source address increment
SRCSIZE	25:24	0	8-bit source data size
reserved	23:18	0	Reserved
ARBSIZE	17:14	2	Arbitrates after 4 transfers
XFERSIZE	13:4	63	Transfer 64 items
NXTUSEBURST	3	0	N/A for this transfer type
XFERMODE	2:0	1	Use Basic transfer mode

**Note:** In this example, it is not important if the peripheral makes a single request or a burst request. Because the peripheral has a FIFO that triggers at a level of 4, the arbitration size is set to 4. If the peripheral does make a burst request, then 4 bytes are transferred, which is what the FIFO can accommodate. If the peripheral makes a single request (if there is any space in the FIFO), then one byte is transferred at a time. If it is important to the application that transfers only be made in bursts, then the Channel Useburst `SET[7]` bit should be set in the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register.

### 8.3.3.3 Start the Transfer

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 7 of the **DMA Channel Enable Set (DMAENASET)** register.

The  $\mu$ DMA controller is now configured for transfer on channel 7. The controller makes transfers to the peripheral whenever the peripheral asserts a DMA request. The transfers continue until the entire buffer of 64 bytes has been transferred. When that happens, the  $\mu$ DMA controller disables the channel and sets the `XFERMODE` field of the channel control word to 0 (Stopped). The status of the transfer can be checked by reading bit 7 of the **DMA Channel Enable Set (DMAENASET)** register. This bit is automatically cleared when the transfer is complete. The status can also be checked by reading the `XFERMODE` field of the channel control word at offset 0x078. This field is automatically cleared at the end of the transfer.

If peripheral interrupts are enabled, then the peripheral interrupt handler receives an interrupt when the entire transfer is complete.

## 8.3.4 Configuring a Peripheral for Ping-Pong Receive

This example configures the  $\mu$ DMA controller to continuously receive 8-bit data from a peripheral into a pair of 64-byte buffers. The peripheral has a receive FIFO with a trigger level of 8. The example peripheral uses  $\mu$ DMA channel 8.

### 8.3.4.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Configure bit 8 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.
2. Set bit 8 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.
3. Set bit 8 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the  $\mu$ DMA controller to respond to single and burst requests.
4. Set bit 8 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the  $\mu$ DMA controller to recognize requests for this channel.

### 8.3.4.2 Configure the Channel Control Structure

This example transfers bytes from the peripheral's receive FIFO register into two memory buffers of 64 bytes each. As data is received, when one buffer is full, the  $\mu$ DMA controller switches to use the other.

To use Ping-Pong buffering, both primary and alternate channel control structures must be used. The primary control structure for channel 8 is at offset 0x080 of the channel control table, and the alternate channel control structure is at offset 0x280. The channel control structures for channel 8 are located at the offsets shown in Table 8-11.

**Table 8-11. Primary and Alternate Channel Control Structure Offsets for Channel 8**

Offset	Description
Control Table Base + 0x080	Channel 8 Primary Source End Pointer
Control Table Base + 0x084	Channel 8 Primary Destination End Pointer
Control Table Base + 0x088	Channel 8 Primary Control Word
Control Table Base + 0x280	Channel 8 Alternate Source End Pointer
Control Table Base + 0x284	Channel 8 Alternate Destination End Pointer
Control Table Base + 0x288	Channel 8 Alternate Control Word

#### **Configure the Source and Destination**

The source and destination end pointers must be set to the last address for the transfer (inclusive). Because the peripheral pointer does not change, it simply points to the peripheral's data register. Both the primary and alternate sets of pointers must be configured.

1. Program the primary source end pointer at offset 0x080 to the address of the peripheral's receive buffer.
2. Program the primary destination end pointer at offset 0x084 to the address of ping-pong buffer A + 0x3F.
3. Program the alternate source end pointer at offset 0x280 to the address of the peripheral's receive buffer.
4. Program the alternate destination end pointer at offset 0x284 to the address of ping-pong buffer B + 0x3F.

The primary control word at offset 0x088 and the alternate control word at offset 0x288 must be programmed according to Table 8-10 on page 237. Both control words are initially programmed the same way.

1. Program the primary channel control word at offset 0x088 according to Table 8-12.
2. Program the alternate channel control word at offset 0x288 according to Table 8-12.

**Table 8-12. Channel Control Word Configuration for Peripheral Ping-Pong Receive Example**

Field in DMACHCTL	Bits	Value	Description
DSTINC	31:30	0	8-bit destination address increment
DSTSIZE	29:28	0	8-bit destination data size
SRCINC	27:26	3	Source address does not increment
SRCSIZE	25:24	0	8-bit source data size
reserved	23:18	0	Reserved
ARBSIZE	17:14	3	Arbitrates after 8 transfers
XFERSIZE	13:4	63	Transfer 64 items
NXTUSEBURST	3	0	N/A for this transfer type
XFERMODE	2:0	3	Use Ping-Pong transfer mode

**Note:** In this example, it is not important if the peripheral makes a single request or a burst request. Because the peripheral has a FIFO that triggers at a level of 8, the arbitration size is set to 8. If the peripheral does make a burst request, then 8 bytes are transferred, which is what the FIFO can accommodate. If the peripheral makes a single request (if there is any data in the FIFO), then one byte is transferred at a time. If it is important to the application that transfers only be made in bursts, then the Channel Useburst `SET[8]` bit should be set in the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register.

### 8.3.4.3 Configure the Peripheral Interrupt

An interrupt handler should be configured when using  $\mu$ DMA Ping-Pong mode, it is best to use an interrupt handler. However, the Ping-Pong mode can be configured without interrupts by polling. The interrupt handler is triggered after each buffer is complete.

1. Configure and enable an interrupt handler for the peripheral.

### 8.3.4.4 Enable the $\mu$ DMA Channel

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 8 of the **DMA Channel Enable Set (DMAENASET)** register.

### 8.3.4.5 Process Interrupts

The  $\mu$ DMA controller is now configured and enabled for transfer on channel 8. When the peripheral asserts the DMA request signal, the  $\mu$ DMA controller makes transfers into buffer A using the primary channel control structure. When the primary transfer to buffer A is complete, it switches to the alternate channel control structure and makes transfers into buffer B. At the same time, the primary channel control word mode field is configured to indicate Stopped, and an interrupt is

When an interrupt is triggered, the interrupt handler must determine which buffer is complete and process the data or set a flag that the data must be processed by non-interrupt buffer processing code. Then the next buffer transfer must be set up.

In the interrupt handler:

1. Read the primary channel control word at offset 0x088 and check the `XFERMODE` field. If the field is 0, this means buffer A is complete. If buffer A is complete, then:

- a. Process the newly received data in buffer A or signal the buffer processing code that buffer A has data available.
  - b. Reprogram the primary channel control word at offset 0x88 according to Table 8-12 on page 239.
2. Read the alternate channel control word at offset 0x288 and check the `XFERMODE` field. If the field is 0, this means buffer B is complete. If buffer B is complete, then:
    - a. Process the newly received data in buffer B or signal the buffer processing code that buffer B has data available.
    - b. Reprogram the alternate channel control word at offset 0x288 according to Table 8-12 on page 239.

### 8.3.5 Configuring Alternate Channels

Alternate peripherals can be assigned to each  $\mu$ DMA channel using the `DMACHALT` register. Each bit represents a  $\mu$ DMA channel. If the bit is set, then the alternate peripheral is used for the channel.

Refer to Table 8-1 on page 221 for alternate channel assignments.

For example, to use SS11 Receive on channel 8 instead of UART0, set bit 8 of the `DMACHALT` register to 1.

## 8.4 Register Map

Table 8-13 on page 240 lists the  $\mu$ DMA channel control structures and registers. The channel control structure shows the layout of one entry in the channel control table. The channel control table is located in system memory, and the location is determined by the application, that is, the base address is n/a (not applicable). In the table below, the offset for the channel control structures is the offset from the entry in the channel control table. See “Channel Configuration” on page 223 and Table 8-3 on page 223 for a description of how the entries in the channel control table are located in memory. The  $\mu$ DMA register addresses are given as a hexadecimal increment, relative to the  $\mu$ DMA base address of 0x400F.F000. Note that the  $\mu$ DMA module clock must be enabled before the registers can be programmed (see page 175).

**Table 8-13.  $\mu$ DMA Register Map**

Offset	Name	Type	Reset	Description	See page
<b><math>\mu</math>DMA Channel Control Structure</b>					
0x000	DMASRCENDP	R/W	-	DMA Channel Source Address End Pointer	242
0x004	DMADSTENDP	R/W	-	DMA Channel Destination Address End Pointer	243
0x008	DMACHCTL	R/W	-	DMA Channel Control Word	244
<b><math>\mu</math>DMA Registers</b>					
0x000	DMASTAT	RO	0x001F.0000	DMA Status	249
0x004	DMACFG	WO	-	DMA Configuration	251
0x008	DMACTLBASE	R/W	0x0000.0000	DMA Channel Control Base Pointer	252
0x00C	DMAALTBASE	RO	0x0000.0200	DMA Alternate Channel Control Base Pointer	253

Offset	Name	Type	Reset	Description	See page
0x010	DMAWAITSTAT	RO	0x0000.0000	DMA Channel Wait-on-Request Status	254
0x014	DMASWREQ	WO	-	DMA Channel Software Request	255
0x018	DMAUSEBURSTSET	R/W	0x0000.0000	DMA Channel Useburst Set	256
0x01C	DMAUSEBURSTCLR	WO	-	DMA Channel Useburst Clear	258
0x020	DMAREQMASKSET	R/W	0x0000.0000	DMA Channel Request Mask Set	259
0x024	DMAREQMASKCLR	WO	-	DMA Channel Request Mask Clear	261
0x028	DMAENASET	R/W	0x0000.0000	DMA Channel Enable Set	262
0x02C	DMAENACL	WO	-	DMA Channel Enable Clear	264
0x030	DMAALTSET	R/W	0x0000.0000	DMA Channel Primary Alternate Set	265
0x034	DMAALTCLR	WO	-	DMA Channel Primary Alternate Clear	267
0x038	DMAPRIOSET	R/W	0x0000.0000	DMA Channel Priority Set	268
0x03C	DMAPRIOCLR	WO	-	DMA Channel Priority Clear	270
0x04C	DMAERRCLR	R/W	0x0000.0000	DMA Bus Error Clear	271
0x500	DMACHALT	R/W	0x0000.0000	DMA Channel Alternate Select	273
0x504	DMACHIS	R/W1C	0x0000.0000	DMA Channel Interrupt Status	274
0xFD0	DMAPeriphID4	RO	0x0000.0004	DMA Peripheral Identification 4	279
0xFE0	DMAPeriphID0	RO	0x0000.0030	DMA Peripheral Identification 0	275
0xFE4	DMAPeriphID1	RO	0x0000.00B2	DMA Peripheral Identification 1	276
0xFE8	DMAPeriphID2	RO	0x0000.000B	DMA Peripheral Identification 2	277
0xFEC	DMAPeriphID3	RO	0x0000.0000	DMA Peripheral Identification 3	278
0xFF0	DMAPrimeCellID0	RO	0x0000.000D	DMA PrimeCell Identification 0	280
0xFF4	DMAPrimeCellID1	RO	0x0000.00F0	DMA PrimeCell Identification 1	281
0xFF8	DMAPrimeCellID2	RO	0x0000.0005	DMA PrimeCell Identification 2	282
0xFFC	DMAPrimeCellID3	RO	0x0000.00B1	DMA PrimeCell Identification 3	283

## 8.5 $\mu$ DMA Channel Control Structure

The  $\mu$ DMA Channel Control Structure holds the  $\mu$ DMA transfer settings for a  $\mu$ DMA channel. Each channel has two control structures, which are located in a table in system memory. Refer to “Channel Configuration” on page 223 for an explanation of the Channel Control Table and the Channel Control Structure.

The channel control structure is one entry in the channel control table. Each channel has a primary and alternate structure. The primary control structures are located at offsets 0x0, 0x10, 0x20 and so on. The alternate control structures are located at offsets 0x200, 0x210, 0x220, and so on.

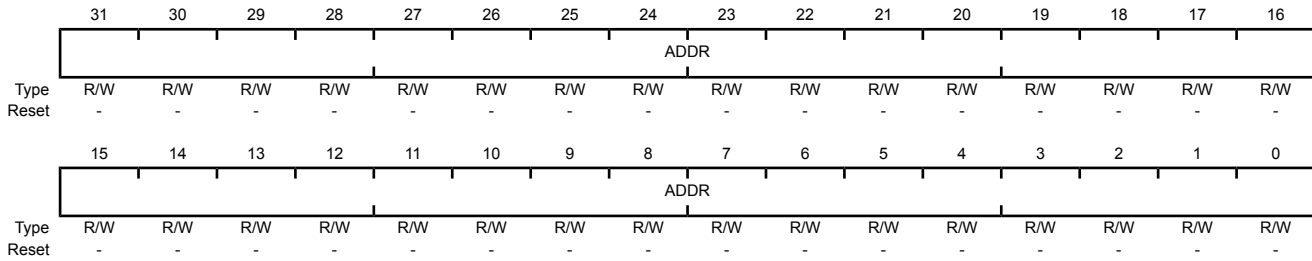
### Register 1: DMA Channel Source Address End Pointer (DMASRCENDP), offset 0x000

**DMA Channel Source Address End Pointer (DMASRCENDP)** is part of the Channel Control Structure and is used to specify the source address for a  $\mu$ DMA transfer.

**Note:** The offset specified is from the base address of the control structure in system memory, not the  $\mu$ DMA module base address.

#### DMA Channel Source Address End Pointer (DMASRCENDP)

Base n/a  
Offset 0x000  
Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31:0	ADDR	R/W	-	Source Address End Pointer  This field points to the last address of the DMA transfer source (inclusive). If the source address is not incrementing (the SRCINC field in the DMACHCTL register is 0x3), then this field points at the source location itself (such as a peripheral data register).

## Register 2: DMA Channel Destination Address End Pointer (DMADSTENDP), offset 0x004

**DMA Channel Destination Address End Pointer (DMADSTENDP)** is part of the Channel Control Structure and is used to specify the destination address for a  $\mu$ DMA transfer.

**Note:** The offset specified is from the base address of the control structure in system memory, not the  $\mu$ DMA module base address.

### DMA Channel Destination Address End Pointer (DMADSTENDP)

Base n/a

Offset 0x004

Type R/W, reset -

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

Bit/Field	Name	Type	Reset	Description
31:0	ADDR	R/W	-	Destination Address End Pointer

This field points to the last address of the DMA transfer destination (inclusive). If the destination address is not incrementing (the `DSTINC` field in the `DMACHCTL` register is 0x3), then this field points at the destination location itself (such as a peripheral data register).

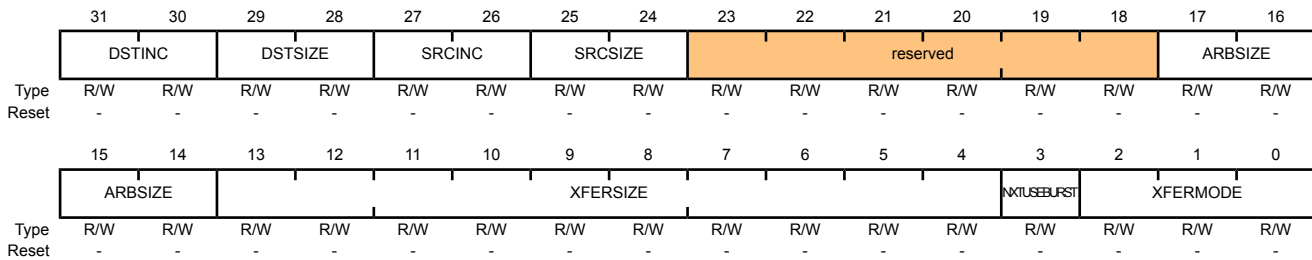
### Register 3: DMA Channel Control Word (DMACHCTL), offset 0x008

DMA Channel Control Word (DMACHCTL) is part of the Channel Control Structure and is used to specify parameters of a  $\mu$ DMA transfer.

**Note:** The offset specified is from the base address of the control structure in system memory, not the  $\mu$ DMA module base address.

#### DMA Channel Control Word (DMACHCTL)

Base n/a  
Offset 0x008  
Type R/W, reset -



Bit/Field	Name	Type	Reset	Description										
31:30	DSTINC	R/W	-	<p>Destination Address Increment</p> <p>This field configures the destination address increment.</p> <p>The address increment value must be equal or greater than the value of the destination size (DSTSIZE).</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Byte Increment by 8-bit locations</td> </tr> <tr> <td>0x1</td> <td>Half-word Increment by 16-bit locations</td> </tr> <tr> <td>0x2</td> <td>Word Increment by 32-bit locations</td> </tr> <tr> <td>0x3</td> <td>No increment Address remains set to the value of the Destination Address End Pointer (DMADSTENDP) for the channel</td> </tr> </tbody> </table>	Value	Description	0x0	Byte Increment by 8-bit locations	0x1	Half-word Increment by 16-bit locations	0x2	Word Increment by 32-bit locations	0x3	No increment Address remains set to the value of the Destination Address End Pointer (DMADSTENDP) for the channel
Value	Description													
0x0	Byte Increment by 8-bit locations													
0x1	Half-word Increment by 16-bit locations													
0x2	Word Increment by 32-bit locations													
0x3	No increment Address remains set to the value of the Destination Address End Pointer (DMADSTENDP) for the channel													



Bit/Field	Name	Type	Reset	Description										
29:28	DSTSIZE	R/W	-	<p>Destination Data Size</p> <p>This field configures the destination item data size.</p> <p><b>Note:</b> DSTSIZE must be the same as SRCSIZE.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Byte 8-bit data size</td> </tr> <tr> <td>0x1</td> <td>Half-word 16-bit data size</td> </tr> <tr> <td>0x2</td> <td>Word 32-bit data size</td> </tr> <tr> <td>0x3</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Description	0x0	Byte 8-bit data size	0x1	Half-word 16-bit data size	0x2	Word 32-bit data size	0x3	Reserved
Value	Description													
0x0	Byte 8-bit data size													
0x1	Half-word 16-bit data size													
0x2	Word 32-bit data size													
0x3	Reserved													
27:26	SRCINC	R/W	-	<p>Source Address Increment</p> <p>This field configures the source address increment.</p> <p>The address increment value must be equal or greater than the value of the source size (SRCSIZE).</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Byte Increment by 8-bit locations</td> </tr> <tr> <td>0x1</td> <td>Half-word Increment by 16-bit locations</td> </tr> <tr> <td>0x2</td> <td>Word Increment by 32-bit locations</td> </tr> <tr> <td>0x3</td> <td>No increment Address remains set to the value of the Source Address End Pointer (DMASRCENDE) for the channel</td> </tr> </tbody> </table>	Value	Description	0x0	Byte Increment by 8-bit locations	0x1	Half-word Increment by 16-bit locations	0x2	Word Increment by 32-bit locations	0x3	No increment Address remains set to the value of the Source Address End Pointer (DMASRCENDE) for the channel
Value	Description													
0x0	Byte Increment by 8-bit locations													
0x1	Half-word Increment by 16-bit locations													
0x2	Word Increment by 32-bit locations													
0x3	No increment Address remains set to the value of the Source Address End Pointer (DMASRCENDE) for the channel													
25:24	SRCSIZE	R/W	-	<p>Source Data Size</p> <p>This field configures the source item data size.</p> <p><b>Note:</b> DSTSIZE must be the same as SRCSIZE.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Byte 8-bit data size.</td> </tr> <tr> <td>0x1</td> <td>Half-word 16-bit data size.</td> </tr> <tr> <td>0x2</td> <td>Word 32-bit data size.</td> </tr> <tr> <td>0x3</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Description	0x0	Byte 8-bit data size.	0x1	Half-word 16-bit data size.	0x2	Word 32-bit data size.	0x3	Reserved
Value	Description													
0x0	Byte 8-bit data size.													
0x1	Half-word 16-bit data size.													
0x2	Word 32-bit data size.													
0x3	Reserved													

Bit/Field	Name	Type	Reset	Description																										
23:18	reserved	R/W	-	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.																										
17:14	ARBSIZE	R/W	-	<p>Arbitration Size</p> <p>This field configures the number of transfers that can occur before the <math>\mu</math>DMA controller re-arbitrates. The possible arbitration rate configurations represent powers of 2 and are shown below.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>1 Transfer</td> </tr> <tr> <td></td> <td>Arbitrates after each <math>\mu</math>DMA transfer</td> </tr> <tr> <td>0x1</td> <td>2 Transfers</td> </tr> <tr> <td>0x2</td> <td>4 Transfers</td> </tr> <tr> <td>0x3</td> <td>8 Transfers</td> </tr> <tr> <td>0x4</td> <td>16 Transfers</td> </tr> <tr> <td>0x5</td> <td>32 Transfers</td> </tr> <tr> <td>0x6</td> <td>64 Transfers</td> </tr> <tr> <td>0x7</td> <td>128 Transfers</td> </tr> <tr> <td>0x8</td> <td>256 Transfers</td> </tr> <tr> <td>0x9</td> <td>512 Transfers</td> </tr> <tr> <td>0xA-0xF</td> <td>1024 Transfers</td> </tr> </tbody> </table> <p>In this configuration, no arbitration occurs during the <math>\mu</math>DMA transfer because the maximum transfer size is 1024.</p>	Value	Description	0x0	1 Transfer		Arbitrates after each $\mu$ DMA transfer	0x1	2 Transfers	0x2	4 Transfers	0x3	8 Transfers	0x4	16 Transfers	0x5	32 Transfers	0x6	64 Transfers	0x7	128 Transfers	0x8	256 Transfers	0x9	512 Transfers	0xA-0xF	1024 Transfers
Value	Description																													
0x0	1 Transfer																													
	Arbitrates after each $\mu$ DMA transfer																													
0x1	2 Transfers																													
0x2	4 Transfers																													
0x3	8 Transfers																													
0x4	16 Transfers																													
0x5	32 Transfers																													
0x6	64 Transfers																													
0x7	128 Transfers																													
0x8	256 Transfers																													
0x9	512 Transfers																													
0xA-0xF	1024 Transfers																													
13:4	XFERSIZE	R/W	-	<p>Transfer Size (minus 1)</p> <p>This field configures the total number of items to transfer. The value of this field is 1 less than the number to transfer (value 0 means transfer 1 item). The maximum value for this 10-bit field is 1023 which represents a transfer size of 1024 items.</p> <p>The transfer size is the number of items, not the number of bytes. If the data size is 32 bits, then this value is the number of 32-bit words to transfer.</p> <p>The <math>\mu</math>DMA controller updates this field immediately prior to entering the arbitration process, so it contains the number of outstanding DMA items that is necessary to complete the <math>\mu</math>DMA cycle.</p>																										
3	NXTUSEBURST	R/W	-	<p>Next Useburst</p> <p>This field controls whether the Useburst <code>SET[n]</code> bit is automatically set for the last transfer of a peripheral scatter-gather operation. Normally, for the last transfer, if the number of remaining items to transfer is less than the arbitration size, the <math>\mu</math>DMA controller uses single transfers to complete the transaction. If this bit is set, then the controller uses a burst transfer to complete the last transfer.</p>																										

Bit/Field	Name	Type	Reset	Description																		
2:0	XFERMODE	R/W	-	<p>DMA Transfer Mode</p> <p>This field configures the operating mode of the <math>\mu</math>DMA cycle. Refer to “Transfer Modes” on page 224 for a detailed explanation of transfer modes.</p> <p>Because this register is in system RAM, it has no reset value. Therefore, this field should be initialized to 0 before the channel is enabled.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Stop</td> </tr> <tr> <td>0x1</td> <td>Basic</td> </tr> <tr> <td>0x2</td> <td>Auto-Request</td> </tr> <tr> <td>0x3</td> <td>Ping-Pong</td> </tr> <tr> <td>0x4</td> <td>Memory Scatter-Gather</td> </tr> <tr> <td>0x5</td> <td>Alternate Memory Scatter-Gather</td> </tr> <tr> <td>0x6</td> <td>Peripheral Scatter-Gather</td> </tr> <tr> <td>0x7</td> <td>Alternate Peripheral Scatter-Gather</td> </tr> </tbody> </table>	Value	Description	0x0	Stop	0x1	Basic	0x2	Auto-Request	0x3	Ping-Pong	0x4	Memory Scatter-Gather	0x5	Alternate Memory Scatter-Gather	0x6	Peripheral Scatter-Gather	0x7	Alternate Peripheral Scatter-Gather
Value	Description																					
0x0	Stop																					
0x1	Basic																					
0x2	Auto-Request																					
0x3	Ping-Pong																					
0x4	Memory Scatter-Gather																					
0x5	Alternate Memory Scatter-Gather																					
0x6	Peripheral Scatter-Gather																					
0x7	Alternate Peripheral Scatter-Gather																					

#### XFERMODE Bit Field Values.

##### Stop

Channel is stopped or configuration data is invalid. No more transfers can occur.

##### Basic

For each trigger (whether from a peripheral or a software request), the  $\mu$ DMA controller performs the number of transfers specified by the `ARBSIZE` field.

##### Auto-Request

The initial request (software- or peripheral-initiated) is sufficient to complete the entire transfer of `XFERSIZE` items without any further requests.

##### Ping-Pong

This mode uses both the primary and alternate control structures for this channel. When the number of transfers specified by the `XFERSIZE` field have completed for the current control structure (primary or alternate), the  $\mu$ DMA controller switches to the other one. These switches continue until one of the control structures is not set to ping-pong mode. At that point, the  $\mu$ DMA controller stops. An interrupt is generated upon completion of the transfers configured by each control structure. See “Ping-Pong” on page 225.

##### Memory Scatter-Gather

When using this mode, the primary control structure for the channel is configured to allow a list of operations (tasks) to be performed. The source address pointer specifies the start of a table of tasks to be copied to the alternate control structure for this channel. The `XFERMODE` field for the alternate control structure should be configured to 0x5 (Alternate memory scatter-gather) to perform the task. When the task completes, the  $\mu$ DMA switches back to the primary channel control structure, which then copies the next task to the alternate control structure. This process continues until the table of tasks is empty. The last task must have an `XFERMODE` value other than 0x5. Note that for continuous operation, the last task can update the primary channel control structure back to the start of the list or to another list. See “Memory Scatter-Gather” on page 226.

Alternate Memory Scatter-Gather

This value must be used in the alternate channel control data structure when the μDMA controller operates in Memory Scatter-Gather mode.

Peripheral Scatter-Gather

This value must be used in the primary channel control data structure when the μDMA controller operates in Peripheral Scatter-Gather mode. In this mode, the μDMA controller operates exactly the same as in Memory Scatter-Gather mode, except that instead of performing the number of transfers specified by the `XFERSIZE` field in the alternate control structure at one time, the μDMA controller only performs the number of transfers specified by the `ARBSIZE` field per trigger; see Basic mode for details. See “Peripheral Scatter-Gather” on page 230.

Alternate Peripheral Scatter-Gather

This value must be used in the alternate channel control data structure when the μDMA controller operates in Peripheral Scatter-Gather mode.

## **8.6 μDMA Register Descriptions**

The register addresses given are relative to the μDMA base address of 0x400F.F000.

## Register 4: DMA Status (DMASTAT), offset 0x000

The **DMA Status (DMASTAT)** register returns the status of the  $\mu$ DMA controller. You cannot read this register when the  $\mu$ DMA controller is in the reset state.

### DMA Status (DMASTAT)

Base 0x400F.F000

Offset 0x000

Type RO, reset 0x001F.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved											DMACHANS				
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								STATE				reserved			MASTEN
Type	RO	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
31:21	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.
20:16	DMACHANS	RO	0x1F	Available DMA Channels Minus 1  This field contains a value equal to the number of DMA channels the $\mu$ DMA controller is configured to use, minus one. The value of 0x1F corresponds to 32 DMA channels.
15:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description																								
7:4	STATE	RO	0x0	<p>Control State Machine Status</p> <p>This field shows the current status of the control state machine. Status can be one of the following.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Idle</td> </tr> <tr> <td>0x1</td> <td>Read Chan Control Data Reading channel controller data.</td> </tr> <tr> <td>0x2</td> <td>Read Source End Ptr Reading source end pointer.</td> </tr> <tr> <td>0x3</td> <td>Read Dest End Ptr Reading destination end pointer.</td> </tr> <tr> <td>0x4</td> <td>Read Source Data Reading source data.</td> </tr> <tr> <td>0x5</td> <td>Write Dest Data Writing destination data.</td> </tr> <tr> <td>0x6</td> <td>Wait for Req Clear Waiting for DMA request to clear.</td> </tr> <tr> <td>0x7</td> <td>Write Chan Control Data Writing channel controller data.</td> </tr> <tr> <td>0x8</td> <td>Stalled</td> </tr> <tr> <td>0x9</td> <td>Done</td> </tr> <tr> <td>0xA-0xF</td> <td>Undefined</td> </tr> </tbody> </table>	Value	Description	0x0	Idle	0x1	Read Chan Control Data Reading channel controller data.	0x2	Read Source End Ptr Reading source end pointer.	0x3	Read Dest End Ptr Reading destination end pointer.	0x4	Read Source Data Reading source data.	0x5	Write Dest Data Writing destination data.	0x6	Wait for Req Clear Waiting for DMA request to clear.	0x7	Write Chan Control Data Writing channel controller data.	0x8	Stalled	0x9	Done	0xA-0xF	Undefined
Value	Description																											
0x0	Idle																											
0x1	Read Chan Control Data Reading channel controller data.																											
0x2	Read Source End Ptr Reading source end pointer.																											
0x3	Read Dest End Ptr Reading destination end pointer.																											
0x4	Read Source Data Reading source data.																											
0x5	Write Dest Data Writing destination data.																											
0x6	Wait for Req Clear Waiting for DMA request to clear.																											
0x7	Write Chan Control Data Writing channel controller data.																											
0x8	Stalled																											
0x9	Done																											
0xA-0xF	Undefined																											
3:1	reserved	RO	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	MASTEN	RO	0	<p>Master Enable</p> <p>This bit shows the status of the <math>\mu</math>DMA controller.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disabled</td> </tr> <tr> <td>1</td> <td>Enabled</td> </tr> </tbody> </table>	Value	Description	0	Disabled	1	Enabled																		
Value	Description																											
0	Disabled																											
1	Enabled																											

## Register 5: DMA Configuration (DMACFG), offset 0x004

The **DMACFG** register controls the configuration of the  $\mu$ DMA controller.

### DMA Configuration (DMACFG)

Base 0x400F.F000

Offset 0x004

Type WO, reset -

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	WO	-	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	MASTEN	WO	-	Controller Master Enable This bit enables the $\mu$ DMA controller.
				Value Description
				0 Disable
				1 Enable

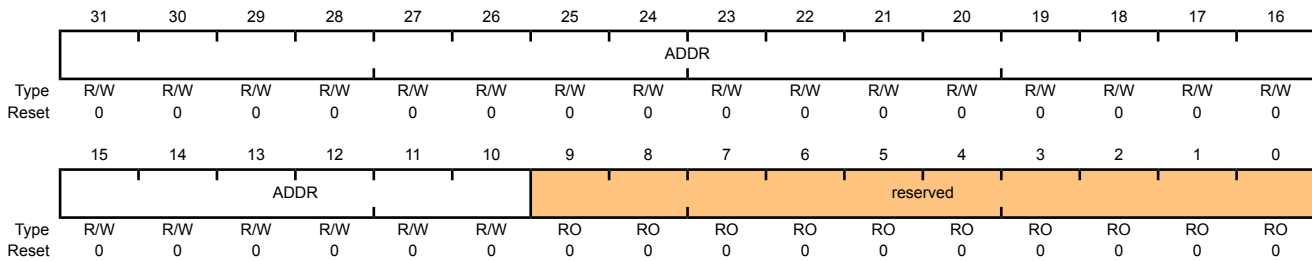
### Register 6: DMA Channel Control Base Pointer (DMACTLBASE), offset 0x008

The **DMACTLBASE** register must be configured so that the base pointer points to a location in system memory.

The amount of system memory that must be assigned to the  $\mu$ DMA controller depends on the number of DMA channels used and whether the alternate channel control data structure is used. See “Channel Configuration” on page 223 for details about the Channel Control Table. The base address must be aligned on a 1024-byte boundary. This register cannot be read when the  $\mu$ DMA controller is in the reset state.

#### DMA Channel Control Base Pointer (DMACTLBASE)

Base 0x400F.F000  
 Offset 0x008  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:10	ADDR	R/W	0x0000.00	Channel Control Base Address  This field contains the pointer to the base address of the channel control table. The base address must be 1024-byte aligned.
9:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.



## Register 7: DMA Alternate Channel Control Base Pointer (DMAALTBASE), offset 0x00C

The **DMAALTBASE** register returns the base address of the alternate channel control data. This register removes the necessity for application software to calculate the base address of the alternate channel control structures. This register cannot be read when the  $\mu$ DMA controller is in the reset state.

### DMA Alternate Channel Control Base Pointer (DMAALTBASE)

Base 0x400F.F000

Offset 0x00C

Type RO, reset 0x0000.0200

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ADDR															
Type	RO	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
	ADDR															
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	0	0	0	0	0	0	0	0	0

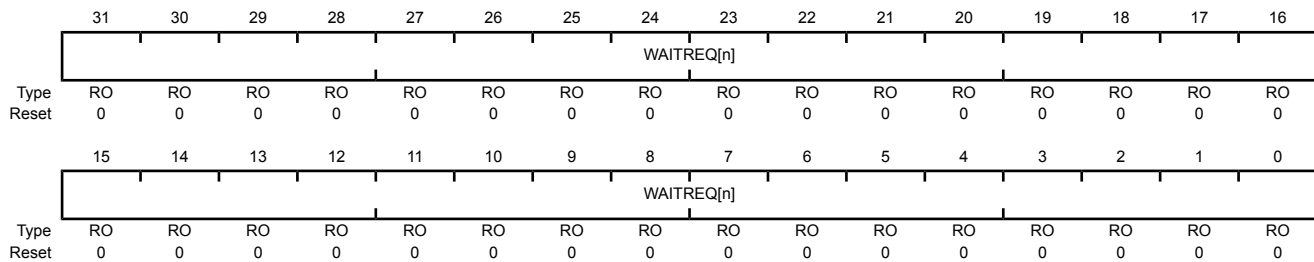
Bit/Field	Name	Type	Reset	Description
31:0	ADDR	RO	0x0000.0200	Alternate Channel Address Pointer This field provides the base address of the alternate channel control structures.

### Register 8: DMA Channel Wait-on-Request Status (DMAWAITSTAT), offset 0x010

This read-only register indicates that the  $\mu$ DMA channel is waiting on a request. A peripheral can pull this Low to hold off the  $\mu$ DMA from performing a single request until the peripheral is ready for a burst request. The use of this feature is dependent on the design of the peripheral and is used to enhance performance of the  $\mu$ DMA with that peripheral. This register cannot be read when the  $\mu$ DMA controller is in the reset state.

#### DMA Channel Wait-on-Request Status (DMAWAITSTAT)

Base 0x400F.F000  
 Offset 0x010  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	WAITREQ[n]	RO	0x0000.0000	Channel [n] Wait Status  These bits provide the channel wait-on-request status. Bit 0 corresponds to channel 0.  Value Description 1 The corresponding channel is waiting on a request. 0 The corresponding channel is not waiting on a request.

## Register 9: DMA Channel Software Request (DMASWREQ), offset 0x014

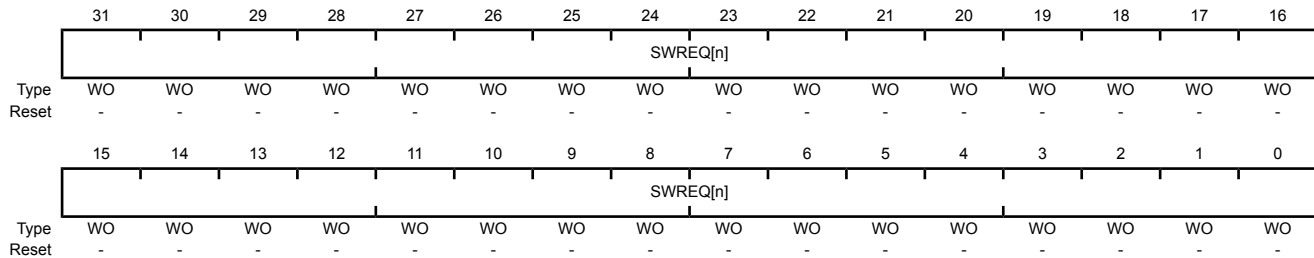
Each bit of the **DMASWREQ** register represents the corresponding  $\mu$ DMA channel. Setting a bit generates a request for the specified  $\mu$ DMA channel.

### DMA Channel Software Request (DMASWREQ)

Base 0x400F.F000

Offset 0x014

Type WO, reset -



Bit/Field	Name	Type	Reset	Description
31:0	SWREQ[n]	WO	-	Channel [n] Software Request These bits generate software requests. Bit 0 corresponds to channel 0.
				Value Description
				1 Generate a software request for the corresponding channel.
				0 No request generated.

### Register 10: DMA Channel Useburst Set (DMAUSEBURSTSET), offset 0x018

Each bit of the **DMAUSEBURSTSET** register represents the corresponding  $\mu$ DMA channel. Setting a bit disables the channel's single request input from generating requests, configuring the channel to only accept burst requests. Reading the register returns the status of USEBURST.

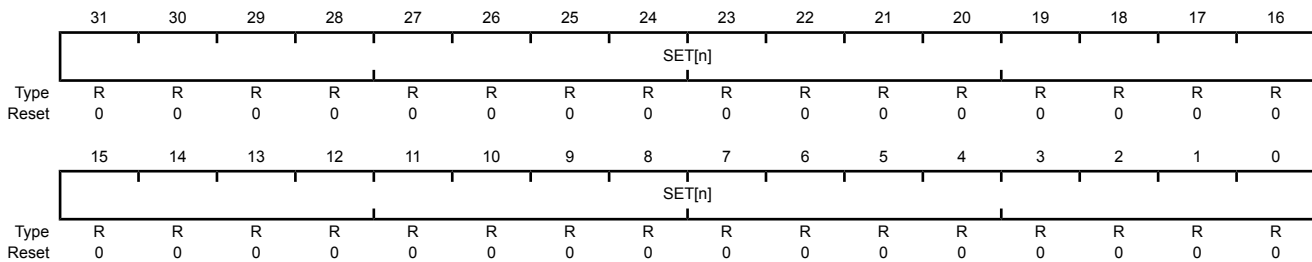
When there are fewer items remaining to transfer than the arbitration (burst) size, the  $\mu$ DMA controller automatically clears the corresponding `SET[n]` bit, allowing the remaining items to transfer using single requests. A bit should not be set if the corresponding peripheral does not support the burst request model.

Refer to "Request Types" on page 222 for more details about request types.

#### Reads

##### DMA Channel Useburst Set (DMAUSEBURSTSET)

Base 0x400F.F000  
 Offset 0x018  
 Type RO, reset 0x0000.0000

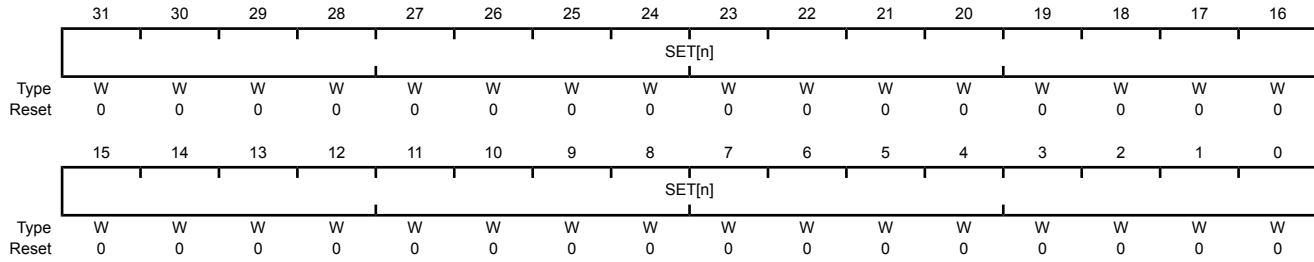


Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R	0x00	Channel [n] Useburst Status Returns the useburst status of channel [n].
				Value Description
				0 Single and Burst $\mu$ DMA channel [n] responds to single or burst requests.
				1 Burst Only $\mu$ DMA channel [n] responds only to burst requests.

**Writes**

**DMA Channel Useburst Set (DMAUSEBURSTSET)**

Base 0x400F.F000  
 Offset 0x018  
 Type WO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	W	0x00	Channel [n] Useburst Set
				Sets useburst bit on channel [n]. Use the <b>DMAUSEBURSTCLR</b> register to clear bit [n].
				Value Description
				0 No Effect
				1 Burst Only
				μDMA channel [n] responds only to burst requests.

**Register 11: DMA Channel Useburst Clear (DMAUSEBURSTCLR), offset 0x01C**

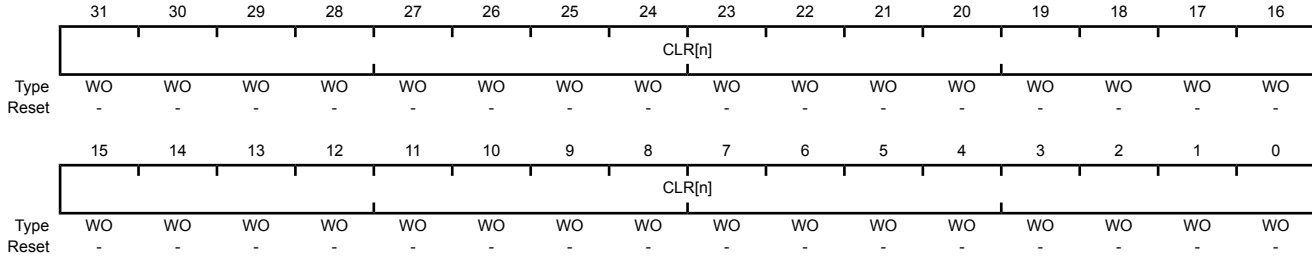
Each bit of the **DMAUSEBURSTCLR** register represents the corresponding DMA channel. Writing a 1 enables `dma_sreq[n]` to generate requests.

DMA Channel Useburst Clear (DMAUSEBURSTCLR)

Base 0x400F.F000

Offset 0x01C

Type WO, reset -



Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Channel [n] Useburst Clear Clears useburst bit on channel [n].
				Value Description
				0 No Effect
				Use the <b>DMAUSEBURSTSET</b> to set bit [n] to 1.
				1 Single and Burst
				DMA channel [n] responds to single and burst requests.

## Register 12: DMA Channel Request Mask Set (DMAREQMASKSET), offset 0x020

Each bit of the **DMAREQMASKSET** register represents the corresponding DMA channel. Writing a 1 disables DMA requests for the channel. Reading the register returns the request mask status. When a  $\mu$ DMA channel's request is masked, that means the peripheral can no longer request  $\mu$ DMA transfers. The channel can then be used for software-initiated transfers.

### Reads

#### DMA Channel Request Mask Set (DMAREQMASKSET)

Base 0x400F.F000  
Offset 0x020  
Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R	0x00	Channel [n] Request Mask Status Returns the channel request mask status.
	Value	Description		
	0	Enabled		External requests are not masked for channel [n].
	1	Masked		External requests are masked for channel [n].

### Writes

#### DMA Channel Request Mask Set (DMAREQMASKSET)

Base 0x400F.F000  
Offset 0x020  
Type WO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	W	0x00	Channel [n] Request Mask Set Masks (disables) the corresponding channel [n] from generating DMA requests.  Value Description 0 No Effect Use the <b>DMAREQMASKCLR</b> register to clear the request mask. 1 Masked Masks (disables) DMA requests on channel [n].



## Register 13: DMA Channel Request Mask Clear (DMAREQMASKCLR), offset 0x024

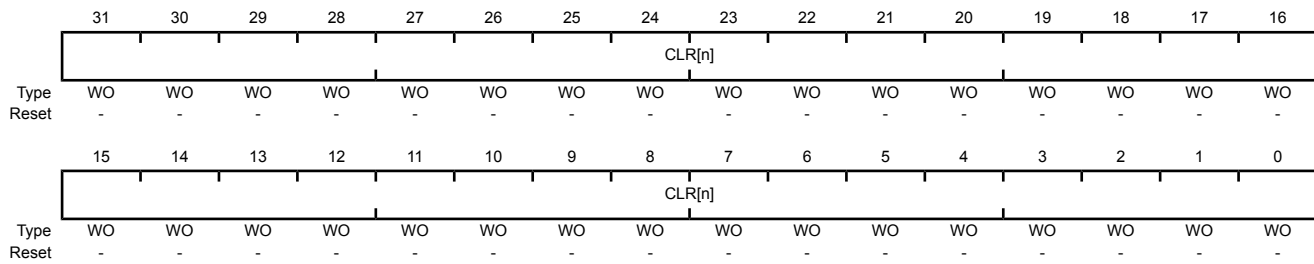
Each bit of the **DMAREQMASKCLR** register represents the corresponding DMA channel. Writing a 1 clears the request mask for the channel, and enables the channel to receive DMA requests.

### DMA Channel Request Mask Clear (DMAREQMASKCLR)

Base 0x400F.F000

Offset 0x024

Type WO, reset -



Bit/Field	Name	Type	Reset	Description						
31:0	CLR[n]	WO	-	<p>Channel [n] Request Mask Clear</p> <p>Set the appropriate bit to clear the DMA request mask for channel [n]. This will enable DMA requests for the channel.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No Effect</td> </tr> <tr> <td>1</td> <td>Clear Mask</td> </tr> </tbody> </table> <p>Use the <b>DMAREQMASKSET</b> register to set the request mask.</p> <p>Clears the request mask for the DMA channel. This enables DMA requests for the channel.</p>	Value	Description	0	No Effect	1	Clear Mask
Value	Description									
0	No Effect									
1	Clear Mask									

### Register 14: DMA Channel Enable Set (DMAENASET), offset 0x028

Each bit of the **DMAENASET** register represents the corresponding DMA channel. Writing a 1 enables the DMA channel. Reading the register returns the enable status of the channels. If a channel is enabled but the request mask is set (**DMAREQMASKSET**), then the channel can be used for software-initiated transfers.

#### Reads

##### DMA Channel Enable Set (DMAENASET)

Base 0x400F.F000  
 Offset 0x028  
 Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R	0x00	Channel [n] Enable Status Returns the enable status of the channels.
	Value	Description		
	0	Disabled		
	1	Enabled		

#### Writes

##### DMA Channel Enable Set (DMAENASET)

Base 0x400F.F000  
 Offset 0x028  
 Type WO, reset 0x0000.0000

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

---

Bit/Field	Name	Type	Reset	Description										
31:0	SET[n]	W	0x00	<p>Channel [n] Enable Set</p> <p>Enables the corresponding channels.</p> <p><b>Note:</b> The controller disables a channel when it completes the DMA cycle.</p> <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0</td><td>No Effect</td></tr><tr><td></td><td>Use the <b>DMAENACL</b>R register to disable a channel.</td></tr><tr><td>1</td><td>Enable</td></tr><tr><td></td><td>Enables channel [n].</td></tr></tbody></table>	Value	Description	0	No Effect		Use the <b>DMAENACL</b> R register to disable a channel.	1	Enable		Enables channel [n].
Value	Description													
0	No Effect													
	Use the <b>DMAENACL</b> R register to disable a channel.													
1	Enable													
	Enables channel [n].													

### Register 15: DMA Channel Enable Clear (DMAENACLR), offset 0x02C

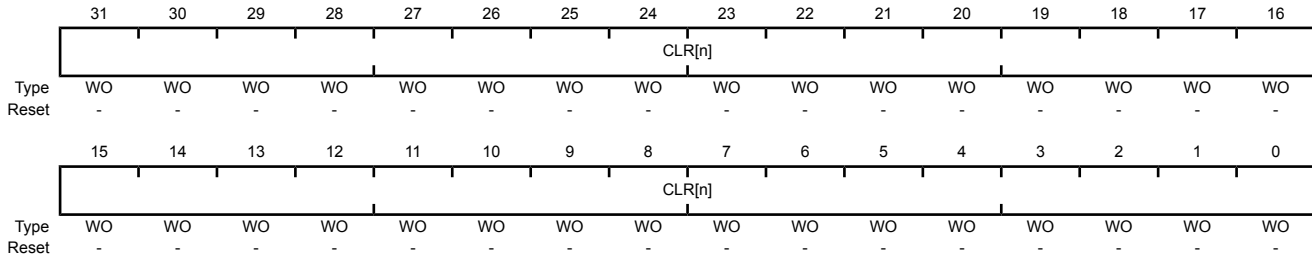
Each bit of the **DMAENACLR** register represents the corresponding DMA channel. Writing a 1 disables the specified DMA channel.

#### DMA Channel Enable Clear (DMAENACLR)

Base 0x400F.F000

Offset 0x02C

Type WO, reset -



Bit/Field	Name	Type	Reset	Description										
31:0	CLR[n]	WO	-	<p>Clear Channel [n] Enable</p> <p>Set the appropriate bit to disable the corresponding DMA channel.</p> <p><b>Note:</b> The controller disables a channel when it completes the DMA cycle.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No Effect</td> </tr> <tr> <td></td> <td>Use the <b>DMAENASET</b> register to enable DMA channels.</td> </tr> <tr> <td>1</td> <td>Disable</td> </tr> <tr> <td></td> <td>Disables channel [n].</td> </tr> </tbody> </table>	Value	Description	0	No Effect		Use the <b>DMAENASET</b> register to enable DMA channels.	1	Disable		Disables channel [n].
Value	Description													
0	No Effect													
	Use the <b>DMAENASET</b> register to enable DMA channels.													
1	Disable													
	Disables channel [n].													

## Register 16: DMA Channel Primary Alternate Set (DMAALTSET), offset 0x030

Each bit of the **DMAALTSET** register represents the corresponding DMA channel. Writing a 1 configures the DMA channel to use the alternate control data structure. Reading the register returns the status of which control data structure is in use for the corresponding DMA channel.

### Reads

#### DMA Channel Primary Alternate Set (DMAALTSET)

Base 0x400F.F000  
Offset 0x030  
Type RO, reset 0x0000.0000

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

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

31:0	SET[n]	R	0x00	Channel [n] Alternate Status
------	--------	---	------	------------------------------

Returns the channel control data structure status.

Value Description

0 Primary

DMA channel [n] is using the primary control structure.

1 Alternate

DMA channel [n] is using the alternate control structure.

### Writes

#### DMA Channel Primary Alternate Set (DMAALTSET)

Base 0x400F.F000  
Offset 0x030  
Type WO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description						
31:0	SET[n]	W	0x00	<p>Channel [n] Alternate Set</p> <p>Selects the alternate channel control data structure for the corresponding DMA channel.</p> <p><b>Note:</b> For Ping-Pong and Scatter-Gather DMA cycle types, the controller automatically sets these bits to select the alternate channel control data structure.</p> <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0</td><td>No Effect</td></tr><tr><td>1</td><td>Alternate</td></tr></tbody></table> <p>Use the <b>DMAALTCLR</b> register to set bit [n] to 0.</p> <p>Selects the alternate control data structure for channel [n].</p>	Value	Description	0	No Effect	1	Alternate
Value	Description									
0	No Effect									
1	Alternate									

## Register 17: DMA Channel Primary Alternate Clear (DMAALTCLR), offset 0x034

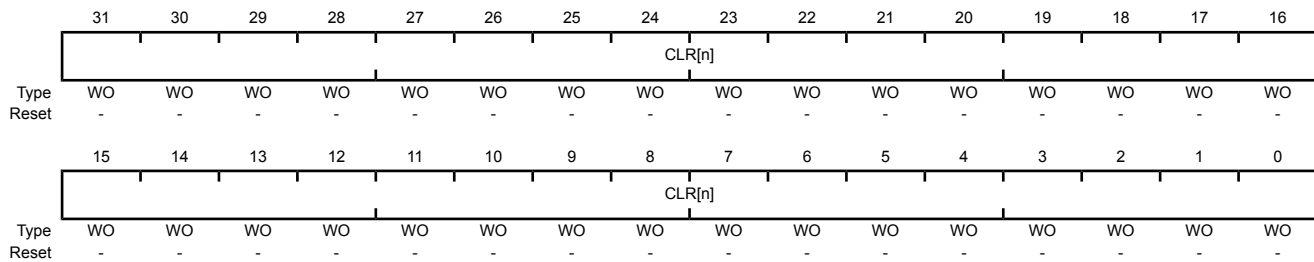
Each bit of the **DMAALTCLR** register represents the corresponding DMA channel. Writing a 1 configures the DMA channel to use the primary control data structure.

### DMA Channel Primary Alternate Clear (DMAALTCLR)

Base 0x400F.F000

Offset 0x034

Type WO, reset -



Bit/Field	Name	Type	Reset	Description						
31:0	CLR[n]	WO	-	<p>Channel [n] Alternate Clear</p> <p>Set the appropriate bit to select the primary control data structure for the corresponding DMA channel.</p> <p><b>Note:</b> For Ping-Pong and Scatter-Gather DMA cycle types, the controller sets these bits to select the primary channel control data structure.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No Effect</td> </tr> <tr> <td>1</td> <td>Primary</td> </tr> </tbody> </table> <p>Selects the primary control data structure for channel [n].</p>	Value	Description	0	No Effect	1	Primary
Value	Description									
0	No Effect									
1	Primary									

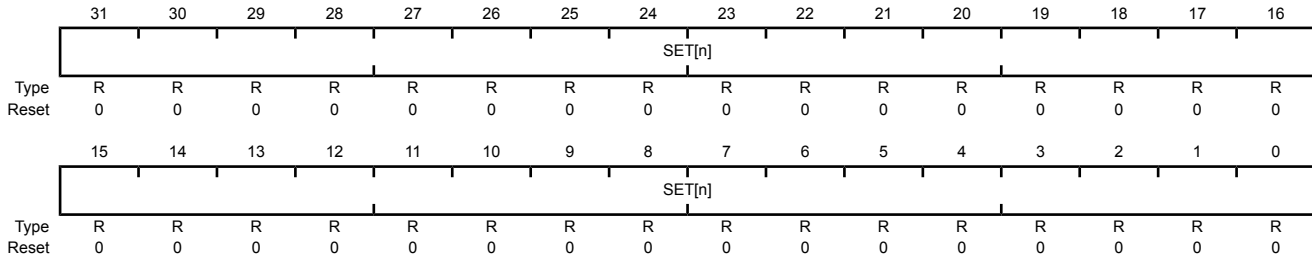
### Register 18: DMA Channel Priority Set (DMAPRIOSET), offset 0x038

Each bit of the **DMAPRIOSET** register represents the corresponding DMA channel. Writing a 1 configures the DMA channel to have a high priority level. Reading the register returns the status of the channel priority mask.

#### Reads

##### DMA Channel Priority Set (DMAPRIOSET)

Base 0x400F.F000  
 Offset 0x038  
 Type RO, reset 0x0000.0000



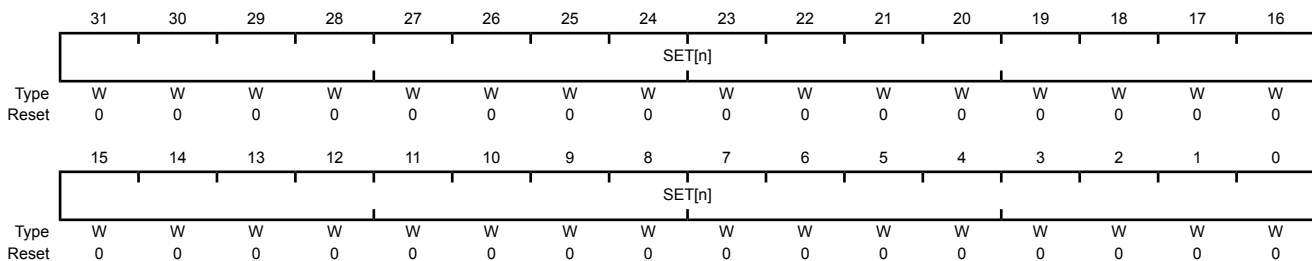
Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R	0x00	Channel [n] Priority Status Returns the channel priority status.

Value	Description
0	Default Priority DMA channel [n] is using the default priority level.
1	High Priority DMA channel [n] is using a High Priority level.

#### Writes

##### DMA Channel Priority Set (DMAPRIOSET)

Base 0x400F.F000  
 Offset 0x038  
 Type WO, reset 0x0000.0000





---

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	W	0x00	Channel [n] Priority Set Sets the channel priority to high.  Value Description 0 No Effect Use the <b>DMAPRIOCLR</b> register to set channel [n] to the default priority level. 1 High Priority Sets DMA channel [n] to a High Priority level.

### Register 19: DMA Channel Priority Clear (DMAPRIOCLR), offset 0x03C

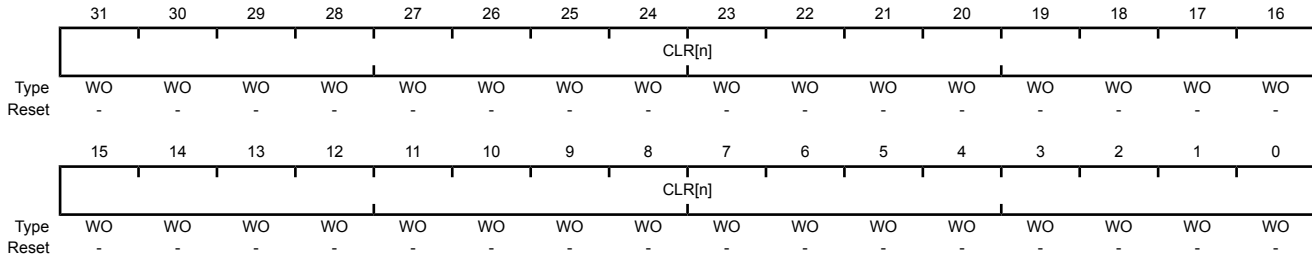
Each bit of the **DMAPRIOCLR** register represents the corresponding DMA channel. Writing a 1 configures the DMA channel to have the default priority level.

#### DMA Channel Priority Clear (DMAPRIOCLR)

Base 0x400F.F000

Offset 0x03C

Type WO, reset -



Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	<p>Channel [n] Priority Clear</p> <p>Set the appropriate bit to clear the high priority level for the specified DMA channel.</p> <p>Value Description</p> <p>0 No Effect</p> <p>Use the <b>DMAPRIOSET</b> register to set channel [n] to the High priority level.</p> <p>1 Default Priority</p> <p>Sets DMA channel [n] to a Default priority level.</p>

## Register 20: DMA Bus Error Clear (DMAERRCLR), offset 0x04C

The **DMAERRCLR** register is used to read and clear the  $\mu$ DMA bus error status. The error status is set if the  $\mu$ DMA controller encountered a bus error while performing a DMA transfer. If a bus error occurs on a channel, that channel is automatically disabled by the  $\mu$ DMA controller. The other channels are unaffected.

### Reads

#### DMA Bus Error Clear (DMAERRCLR)

Base 0x400F.F000  
Offset 0x04C  
Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	ERRCLR	R	0	DMA Bus Error Status
				Value Description
			1	Low No bus error is pending.
			1	High Bus error is pending.

### Writes

#### DMA Bus Error Clear (DMAERRCLR)

Base 0x400F.F000  
Offset 0x04C  
Type WO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	ERRCLR	W	0	DMA Bus Error Clear Clears the bus error.  Value Description 0 No Effect Bus error status is unchanged. 1 Clear Clears a pending bus error.

## Register 21: DMA Channel Alternate Select (DMACHALT), offset 0x500

Each bit of the **DMACHALT** register represents the corresponding  $\mu$ DMA channel. Setting a bit selects the alternate channel assignment as specified in Table 8-1 on page 221.

### DMA Channel Alternate Select (DMACHALT)

Base 0x400F.F000

Offset 0x500

Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CHALT[n]															
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	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CHALT[n]															
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	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

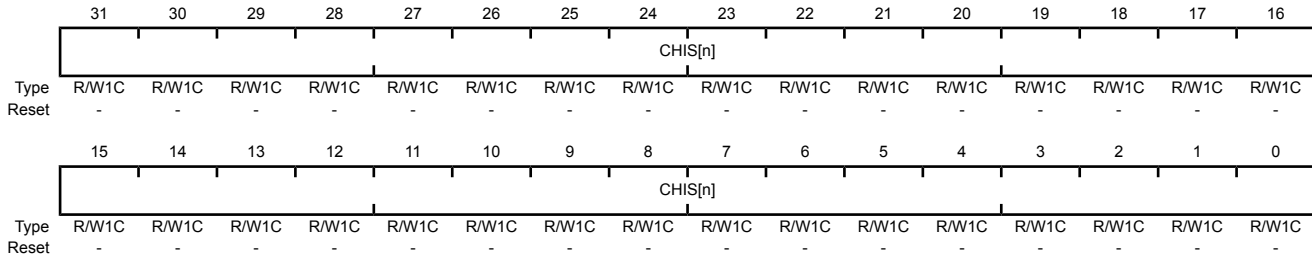
Bit/Field	Name	Type	Reset	Description
31:0	CHALT[n]	R/W	-	Channel [n] Alternate Assignment Select
				Value Description
				0 Use the primary channel assignment.
				1 Use the alternate channel assignment.

## Register 22: DMA Channel Interrupt Status (DMACHIS), offset 0x504

Each bit of the **DMACHIS** register represents the corresponding  $\mu$ DMA channel. A bit is set when that  $\mu$ DMA channel causes an interrupt. The bits are sticky and cleared by a writing a 1.

### DMA Channel Interrupt Status (DMACHIS)

Base 0x400F.F000  
 Offset 0x504  
 Type R/W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:0	CHIS[n]	R/W1C	-	<p>Channel [n] Interrupt Status</p> <p>A read of 1 indicates that channel caused an interrupt. Writing a 1 clears the channel if an interrupt was set.</p> <p>Value Description</p> <table border="0"> <tr> <td>1</td> <td>When read, this bit indicates that the corresponding channel caused an interrupt.</td> </tr> <tr> <td></td> <td>Writing a 1 clears the channel if an interrupt was set.</td> </tr> <tr> <td>0</td> <td>The corresponding channel has not caused an interrupt.</td> </tr> </table>	1	When read, this bit indicates that the corresponding channel caused an interrupt.		Writing a 1 clears the channel if an interrupt was set.	0	The corresponding channel has not caused an interrupt.
1	When read, this bit indicates that the corresponding channel caused an interrupt.									
	Writing a 1 clears the channel if an interrupt was set.									
0	The corresponding channel has not caused an interrupt.									

## Register 23: DMA Peripheral Identification 0 (DMAPeriphID0), offset 0xFE0

The **DMAPeriphIDn** registers are hard-coded, and the fields within the registers determine the reset values.

### DMA Peripheral Identification 0 (DMAPeriphID0)

Base 0x400F.F000

Offset 0xFE0

Type RO, reset 0x0000.0030

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

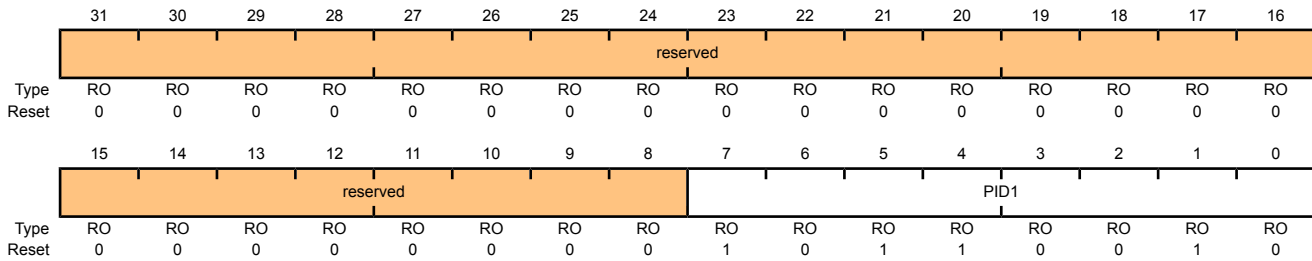
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x30	DMA Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

### Register 24: DMA Peripheral Identification 1 (DMAPeriphID1), offset 0xFE4

The DMAPeriphIDn registers are hard-coded, and the fields within the registers determine the reset values.

#### DMA Peripheral Identification 1 (DMAPeriphID1)

Base 0x400F.F000  
 Offset 0xFE4  
 Type RO, reset 0x0000.00B2



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0xB2	DMA Peripheral ID Register[ 15:8] Can be used by software to identify the presence of this peripheral.



## Register 25: DMA Peripheral Identification 2 (DMAPeriphID2), offset 0xFE8

The **DMAPeriphIDn** registers are hard-coded, and the fields within the registers determine the reset values.

### DMA Peripheral Identification 2 (DMAPeriphID2)

Base 0x400F.F000

Offset 0xFE8

Type RO, reset 0x0000.000B

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

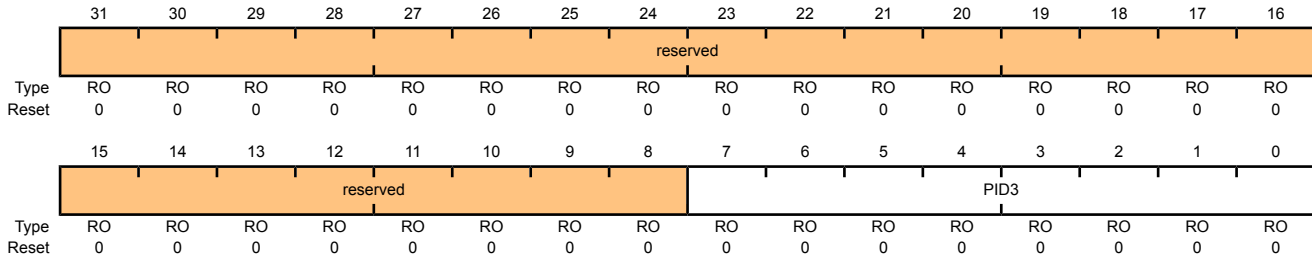
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x0B	DMA Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

## Register 26: DMA Peripheral Identification 3 (DMAPeriphID3), offset 0xFEC

The **DMAPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

### DMA Peripheral Identification 3 (DMAPeriphID3)

Base 0x400F.F000  
 Offset 0xFEC  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x00	DMA Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

## Register 27: DMA Peripheral Identification 4 (DMAPeriphID4), offset 0xFD0

The **DMAPeriphIDn** registers are hard-coded, and the fields within the registers determine the reset values.

### DMA Peripheral Identification 4 (DMAPeriphID4)

Base 0x400F.F000

Offset 0xFD0

Type RO, reset 0x0000.0004

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

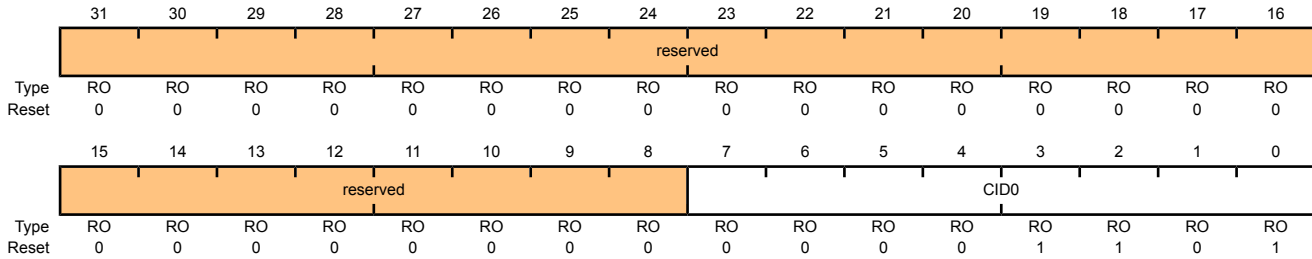
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x04	DMA Peripheral ID Register Can be used by software to identify the presence of this peripheral.

### Register 28: DMA PrimeCell Identification 0 (DMAPCellID0), offset 0xFF0

The **DMAPCellIDn** registers are hard-coded, and the fields within the registers determine the reset values.

#### DMA PrimeCell Identification 0 (DMAPCellID0)

Base 0x400F.F000  
 Offset 0xFF0  
 Type RO, reset 0x0000.000D



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	DMA PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

## Register 29: DMA PrimeCell Identification 1 (DMAPCellID1), offset 0xFF4

The **DMAPCellIDn** registers are hard-coded, and the fields within the registers determine the reset values.

### DMA PrimeCell Identification 1 (DMAPCellID1)

Base 0x400F.F000

Offset 0xFF4

Type RO, reset 0x0000.00F0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

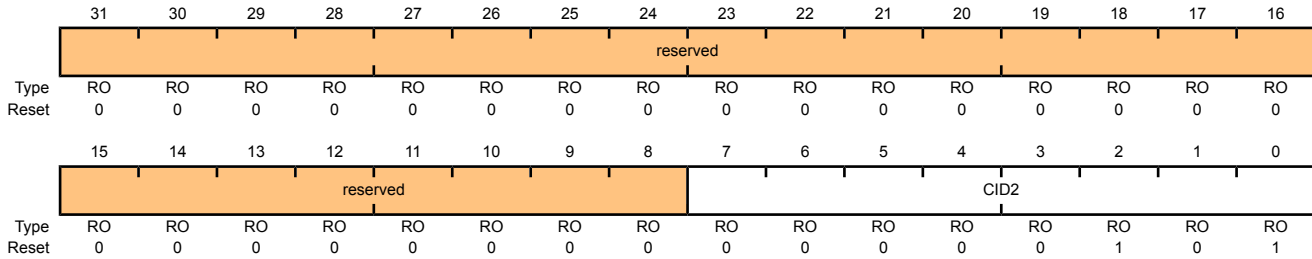
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	DMA PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system.

### Register 30: DMA PrimeCell Identification 2 (DMAPCellID2), offset 0xFF8

The **DMAPCellIDn** registers are hard-coded, and the fields within the registers determine the reset values.

#### DMA PrimeCell Identification 2 (DMAPCellID2)

Base 0x400F.F000  
 Offset 0xFF8  
 Type RO, reset 0x0000.0005



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

## Register 31: DMA PrimeCell Identification 3 (DMAPCellID3), offset 0xFFC

The **DMAPCellIDn** registers are hard-coded, and the fields within the registers determine the reset values.

### DMA PrimeCell Identification 3 (DMAPCellID3)

Base 0x400F.F000

Offset 0xFFC

Type RO, reset 0x0000.00B1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID3							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	DMA PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system.

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

The GPIO module is composed of nine physical GPIO blocks, each corresponding to an individual GPIO port (Port A, Port B, Port C, Port D, Port E, Port F, Port G, Port H, Port J). The GPIO module supports 0-72 programmable input/output pins, depending on the peripherals being used.

The GPIO module has the following features:

- 0-72 GPIOs, depending on configuration
- Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
- 5-V-tolerant input/outputs
- Fast toggle capable of a change every two clock cycles
- Two means of port access: either Advanced Host Bus (AHB) with better back-to-back access performance, or the legacy Advanced Peripheral Bus (APB) for backwards-compatibility with existing code
- Programmable control for GPIO interrupts
  - Interrupt generation masking
  - Edge-triggered on rising, falling, or both
  - Level-sensitive on High or Low values
- Bit masking in both read and write operations through address lines
- Can be used to initiate an ADC sample sequence
- Pins configured as digital inputs are Schmitt-triggered
- Programmable control for GPIO pad configuration
  - Weak pull-up or pull-down resistors
  - 2-mA, 4-mA, and 8-mA pad drive for digital communication; up to four pads can be configured with an 18-mA pad drive for high-current applications
  - Slew rate control for the 8-mA drive
  - Open drain enables
  - Digital input enables

### 9.1 Functional Description

**Important:** All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL=0**, **GPIODEN=0**, **GPIOPDR=0**, and **GPIOPUR=0**) with the exception of the pins shown in Table 9-1. A Power-On-Reset ( $\overline{POR}$ ) or asserting  $\overline{RST}$  puts the pins back to their default state.



**Table 9-1. GPIO Pins With Non-Zero Reset Values**

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	1	1	0	0	0x1
PA[5:2]	SSI0	1	1	0	0	0x1
PB[3:2]	I <sup>2</sup> C0	1	1	0	0	0x1
PC[3:0]	JTAG	1	1	0	1	0x3

Each GPIO port is a separate hardware instantiation of the same physical block (see Figure 9-1 on page 285 and Figure 9-2 on page 286). The LM3S5791 microcontroller contains nine ports and thus nine of these physical GPIO blocks. Some GPIO pins can function as I/O signals for the on-chip peripheral modules. For information on which GPIO pins are used for alternate hardware functions, refer to Table 24-5 on page 937.

**Figure 9-1. Digital I/O Pads**

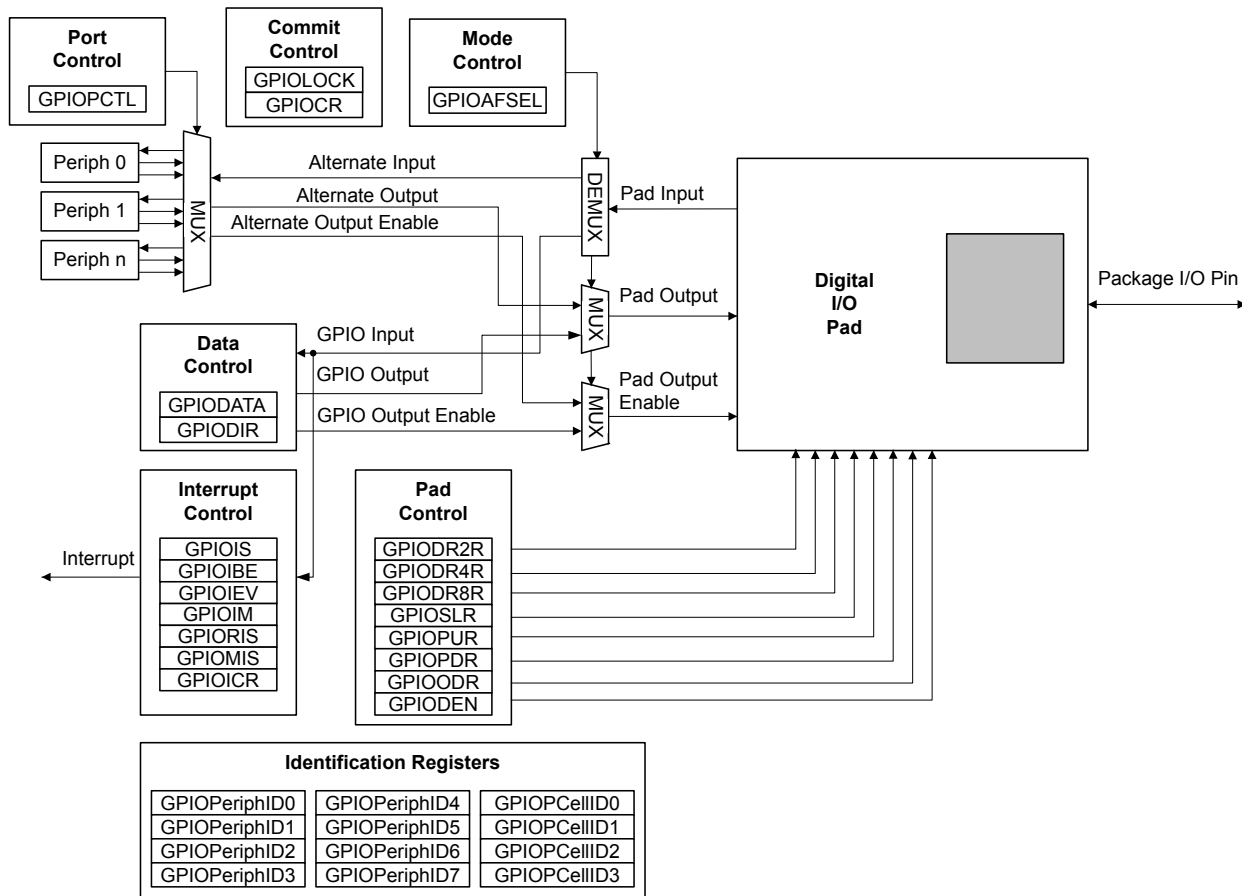
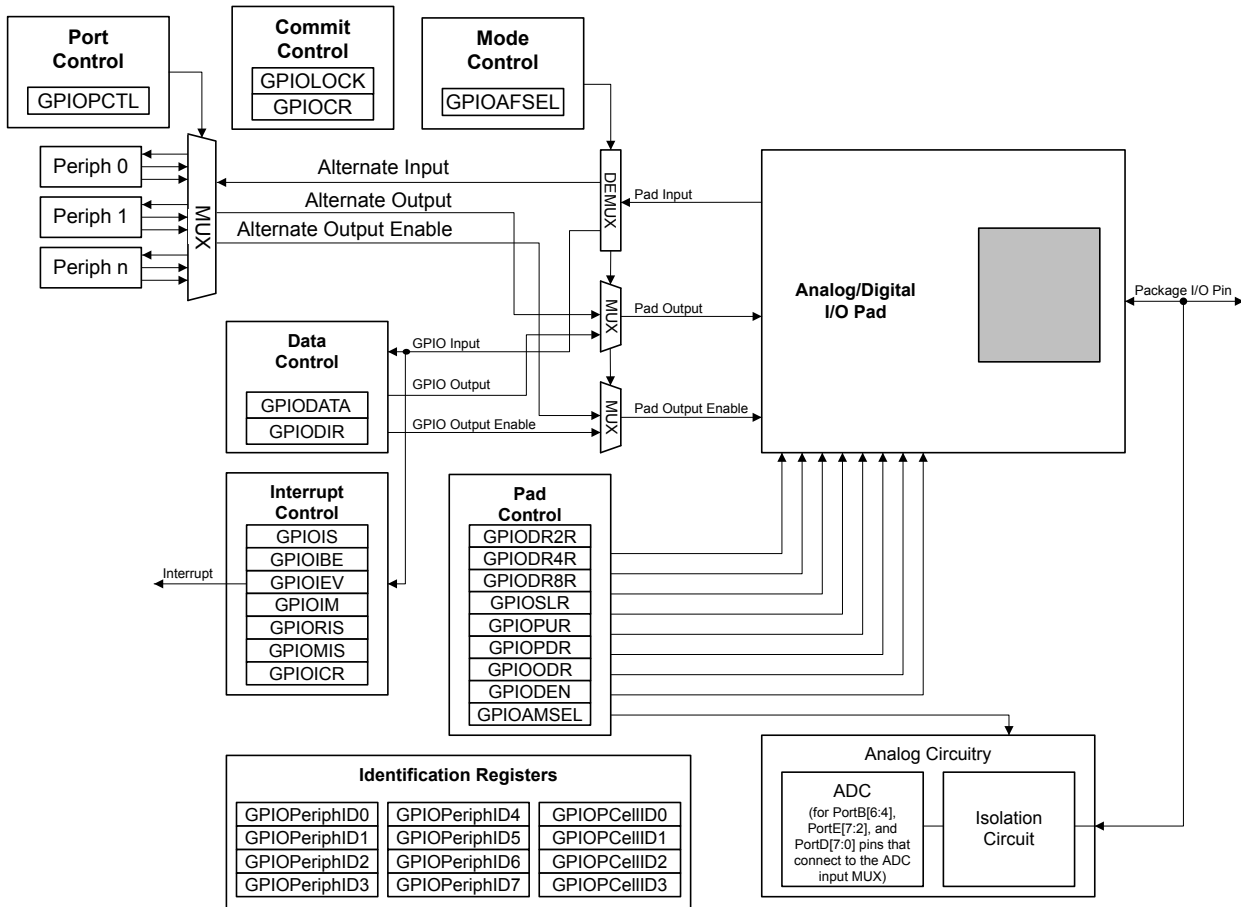


Figure 9-2. Analog/Digital I/O Pads



### 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.

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

#### 9.1.1.1 Data Direction Operation

The **GPIO Direction (GPIODIR)** register (see page 295) is used to configure each individual pin as an input or output. When the data direction bit is cleared, the GPIO is configured as an input, and the corresponding data register bit captures and stores the value on the GPIO port. When the data direction bit is set, the GPIO is configured as an output, and the corresponding data register bit is driven out on the GPIO port.

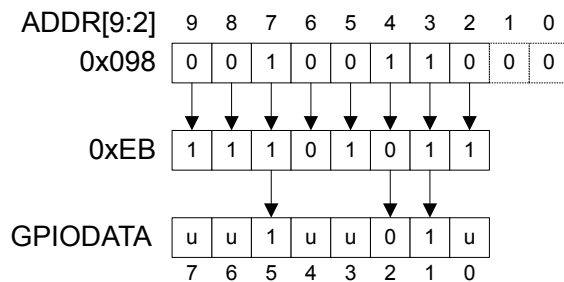
### 9.1.1.2 Data Register Operation

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 294) by using bits [9:2] of the address bus as a mask. In this manner, software drivers can modify individual GPIO pins in a single instruction without affecting the state of the other pins. This method is more efficient than the conventional method of performing a read-modify-write operation to set or clear an individual GPIO pin. To implement 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, the value of the **GPIODATA** register is altered. If the address bit is cleared, the data bit is left unchanged.

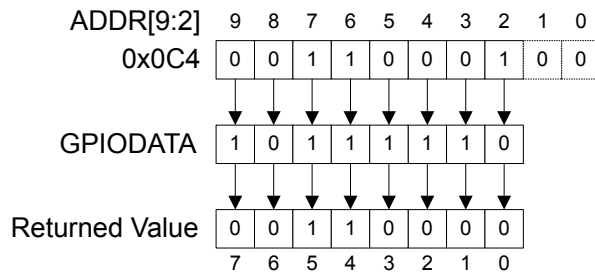
For example, writing a value of 0xEB to the address GPIODATA + 0x098 has the results shown in Figure 9-3, where u indicates that data is unchanged by the write.

**Figure 9-3. GPIODATA Write Example**



During a read, if the address bit associated with the data bit is set, the value is read. If the address bit associated with the data bit is cleared, the data bit is read as a zero, regardless of its actual value. For example, reading address GPIODATA + 0x0C4 yields as shown in Figure 9-4.

**Figure 9-4. GPIODATA Read Example**



## 9.1.2 Interrupt Control

The interrupt capabilities of each GPIO port are controlled by a set of seven registers. These registers are used 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, the external source must hold the level constant for the interrupt to be recognized by the controller.

Three registers define the edge or sense that causes interrupts:

- **GPIO Interrupt Sense (GPIOIS)** register (see page 296)

- **GPIO Interrupt Both Edges (GPIOIBE)** register (see page 297)
- **GPIO Interrupt Event (GPIOIEV)** register (see page 298)

Interrupts are enabled/disabled via the **GPIO Interrupt Mask (GPIOIM)** register (see page 299).

When an interrupt condition occurs, the state of the interrupt signal can be viewed in two locations: the **GPIO Raw Interrupt Status (GPIORIS)** and **GPIO Masked Interrupt Status (GPIOMIS)** registers (see page 300 and page 301). As the name implies, the **GPIOMIS** register only shows interrupt conditions that are allowed to be passed to the interrupt controller. The **GPIORIS** register indicates that a GPIO pin meets the conditions for an interrupt, but has not necessarily been sent to the interrupt controller.

In addition to providing GPIO functionality,  $PB4$  can also be used as an external trigger for the ADC. If  $PB4$  is configured as a non-masked interrupt pin (the appropriate bit of **GPIOIM** is set), an interrupt for Port B is generated, and 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. See page 470.

If no other Port B pins are being used to generate interrupts, the ARM Integrated Nested Vectored Interrupt Controller (NVIC) Interrupt Set Enable (SETENA) register can disable the Port B interrupts, and the ADC interrupt can be used to read back the converted data. Otherwise, the Port B interrupt handler must ignore and clear interrupts on  $PB4$  and wait for the ADC interrupt, or the ADC interrupt must be disabled in the SETENA register and the Port B interrupt handler must poll the ADC registers until the conversion is completed. See the *ARM® Cortex™-M3 Technical Reference Manual* for more information.

Interrupts are cleared by writing a 1 to the appropriate bit of the **GPIO Interrupt Clear (GPIOICR)** register (see page 303).

When programming the interrupt control registers (**GPIOIS**, **GPIOIBE**, or **GPIOIEV**), the interrupts should be masked (**GPIOIM** cleared). Writing any value to an interrupt control register can generate a spurious interrupt if the corresponding bits are enabled.

### 9.1.3 Mode Control

The GPIO pins can be controlled by either software or hardware. Software control is the default for most signals and corresponds to the GPIO mode, where the **GPIO DATA** register is used to read or write the corresponding pins. When hardware control is enabled via the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 304), the pin state is controlled by its alternate function (that is, the peripheral).

Further pin muxing options are provided through the **GPIO Port Control (GPIOCTL)** register which selects one of several peripheral functions for each GPIO. For information on the configuration options, refer to Table 24-5 on page 937.

**Note:** If any pin is to be used as an ADC input, the appropriate bit in the **GPIOAMSEL** register must be set to disable the analog isolation circuit.

### 9.1.4 Commit Control

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the **NMI** pin ( $PB7$ ) and the four **JTAG/SWD** pins ( $PC[3:0]$ ). Writes to protected bits of the **GPIOAFSEL** register, **GPIOPUR** register, **GPIO Pull-Down Select (GPIOPDR)** register (see page 312), and **GPIODEN** register are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 316) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 317) have been set.

### 9.1.5 Pad Control

The pad control registers allow software to configure the GPIO pads based on the application requirements. The pad control registers include the **GPIODR2R**, **GPIODR4R**, **GPIODR8R**, **GPIODR**, **GPIOPUR**, **GPIOPDR**, **GPIOSLR**, and **GPIODEN** registers. These registers control drive strength, open-drain configuration, pull-up and pull-down resistors, slew-rate control and digital input enable for each GPIO.

For special high-current applications, the GPIO output buffers may be used with the following restrictions. With the GPIO pins configured as 8-mA output drivers, a total of four GPIO outputs may be used to sink current loads up to 18 mA each. At 18-mA sink current loading, the  $V_{OL}$  value is specified as 1.2 V. The high-current GPIO package pins must be selected such that there are only a maximum of two per side of the physical package with the total number of high-current GPIO outputs not exceeding four for the entire package.

### 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 **GPIOPCellID0-GPIOPCellID3** registers.

## 9.2 Initialization and Configuration

The GPIO modules may be accessed via two different memory apertures. The legacy aperture, the Advanced Peripheral Bus (APB), is backwards-compatible with previous Stellaris® parts. The other aperture, the Advanced Host Bus (AHB), offers the same register map but provides better back-to-back access performance than the APB bus. These apertures are mutually exclusive. The aperture enabled for a given GPIO port is controlled by the appropriate bit in the **GPIOHCTL** register (see page 115).

To use the pins in a particular GPIO port, the clock for the port must be enabled by setting the appropriate GPIO Port bit field ( $GPIO_n$ ) in the **RCGC2** register (see page 175).

On reset, all GPIO pins are configured out of reset to be undriven (tristate): **GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, and **GPIOPUR**=0, except for the pins shown in Table 9-1 on page 285. Table 9-2 on page 289 shows all possible configurations of the GPIO pads and the control register settings required to achieve them. Table 9-3 on page 290 shows how a rising edge interrupt is configured for pin 2 of a GPIO port.

**Table 9-2. 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

Configuration	GPIO Register Bit Value <sup>a</sup>									
	AFSEL	DIR	ODR	DEN	PUR	PDR	DR2R	DR4R	DR8R	SLR
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 9-3. GPIO Interrupt Configuration Example**

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

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

### 9.3 Register Map

Table 9-5 on page 292 lists the GPIO registers. Each GPIO port can be accessed through one of two bus apertures. The legacy aperture, the Advanced Peripheral Bus (APB), is backwards-compatible with previous Stellaris<sup>®</sup> parts. The other aperture, the Advanced Host Bus (AHB), offers the same register map but provides better back-to-back access performance than the APB bus.

**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 unconnected bits has no effect, and reading unconnected bits returns no meaningful data.

The offset listed is a hexadecimal increment to the register's address, relative to that GPIO port's base address:

- GPIO Port A (APB): 0x4000.4000
- GPIO Port A (AHB): 0x4005.8000
- GPIO Port B (APB): 0x4000.5000
- GPIO Port B (AHB): 0x4005.9000
- GPIO Port C (APB): 0x4000.6000
- GPIO Port C (AHB): 0x4005.A000
- GPIO Port D (APB): 0x4000.7000
- GPIO Port D (AHB): 0x4005.B000
- GPIO Port E (APB): 0x4002.4000
- GPIO Port E (AHB): 0x4005.C000
- GPIO Port F (APB): 0x4002.5000
- GPIO Port F (AHB): 0x4005.D000
- GPIO Port G (APB): 0x4002.6000
- GPIO Port G (AHB): 0x4005.E000
- GPIO Port H (APB): 0x4002.7000
- GPIO Port H (AHB): 0x4005.F000
- GPIO Port J (APB): 0x4003.D000
- GPIO Port J (AHB): 0x4006.0000

Note that each GPIO module clock must be enabled before the registers can be programmed (see page 175).

**Important:** All GPIO pins are configured as GPIOs and tri-stated by default (**GPIAAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, and **GPIOPUR**=0) with the exception of the pins shown in Table 9-1. A Power-On-Reset ( $\overline{POR}$ ) or asserting  $\overline{RST}$  puts the pins back to their default state.

**Table 9-4. GPIO Pins With Non-Zero Reset Values**

GPIO Pins	Default State	GPIAAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	1	1	0	0	0x1
PA[5:2]	SSI0	1	1	0	0	0x1
PB[3:2]	I <sup>2</sup> C0	1	1	0	0	0x1
PC[3:0]	JTAG	1	1	0	1	0x3

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

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

Table 9-5. GPIO Register Map

Offset	Name	Type	Reset	Description	See page
0x000	GPIODATA	R/W	0x0000.0000	GPIO Data	294
0x400	GPIODIR	R/W	0x0000.0000	GPIO Direction	295
0x404	GPIOIS	R/W	0x0000.0000	GPIO Interrupt Sense	296
0x408	GPIOIBE	R/W	0x0000.0000	GPIO Interrupt Both Edges	297
0x40C	GPIOIEV	R/W	0x0000.0000	GPIO Interrupt Event	298
0x410	GPIOIM	R/W	0x0000.0000	GPIO Interrupt Mask	299
0x414	GPIORIS	RO	0x0000.0000	GPIO Raw Interrupt Status	300
0x418	GPIONIS	RO	0x0000.0000	GPIO Masked Interrupt Status	301
0x41C	GPIOICR	W1C	0x0000.0000	GPIO Interrupt Clear	303
0x420	GPIOAFSEL	R/W	-	GPIO Alternate Function Select	304
0x500	GPIODR2R	R/W	0x0000.00FF	GPIO 2-mA Drive Select	306
0x504	GPIODR4R	R/W	0x0000.0000	GPIO 4-mA Drive Select	307
0x508	GPIODR8R	R/W	0x0000.0000	GPIO 8-mA Drive Select	308
0x50C	GPIOODR	R/W	0x0000.0000	GPIO Open Drain Select	309
0x510	GPPOPUR	R/W	-	GPIO Pull-Up Select	310
0x514	GPPOPDR	R/W	0x0000.0000	GPIO Pull-Down Select	312
0x518	GPPOSLR	R/W	0x0000.0000	GPIO Slew Rate Control Select	313
0x51C	GPPODEN	R/W	-	GPIO Digital Enable	314
0x520	GPIOLOCK	R/W	0x0000.0001	GPIO Lock	316
0x524	GPIOCR	-	-	GPIO Commit	317
0x528	GPIOAMSEL	R/W	0x0000.0000	GPIO Analog Mode Select	319
0x52C	GPPOPCTL	R/W	-	GPIO Port Control	321
0xFD0	GPIOPeriphID4	RO	0x0000.0000	GPIO Peripheral Identification 4	323
0xFD4	GPIOPeriphID5	RO	0x0000.0000	GPIO Peripheral Identification 5	324
0xFD8	GPIOPeriphID6	RO	0x0000.0000	GPIO Peripheral Identification 6	325
0xFDC	GPIOPeriphID7	RO	0x0000.0000	GPIO Peripheral Identification 7	326
0xFE0	GPIOPeriphID0	RO	0x0000.0061	GPIO Peripheral Identification 0	327
0xFE4	GPIOPeriphID1	RO	0x0000.0000	GPIO Peripheral Identification 1	328
0xFE8	GPIOPeriphID2	RO	0x0000.0018	GPIO Peripheral Identification 2	329
0xFEC	GPIOPeriphID3	RO	0x0000.0001	GPIO Peripheral Identification 3	330
0xFF0	GPIOPCellID0	RO	0x0000.000D	GPIO PrimeCell Identification 0	331
0xFF4	GPIOPCellID1	RO	0x0000.00F0	GPIO PrimeCell Identification 1	332



---

Offset	Name	Type	Reset	Description	See page
0xFF8	GPIOCellID2	RO	0x0000.0005	GPIO PrimeCell Identification 2	333
0xFFC	GPIOCellID3	RO	0x0000.00B1	GPIO PrimeCell Identification 3	334

## 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 295).

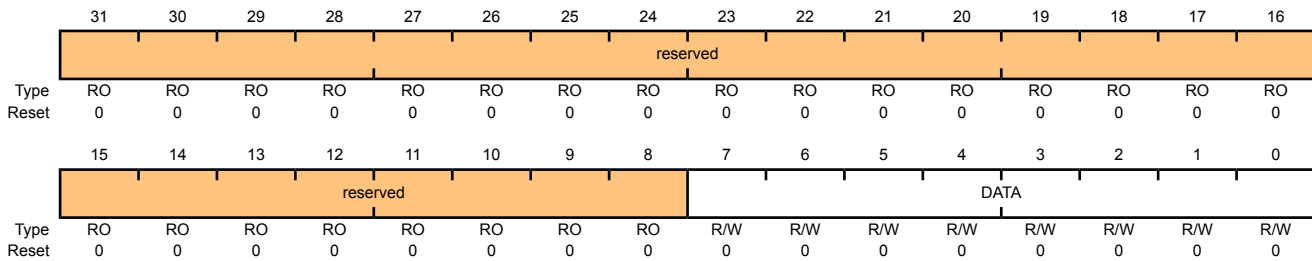
In order to write to **GPIODATA**, the corresponding bits in the mask, resulting from the address bus bits [9:2], must be set. Otherwise, the bit values remain unchanged by the write.

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 set in the address mask cause the corresponding bits in **GPIODATA** to be read, and bits that are clear 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 (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0x000  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x00	GPIO Data

This register is virtually mapped to 256 locations in the address space. To facilitate the reading and writing of data to these registers by independent drivers, the data read from and written to the registers are masked by the eight address lines [9:2]. Reads from this register return its current state. Writes to this register only affect bits that are not masked by ADDR[9:2] and are configured as outputs. See “Data Register Operation” on page 287 for examples of reads and writes.

## Register 2: GPIO Direction (GPIODIR), offset 0x400

The **GPIODIR** register is the data direction register. Setting a bit in the **GPIODIR** register configures the corresponding pin to be an output, while clearing a bit configures the corresponding pin to be an input. All bits are cleared by a reset, meaning all GPIO pins are inputs by default.

### GPIO Direction (GPIODIR)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0x400  
 Type R/W, reset 0x0000.0000

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

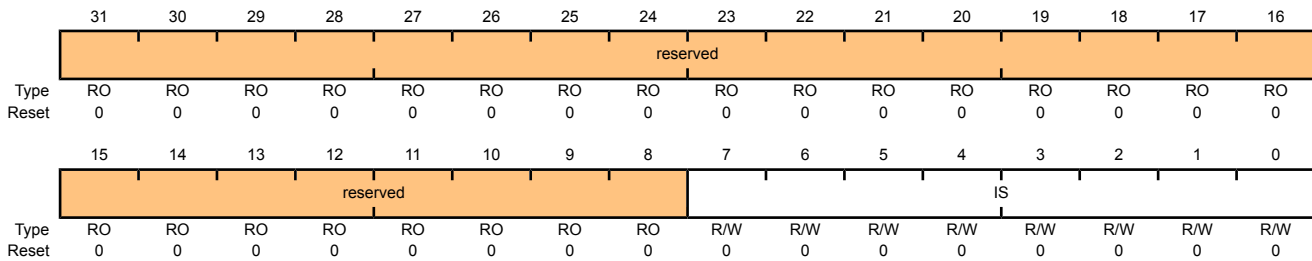
Bit/Field	Name	Type	Reset	Description	
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.	
7:0	DIR	R/W	0x00	GPIO Data Direction	
Value Description					
	0	Corresponding pin is an input.			
	1	Corresponding pins is an output.			

### Register 3: GPIO Interrupt Sense (GPIOIS), offset 0x404

The **GPIOIS** register is the interrupt sense register. Setting a bit in the **GPIOIS** register configures the corresponding pin to detect levels, while clearing a bit configures the corresponding pin to detect edges. All bits are cleared by a reset.

#### GPIO Interrupt Sense (GPIOIS)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0x404  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IS	R/W	0x00	GPIO Interrupt Sense

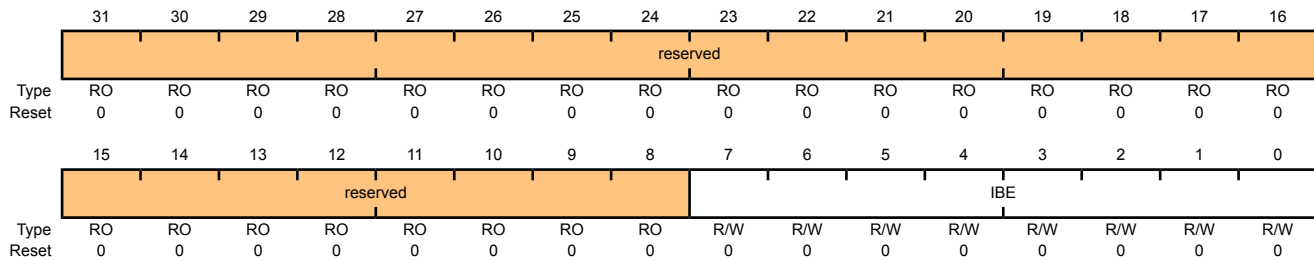
Value	Description
0	The edge on the corresponding pin is detected (edge-sensitive).
1	The level on the corresponding pin is detected (level-sensitive).

### Register 4: GPIO Interrupt Both Edges (GPIOIBE), offset 0x408

The **GPIOIBE** register allows both edges to cause interrupts. When the corresponding bit in the **GPIO Interrupt Sense (GPIOIS)** register (see page 296) is set to detect edges, setting a bit in the **GPIOIBE** register configures the corresponding pin to detect both rising and falling edges, regardless of the corresponding bit in the **GPIO Interrupt Event (GPIOIEV)** register (see page 298). Clearing a bit configures the pin to be controlled by the **GPIOIEV** register. All bits are cleared by a reset.

#### GPIO Interrupt Both Edges (GPIOIBE)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0x408  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IBE	R/W	0x00	GPIO Interrupt Both Edges

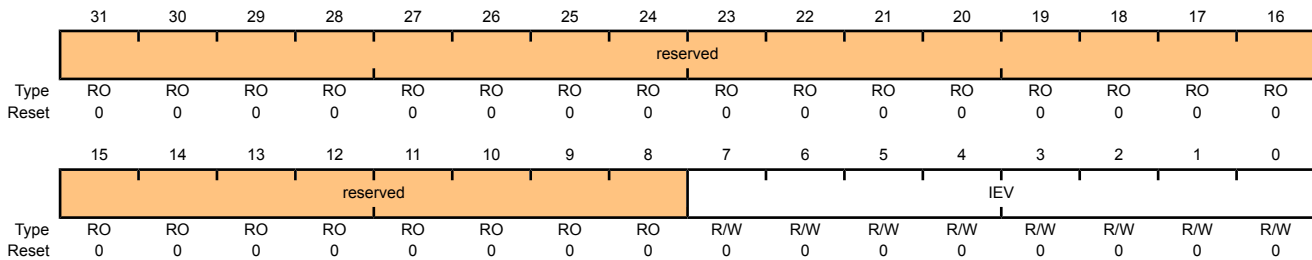
Value	Description
0	Interrupt generation is controlled by the <b>GPIO Interrupt Event (GPIOIEV)</b> register (see page 298).
1	Both edges on the corresponding pin trigger an interrupt.

### Register 5: GPIO Interrupt Event (GPIOIEV), offset 0x40C

The **GPIOIEV** register is the interrupt event register. Setting a bit in the **GPIOIEV** register configures 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 296). Clearing a bit configures the pin to detect falling edges or low levels, depending on the corresponding bit value in the **GPIOIS** register. All bits are cleared by a reset.

#### GPIO Interrupt Event (GPIOIEV)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0x40C  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IEV	R/W	0x00	GPIO Interrupt Event

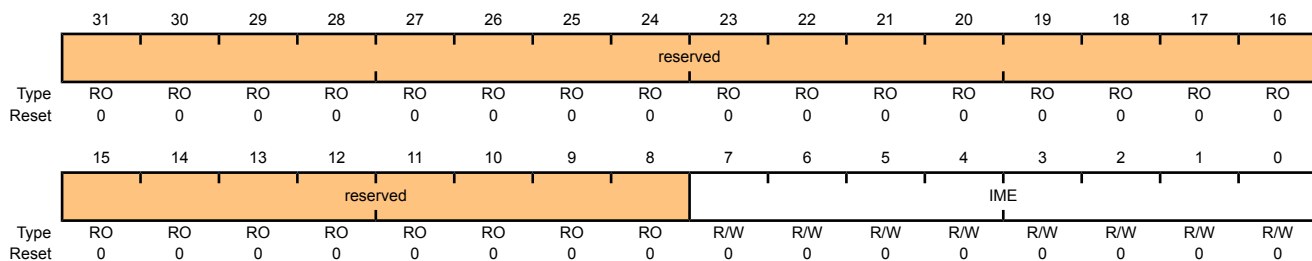
Value	Description
0	A falling edge or a Low level on the corresponding pin triggers an interrupt.
1	A rising edge or a High level on the corresponding pin triggers an interrupt.

## Register 6: GPIO Interrupt Mask (GPIOIM), offset 0x410

The **GPIOIM** register is the interrupt mask register. Setting a bit in the **GPIOIM** register allows interrupts that are generated by the corresponding pin to be sent to the interrupt controller on the combined interrupt signal. Clearing a bit prevents an interrupt on the corresponding pin from being sent to the interrupt controller. All bits are cleared by a reset.

### GPIO Interrupt Mask (GPIOIM)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0x410  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IME	R/W	0x00	GPIO Interrupt Mask Enable

#### Value Description

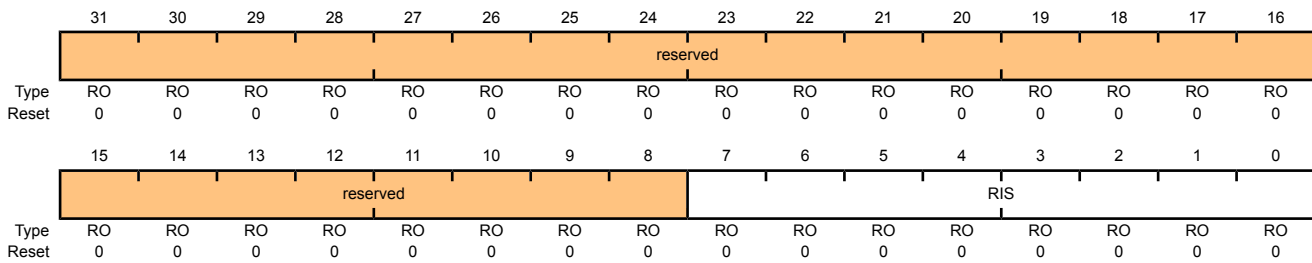
- |   |   |
|---|---|
| 0 | The interrupt from the corresponding pin is masked.                           |
| 1 | The interrupt from the corresponding pin is sent to the interrupt controller. |

### Register 7: GPIO Raw Interrupt Status (GPIORIS), offset 0x414

The **GPIORIS** register is the raw interrupt status register. A bit in this register is set when an interrupt condition occurs on the corresponding GPIO pin. If the corresponding bit in the **GPIO Interrupt Mask (GPIOIM)** register (see page 299) is set, the interrupt is sent to the interrupt controller. Bits read as zero indicate that corresponding input pins have not initiated an interrupt. A bit in this register can be cleared by writing a 1 to the corresponding bit in the **GPIO Interrupt Clear (GPIOICR)** register.

#### GPIO Raw Interrupt Status (GPIORIS)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0x414  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	RIS	RO	0x00	GPIO Interrupt Raw Status

Value	Description
1	An interrupt condition has occurred on the corresponding pin.
0	An interrupt condition has not occurred on the corresponding pin.

A bit is cleared by writing a 1 to the corresponding bit in the **GPIOICR** register.



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

The **GPIOMIS** register is the masked interrupt status register. If a bit is set in this register, the corresponding interrupt has triggered an interrupt to the interrupt controller. If a bit is clear, either no interrupt has been generated, or the interrupt is masked.

In addition to providing GPIO functionality, **PB4** can also be used as an external trigger for the ADC. If **PB4** is configured as a non-masked interrupt pin (the appropriate bit of **GPIOIM** is set), an interrupt for Port B is generated, and 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. See page 470.

If no other Port B pins are being used to generate interrupts, the ARM Integrated Nested Vectored Interrupt Controller (NVIC) Interrupt Set Enable (SETNA) register can disable the Port B interrupts, and the ADC interrupt can be used to read back the converted data. Otherwise, the Port B interrupt handler must ignore and clear interrupts on PB4 and wait for the ADC interrupt, or the ADC interrupt must be disabled in the SETNA register and the Port B interrupt handler must poll the ADC registers until the conversion is completed. See the *ARM® Cortex™-M3 Technical Reference Manual* for more information.

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

### GPIO Masked Interrupt Status (GPIOMIS)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0x418  
 Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
7:0	MIS	RO	0x00	GPIO Masked Interrupt Status
				Value Description
				1 An interrupt condition on the corresponding pin has triggered an interrupt to the interrupt controller.
				0 An interrupt condition on the corresponding pin is masked or has not occurred.
				A bit is cleared by writing a 1 to the corresponding bit in the <b>GPIOICR</b> register.

## 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 bit in the **GPIOIRIS** and **GPIOMIS** registers. Writing a 0 has no effect.

### GPIO Interrupt Clear (GPIOICR)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0x41C  
 Type W1C, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IC	W1C	0x00	GPIO Interrupt Clear

#### Value Description

- 1 The corresponding interrupt is cleared.
- 0 The corresponding interrupt is unaffected.

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

The **GPIOAFSEL** register is the mode control select register. If a bit is clear, the pin is used as a GPIO and is controlled by the GPIO registers. Setting a bit in this register configures the corresponding GPIO line to be controlled by an associated peripheral. Several possible peripheral functions are multiplexed on each GPIO. The **GPIO Port Control (GPIOPCTL)** register is used to select one of the possible functions. Table 24-5 on page 937 details which functions are muxed on each GPIO pin. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 9-1.

**Important:** All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, and **GPIOPUR**=0) with the exception of the pins shown in Table 9-1. A Power-On-Reset ( $\overline{POR}$ ) or asserting  $\overline{RST}$  puts the pins back to their default state.

**Table 9-6. GPIO Pins With Non-Zero Reset Values**

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	1	1	0	0	0x1
PA[5:2]	SSI0	1	1	0	0	0x1
PB[3:2]	I <sup>2</sup> C0	1	1	0	0	0x1
PC[3:0]	JTAG	1	1	0	1	0x3

**Caution – It is possible to create a software sequence that prevents the debugger from connecting to the Stellaris<sup>®</sup> 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. As a result, the debugger may be locked out of the part. This issue can be avoided with a software routine that restores JTAG functionality based on an external or software trigger.**

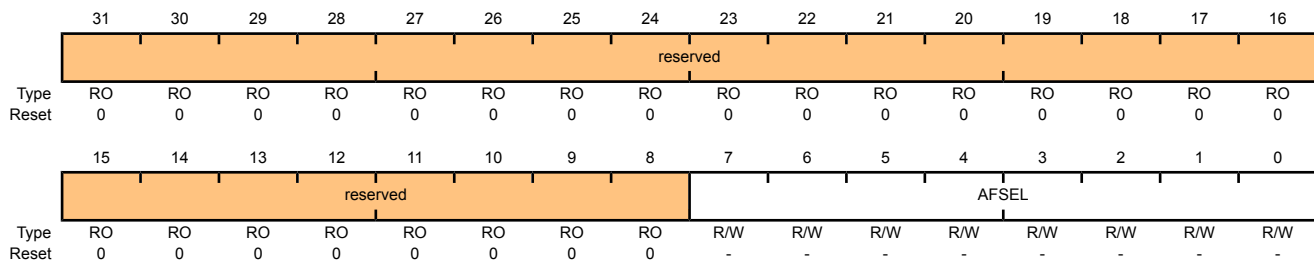
The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the **NMI** pin (**PB7**) and the four **JTAG/SWD** pins (**PC[3:0]**). Writes to protected bits of the **GPIOAFSEL** register, **GPIOPUR** register, **GPIO Pull-Down Select (GPIOPDR)** register (see page 312), and **GPIODEN** register are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 316) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 317) have been set.

When using the I<sup>2</sup>C module, in addition to setting the **GPIOAFSEL** register bits for the I<sup>2</sup>C clock and data pins, the pins should be set to open drain using the **GPIO Open Drain Select (GPIOODR)** register (see examples in “Initialization and Configuration” on page 289).

GPIO Alternate Function Select (GPIOAFSEL)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000

Offset 0x420  
 Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	AFSEL	R/W	-	GPIO Alternate Function Select

Value Description

- 0 The associated pin functions as a GPIO and is controlled by the GPIO registers.
- 1 The associated pin functions as a peripheral signal and is controlled by the alternate hardware function.

The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 9-1 on page 285.

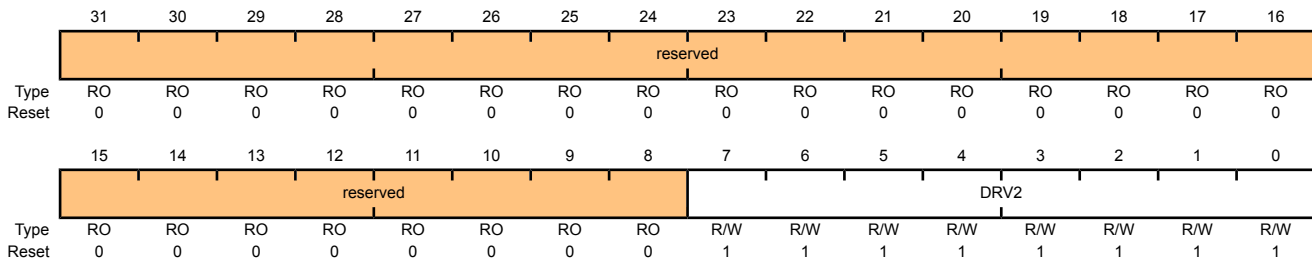
### Register 11: GPIO 2-mA Drive Select (GPIODR2R), offset 0x500

The **GPIODR2R** register is the 2-mA drive control register. Each GPIO signal in the port can be individually configured without affecting the other pads. When setting the **DRV2** bit for a GPIO signal, the corresponding **DRV4** bit in the **GPIODR4R** register and **DRV8** bit in the **GPIODR8R** register are automatically cleared by hardware. By default, all GPIO pins have 2-mA drive.

#### GPIO 2-mA Drive Select (GPIODR2R)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0x500

Type R/W, reset 0x0000.00FF



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV2	R/W	0xFF	Output Pad 2-mA Drive Enable

**Value Description**

- 1 The corresponding GPIO pin has 2-mA drive.
- 0 The drive for the corresponding GPIO pin is controlled by the **GPIODR4R** or **GPIODR8R** register.

Setting a bit in either the **GPIODR4** register or the **GPIODR8** register clears the corresponding 2-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

## Register 12: GPIO 4-mA Drive Select (GPIODR4R), offset 0x504

The **GPIODR4R** register is the 4-mA drive control register. Each GPIO signal in the port can be individually configured without affecting the other pads. When setting the **DRV4** bit for a GPIO signal, the corresponding **DRV2** bit in the **GPIODR2R** register and **DRV8** bit in the **GPIODR8R** register are automatically cleared by hardware.

### GPIO 4-mA Drive Select (GPIODR4R)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0x504

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								DRV4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV4	R/W	0x00	Output Pad 4-mA Drive Enable

#### Value Description

- 1 The corresponding GPIO pin has 4-mA drive.
- 0 The drive for the corresponding GPIO pin is controlled by the **GPIODR2R** or **GPIODR8R** register.

Setting a bit in either the **GPIODR2** register or the **GPIODR8** register clears the corresponding 4-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

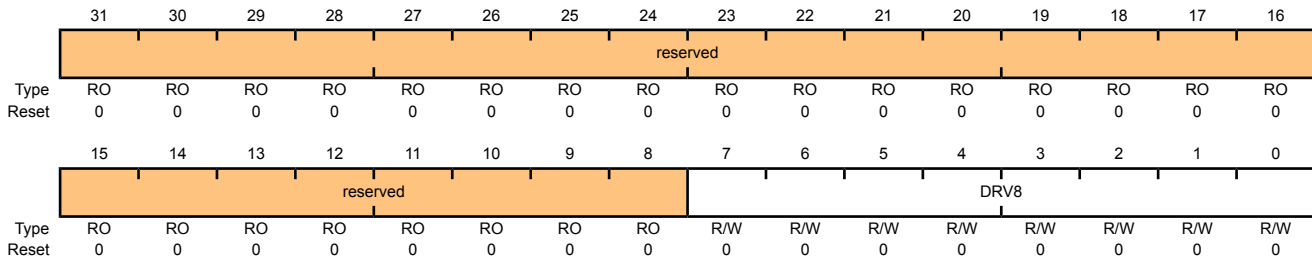
### Register 13: GPIO 8-mA Drive Select (GPIODR8R), offset 0x508

The **GPIODR8R** register is the 8-mA drive control register. Each GPIO signal in the port can be individually configured without affecting the other pads. When setting the **DRV8** bit for a GPIO signal, the corresponding **DRV2** bit in the **GPIODR2R** register and **DRV4** bit in the **GPIODR4R** register are automatically cleared by hardware. The 8-mA setting is also used for high-current operation.

**Note:** There is no configuration difference between 8-mA and high-current operation. The additional current capacity results from a shift in the  $V_{OH}/V_{OL}$  levels. See “Recommended DC Operating Conditions” on page 941 for further information.

#### GPIO 8-mA Drive Select (GPIODR8R)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0x508  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV8	R/W	0x00	Output Pad 8-mA Drive Enable

- Value Description
- 1 The corresponding GPIO pin has 8-mA drive.
  - 0 The drive for the corresponding GPIO pin is controlled by the **GPIODR2R** or **GPIODR4R** register.

Setting a bit in either the **GPIODR2** register or the **GPIODR4** register clears the corresponding 8-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.



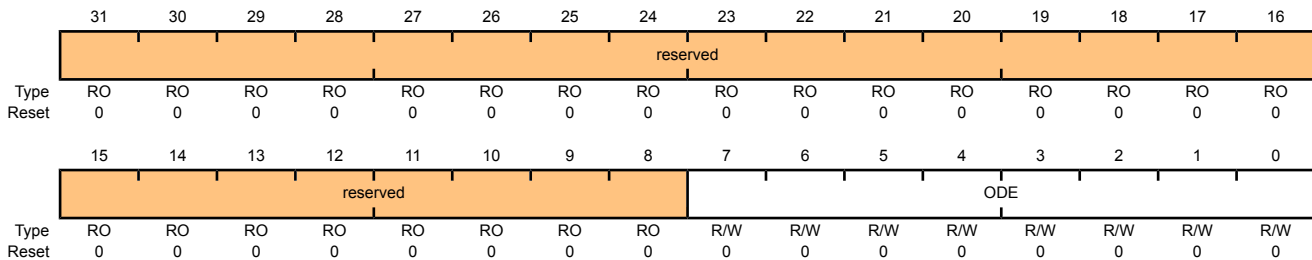
### Register 14: GPIO Open Drain Select (GPIODR), offset 0x50C

The **GPIODR** 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 314). Corresponding bits in the drive strength and slew rate control registers (**GPIODR2R**, **GPIODR4R**, **GPIODR8R**, and **GPIOSLR**) can be set to achieve the desired rise and fall times. The GPIO acts as an open-drain input if the corresponding bit in the **GPIODIR** register is cleared; and as an open-drain output when it is set.

When using the I<sup>2</sup>C module, in addition to configuring the pin to open drain, the **GPIO Alternate Function Select (GPIOAFSEL)** register bits for the I<sup>2</sup>C clock and data pins should be set (see examples in “Initialization and Configuration” on page 289).

#### GPIO Open Drain Select (GPIODR)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0x50C  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ODE	R/W	0x00	Output Pad Open Drain Enable

Value	Description
1	The corresponding pin is configured as open drain.
0	The corresponding pin is not configured as open drain.

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

The **GPIOPUR** register is the pull-up control register. When a bit is set, a weak pull-up resistor on the corresponding GPIO signal is enabled. Setting a bit in **GPIOPUR** automatically clears the corresponding bit in the **GPIO Pull-Down Select (GPIOPDR)** register (see page 312). Write access to this register is protected with the **GPIOCR** register. Bits in **GPIOCR** that are cleared prevent writes to the equivalent bit in this register.

**Important:** All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, and **GPIOPUR**=0) with the exception of the pins shown in Table 9-1. A Power-On-Reset ( $\overline{POR}$ ) or asserting  $\overline{RST}$  puts the pins back to their default state.

Table 9-7. GPIO Pins With Non-Zero Reset Values

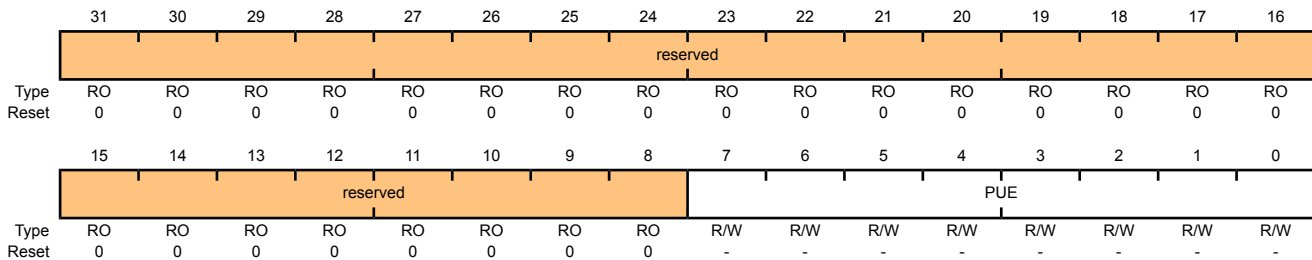
GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	1	1	0	0	0x1
PA[5:2]	SSI0	1	1	0	0	0x1
PB[3:2]	I <sup>2</sup> C0	1	1	0	0	0x1
PC[3:0]	JTAG	1	1	0	1	0x3

**Note:** The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the **NMI** pin (**PB7**) and the four **JTAG/SWD** pins (**PC[3:0]**). Writes to protected bits of the **GPIOAFSEL** register, **GPIOPUR** register, **GPIO Pull-Down Select (GPIOPDR)** register (see page 312), and **GPIODEN** register are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 316) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 317) have been set.

#### GPIO Pull-Up Select (GPIOPUR)

- GPIO Port A (APB) base: 0x4000.4000
- GPIO Port A (AHB) base: 0x4005.8000
- GPIO Port B (APB) base: 0x4000.5000
- GPIO Port B (AHB) base: 0x4005.9000
- GPIO Port C (APB) base: 0x4000.6000
- GPIO Port C (AHB) base: 0x4005.A000
- GPIO Port D (APB) base: 0x4000.7000
- GPIO Port D (AHB) base: 0x4005.B000
- GPIO Port E (APB) base: 0x4002.4000
- GPIO Port E (AHB) base: 0x4005.C000
- GPIO Port F (APB) base: 0x4002.5000
- GPIO Port F (AHB) base: 0x4005.D000
- GPIO Port G (APB) base: 0x4002.6000
- GPIO Port G (AHB) base: 0x4005.E000
- GPIO Port H (APB) base: 0x4002.7000
- GPIO Port H (AHB) base: 0x4005.F000
- GPIO Port J (APB) base: 0x4003.D000
- GPIO Port J (AHB) base: 0x4006.0000

Offset 0x510  
Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PUE	R/W	-	<p>Pad Weak Pull-Up Enable</p> <p>Value Description</p> <p>1 The corresponding pin has a weak pull-up resistor.</p> <p>0 The corresponding pin is not affected.</p> <p>Setting a bit in the <b>GPIOPDR</b> register clears the corresponding bit in the <b>GPIOPUR</b> register. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.</p> <p>The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 9-1 on page 285.</p>

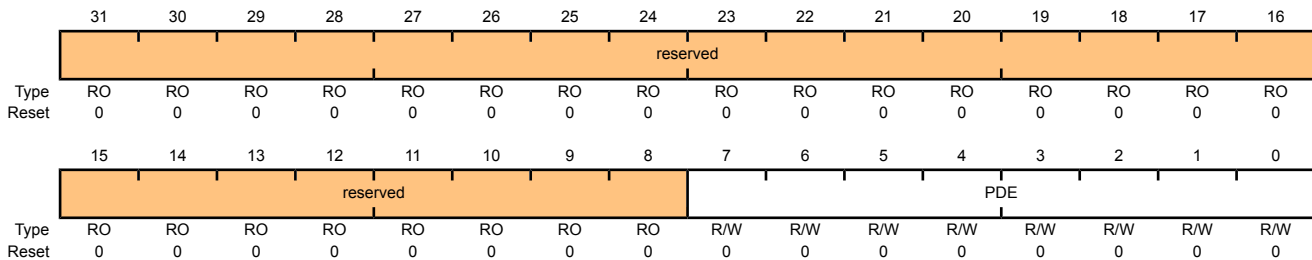
### Register 16: GPIO Pull-Down Select (GPIOPDR), offset 0x514

The **GPIOPDR** register is the pull-down control register. When a bit is set, a weak pull-down resistor on the corresponding GPIO signal is enabled. Setting a bit in **GPIOPDR** automatically clears the corresponding bit in the **GPIO Pull-Up Select (GPIOPUR)** register (see page 310).

**Note:** The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the **NMI** pin (**PB7**) and the four **JTAG/SWD** pins (**PC[3:0]**). Writes to protected bits of the **GPIOAFSEL** register, **GPIOPUR** register, **GPIO Pull-Down Select (GPIOPDR)** register (see page 312), and **GPIODEN** register are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 316) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 317) have been set.

#### GPIO Pull-Down Select (GPIOPDR)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0x514  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PDE	R/W	0x00	Pad Weak Pull-Down Enable

Value	Description
1	The corresponding pin has a weak pull-down resistor.
0	The corresponding pin is not affected.

Setting a bit in the **GPIOPUR** register clears the corresponding bit in the **GPIOPDR** register. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

## 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 308).

### GPIO Slew Rate Control Select (GPIOSLR)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0x518

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	SRL	R/W	0x00	Slew Rate Limit Enable (8-mA drive only)

#### Value Description

- 1 Slew rate control is enabled for the corresponding pin.
- 0 Slew rate control is disabled for the corresponding pin.

### Register 18: GPIO Digital Enable (GPIODEN), offset 0x51C

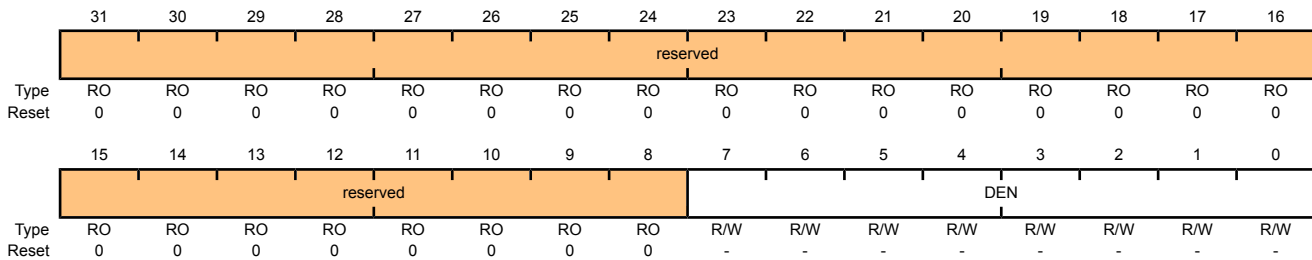
**Note:** Pins configured as digital inputs are Schmitt-triggered.

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

**Note:** The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the **NMI** pin (**PB7**) and the four JTAG/SWD pins (**PC[3:0]**). Writes to protected bits of the **GPIOAFSEL** register, **GIOPUR** register, **GPIO Pull-Down Select (GPIOPDR)** register (see page 312), and **GPIODEN** register are not committed to storage unless the **GPIO Lock (GPIOLCK)** register (see page 316) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 317) have been set.

#### GPIO Digital Enable (GPIODEN)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0x51C  
 Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

---

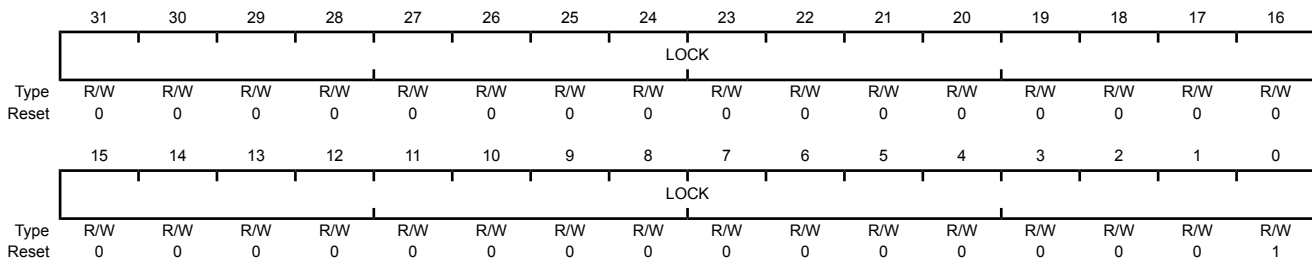
Bit/Field	Name	Type	Reset	Description
7:0	DEN	R/W	-	Digital Enable
				Value Description
				0 The digital functions for the corresponding pin are disabled.
				1 The digital functions for the corresponding pin are enabled.
				The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 9-1 on page 285.

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

The **GPIOLOCK** register enables write access to the **GPIOCR** register (see page 317). Writing 0x4C4F.434B to the **GPIOLOCK** register unlocks 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 0x0000.0001. When write accesses are enabled, or unlocked, reading the **GPIOLOCK** register returns 0x0000.0000.

#### GPIO Lock (GPIOLOCK)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0x520  
 Type R/W, reset 0x0000.0001



Bit/Field	Name	Type	Reset	Description
31:0	LOCK	R/W	0x0000.0001	GPIO Lock

A write of the value 0x4C4F.434B unlocks the **GPIO Commit (GPIOCR)** register for write access. A write of any other value or a write to the **GPIOCR** register reapplies the lock, preventing any register updates.

A read of this register returns the following values:

Value	Description
0x0000.0001	The <b>GPIOCR</b> register is locked and may not be modified.
0x0000.0000	The <b>GPIOCR</b> register is unlocked and may be modified.



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

The **GPIOCR** register is the commit register. The value of the **GPIOCR** register determines which bits of the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, and **GPIODEN** registers are committed when a write to these registers is performed. If a bit in the **GPIOCR** register is cleared, the data being written to the corresponding bit in the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** registers cannot be committed and retains its previous value. If a bit in the **GPIOCR** register is set, the data being written to the corresponding bit of the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** registers is committed to the register and reflects the new value.

The contents of the **GPIOCR** register can only be modified if the status in the **GPIOLCK** register is unlocked. Writes to the **GPIOCR** register are ignored if the status in the **GPIOLCK** register is locked.

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

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

### GPIO Commit (GPIOCR)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000

Offset 0x524  
 Type -, reset -

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

7:0	CR	-	-	GPIO Commit
-----	----	---	---	-------------

Value Description

- 1 The corresponding **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GIODEN** bits can be written.
- 0 The corresponding **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GIODEN** bits cannot be written.

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

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

## Register 21: GPIO Analog Mode Select (GPIOAMSEL), offset 0x528

**Important:** This register is only valid for ports D and E, the corresponding base addresses for the remaining ports are not valid.

If any pin is to be used as an ADC input, the appropriate bit in **GPIOAMSEL** must be set to disable the analog isolation circuit.

The **GPIOAMSEL** register controls isolation circuits to the analog side of a unified I/O pad. Because the GPIOs may be driven by a 5-V source and affect analog operation, analog circuitry requires isolation from the pins when they are not used in their analog function.

Each bit of this register controls the isolation circuitry for the corresponding GPIO signal. For information on which GPIO pins can be used for ADC functions, refer to Table 24-5 on page 937.

### GPIO Analog Mode Select (GPIOAMSEL)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0x528

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								GPIOAMSEL				reserved			
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
7:4	GPIOAMSEL	R/W	0x0	GPIO Analog Mode Select  Value Description 1 The analog function of the pin is enabled, the isolation is disabled, and the pin is capable of analog functions. 0 The analog function of the pin is disabled, the isolation is enabled, and the pin is capable of digital functions as specified by the other GPIO configuration registers.  <b>Note:</b> This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad.  The reset state of this register is 0 for all signals.
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 22: GPIO Port Control (GPIOCTL), offset 0x52C

The **GPIOCTL** register is used in conjunction with the **GPIOAFSEL** register and selects the specific peripheral signal for each GPIO pin when using the alternate function mode. Most bits in the **GPIOAFSEL** register are cleared upon reset, therefore most GPIO pins are configured as GPIOs by default. When a bit is set in the **GPIOAFSEL** register, the corresponding GPIO signal is controlled by an associated peripheral. The **GPIOCTL** register selects one out of a set of peripheral functions for each GPIO, providing additional flexibility in signal definition. For information on the configuration options, refer to Table 24-5 on page 937. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 9-1 on page 285.

**Important:** All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GIODEN**=0, **GPIOPDR**=0, and **GPIOPUR**=0) with the exception of the pins shown in Table 9-1. A Power-On-Reset ( $\overline{POR}$ ) or asserting  $\overline{RST}$  puts the pins back to their default state.

Table 9-8. GPIO Pins With Non-Zero Reset Values

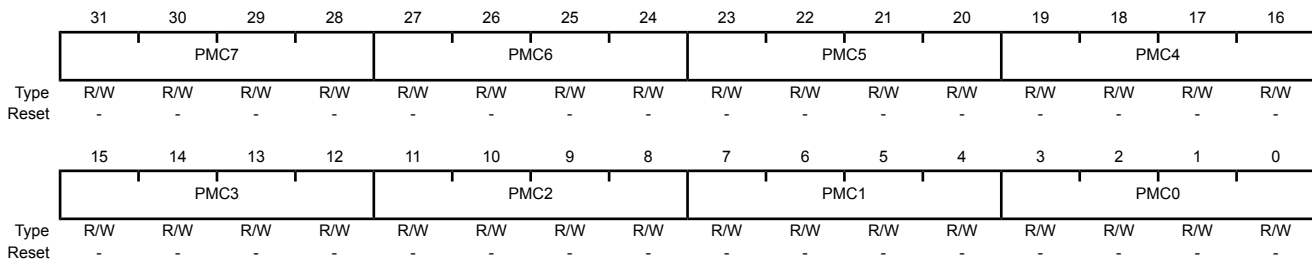
GPIO Pins	Default State	GPIOAFSEL	GIODEN	GPIOPDR	GPIOPUR	GPIOCTL
PA[1:0]	UART0	1	1	0	0	0x1
PA[5:2]	SSI0	1	1	0	0	0x1
PB[3:2]	I <sup>2</sup> C0	1	1	0	0	0x1
PC[3:0]	JTAG	1	1	0	1	0x3

### GPIO Port Control (GPIOCTL)

GPIO Port A (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000

Offset 0x52C

Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31:28	PMC7	R/W	-	Port Mux Control 7 This field controls the configuration for GPIO pin 7. See Table 24-5 on page 937 for configuration options.
27:24	PMC6	R/W	-	Port Mux Control 6 This field controls the configuration for GPIO pin 6.
23:20	PMC5	R/W	-	Port Mux Control 5 This field controls the configuration for GPIO pin 5.
19:16	PMC4	R/W	-	Port Mux Control 4 This field controls the configuration for GPIO pin 4.
15:12	PMC3	R/W	-	Port Mux Control 3 This field controls the configuration for GPIO pin 3.
11:8	PMC2	R/W	-	Port Mux Control 2 This field controls the configuration for GPIO pin 2.
7:4	PMC1	R/W	-	Port Mux Control 1 This field controls the configuration for GPIO pin 1.
3:0	PMC0	R/W	-	Port Mux Control 0 This field controls the configuration for GPIO pin 0.

**Register 23: 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 (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0xFD0

Type RO, reset 0x0000.0000



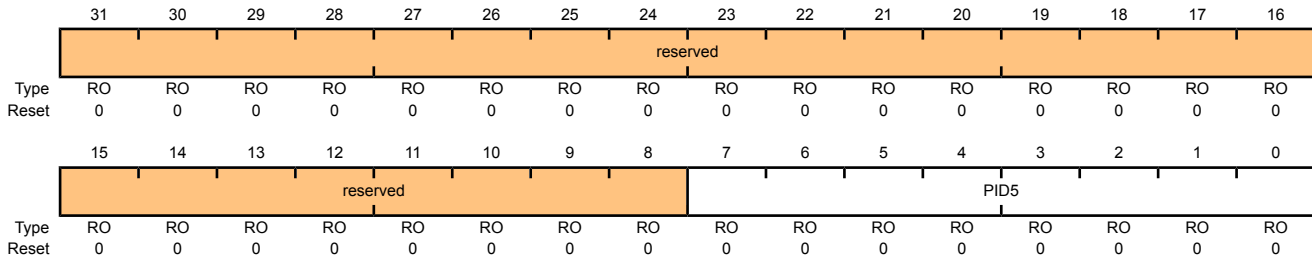
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	GPIO Peripheral ID Register [7:0]

### Register 24: 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 (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0xFD4  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	GPIO Peripheral ID Register [15:8]

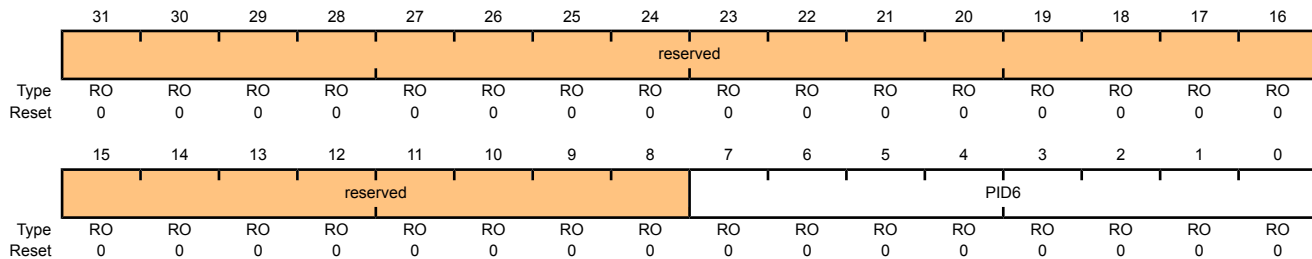


**Register 25: GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8**

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 (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0xFD8  
 Type RO, reset 0x0000.0000



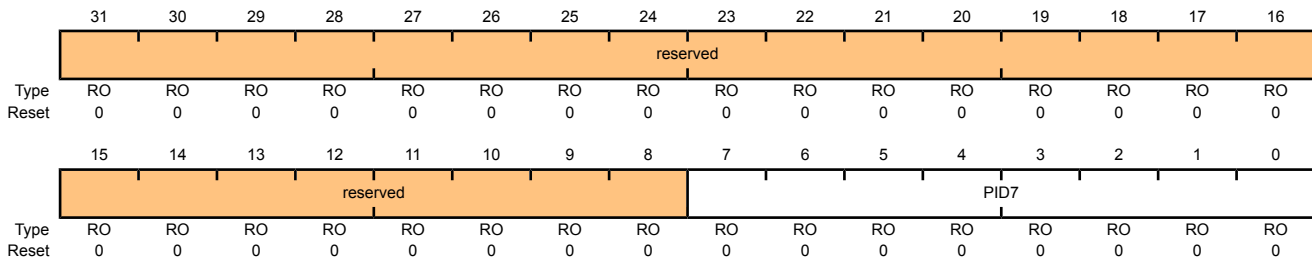
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	GPIO Peripheral ID Register [23:16]

**Register 26: 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 (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0xFDC  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	GPIO Peripheral ID Register [31:24]

**Register 27: 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 (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0xFE0  
 Type RO, reset 0x0000.0061

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1

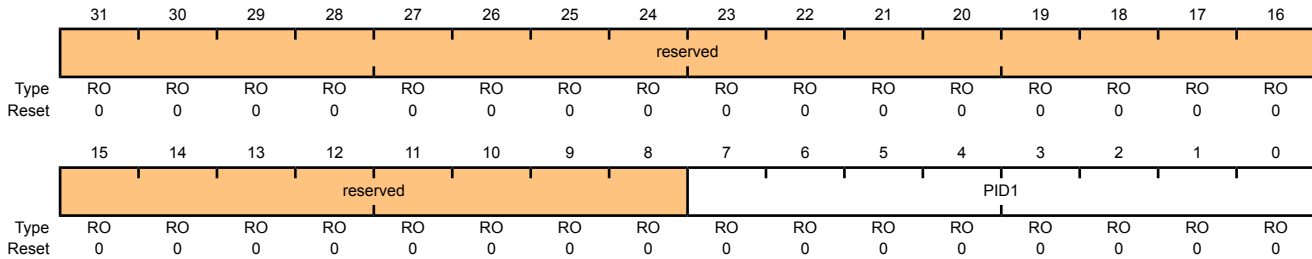
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x61	GPIO Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

### Register 28: 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 (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0xFE4  
 Type RO, reset 0x0000.0000



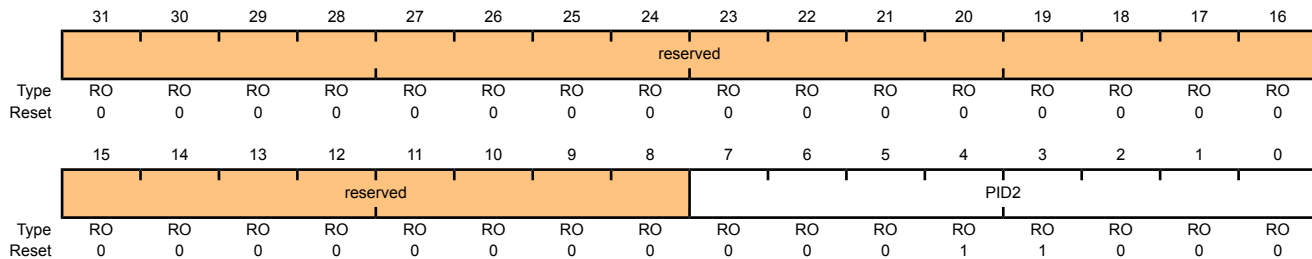
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	GPIO Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

**Register 29: 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 (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0xFE8  
 Type RO, reset 0x0000.0018



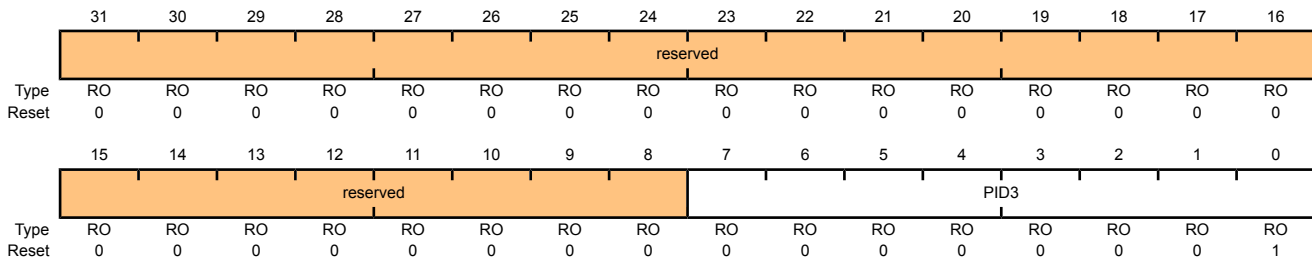
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	GPIO Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

### Register 30: 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 (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0xFEC  
 Type RO, reset 0x0000.0001



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	GPIO Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

## Register 31: 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 (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0xFF0  
 Type RO, reset 0x0000.000D

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

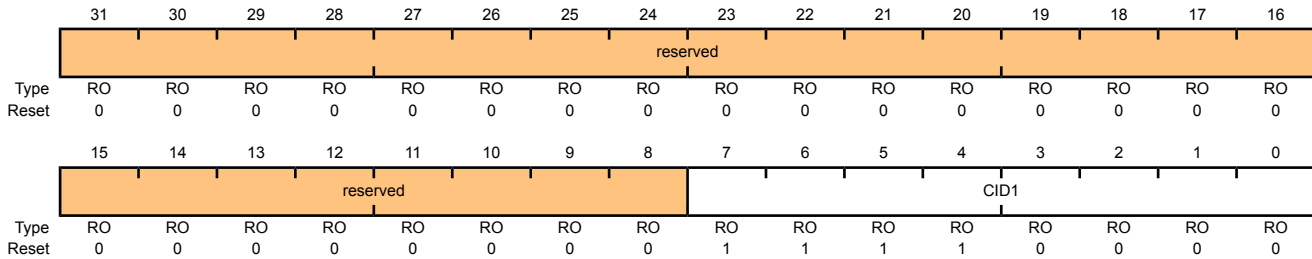
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	GPIO PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

### Register 32: 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 (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0xFF4  
 Type RO, reset 0x0000.00F0



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	GPIO PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system.

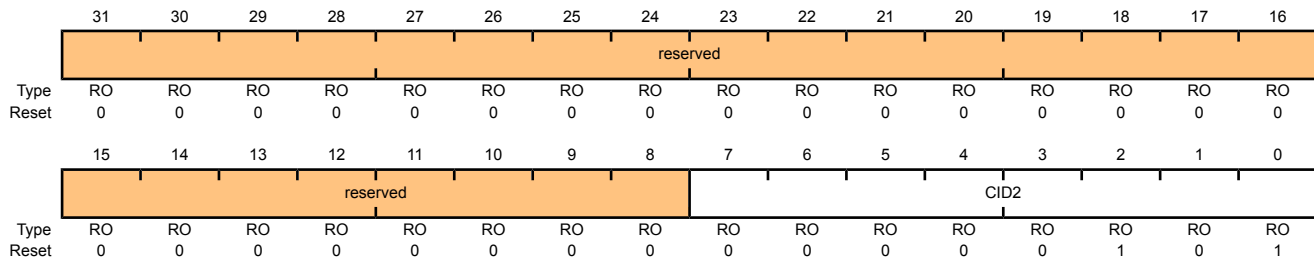


**Register 33: GPIO PrimeCell Identification 2 (GPIOCellID2), offset 0xFF8**

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 (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0xFF8  
 Type RO, reset 0x0000.0005



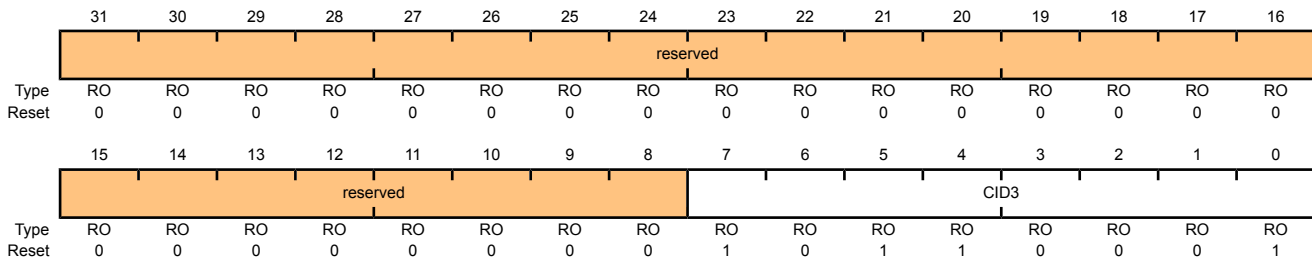
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	GPIO PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system.

### Register 34: 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 (APB) base: 0x4000.4000  
 GPIO Port A (AHB) base: 0x4005.8000  
 GPIO Port B (APB) base: 0x4000.5000  
 GPIO Port B (AHB) base: 0x4005.9000  
 GPIO Port C (APB) base: 0x4000.6000  
 GPIO Port C (AHB) base: 0x4005.A000  
 GPIO Port D (APB) base: 0x4000.7000  
 GPIO Port D (AHB) base: 0x4005.B000  
 GPIO Port E (APB) base: 0x4002.4000  
 GPIO Port E (AHB) base: 0x4005.C000  
 GPIO Port F (APB) base: 0x4002.5000  
 GPIO Port F (AHB) base: 0x4005.D000  
 GPIO Port G (APB) base: 0x4002.6000  
 GPIO Port G (AHB) base: 0x4005.E000  
 GPIO Port H (APB) base: 0x4002.7000  
 GPIO Port H (AHB) base: 0x4005.F000  
 GPIO Port J (APB) base: 0x4003.D000  
 GPIO Port J (AHB) base: 0x4006.0000  
 Offset 0xFFC  
 Type RO, reset 0x0000.00B1



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	GPIO PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system.

## 10 External Peripheral Interface (EPI)

The External Peripheral Interface is a high-speed parallel bus for external peripherals or memory. It has several modes of operation to interface gluelessly to many types of external devices. The External Peripheral Interface is similar to a standard microprocessor address/data bus, except that it must typically be connected to just one type of external device. Enhanced capabilities include  $\mu$ DMA support, clocking control and support for external FIFO buffers.

The EPI has the following features:

- 16-bit dedicated parallel bus for external peripherals and memory
- Memory interface supports contiguous memory access independent of data bus width, thus enabling code execution directly from SDRAM, SRAM and Flash memory
- Blocking and non-blocking reads
- Processor from timing details through use of an internal write FIFO
- Efficient transfers using Micro Direct Memory Access Controller ( $\mu$ DMA)
  - Separate channels for read and write
  - Read channel request asserted by programmable levels on the internal non-blocking read FIFO (NBRFIFO)
  - Write channel request asserted by empty on the internal write FIFO (WFIFO)

The EPI supports three primary functional modes: Synchronous Dynamic Random Access Memory (SDRAM) mode, Traditional Host-Bus mode, and General-Purpose mode. The EPI module also provides custom GPIOs; however, unlike regular GPIOs, the EPI module uses a FIFO in the same way as a communication mechanism and is speed-controlled using clocking.

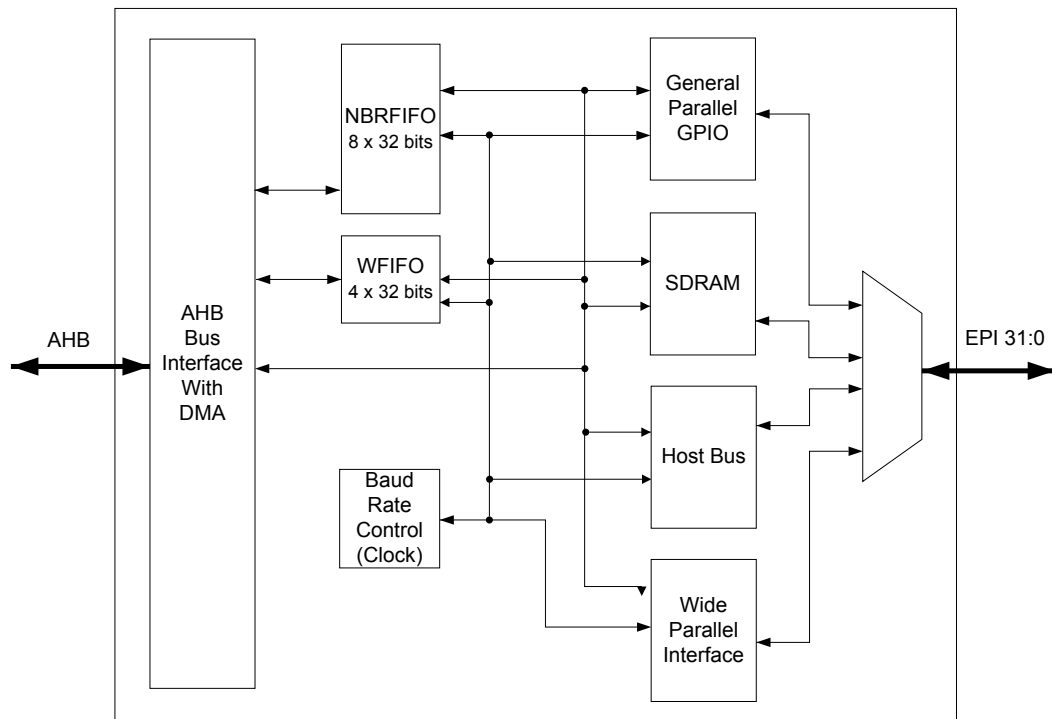
- Synchronous Dynamic Random Access Memory (SDRAM)
  - Supports x16 (single data rate) SDRAM at up to 50 MHz
  - Supports low-cost SDRAMs up to 64 MB (512 Mb)
  - Includes automatic refresh and access to all banks/rows
  - Includes a Sleep/Standby mode to keep contents active with minimal power draw
  - Multiplexed address/data interface for reduced pin count
- Host-bus
  - Traditional x8 MCU bus interface capabilities
  - Similar device compatibility options as PIC, ATmega, 8051, and others
  - Access to SRAM, NOR Flash, and other devices, with up to 1 MB of addressing
  - Support of both muxed and de-muxed address and data

- Access to a range of devices supporting the non-address FIFO x8 interface variant, with support for external FIFO (XFIFO) `EMPTY` and `FULL` signals
- Speed controlled, with read and write data wait-state counters
- Manual chip-enable (or use extra address pins)
- General Purpose
  - Wide parallel interfaces for fast communications with CPLDs and FPGAs
  - Data widths up to 32-bits
  - Data rates up to 150 Mbytes/second
  - Optional “address” sizes from 4-bits to 16-bits
  - Optional clock output, read/write strobes, framing (with counter-based size), and clock-enable input
- General parallel GPIO
  - 1 to 32 bits, FIFOed with speed control
  - Useful for custom peripherals or for digital data acquisition and actuator controls

## 10.1 EPI Block Diagram

Figure 10-1 on page 336 provides a block diagram of a Stellaris<sup>®</sup> EPI module.

**Figure 10-1. EPI Block Diagram**



## 10.2 Functional Description

The EPI controller provides a glueless, programmable interface to a variety of common external peripherals such as SDRAM, Host Bus x8 devices, RAM, Flash memory, CPLDs and FPGAs. In addition, the EPI controller provides custom GPIO that can use a FIFO with speed control by using either the internal write FIFO (WFIFO) or the non-blocking read FIFO (NBRFIFO). The WFIFO can hold 4 words of data that are written to the external interface at the rate controlled by the **EPI Main Baud Rate (EPIBAUD)** register. The NBRFIFO can hold 8 words of data and samples at the rate controlled by the **EPIBAUD** register. The advantage of this solution is that when using regular GPIO, the access rate can vary due arbitration to the GPIO module and delays across any bus bridges. Blocking reads stall the CPU until the transaction completes. Non-blocking reads are performed in the background and allow the processor to continue operation. In addition, write data can also be stored in the WFIFO to allow multiple writes with no stalls.

Main read and write operations can be performed in subsets of the range 0x6000.0000 to 0xCFFF.FFFF. A read from an address mapped location uses the offset and size to control the address and size of the external operation. When performing a multi-value load, the read is done as a burst (when available) to maximize performance. A write to an address mapped location uses the offset and size to control the address and size of the external operation. When performing a multi-value store, the write is done as a burst (when available) to maximize performance.

### 10.2.1 Non-blocking reads

The EPI Controller supports a special kind of read called a non-blocking read, also referred to as a posted read. Where a normal read stalls the processor or  $\mu$ DMA until the data is returned, a non-blocking read is performed in the background.

A non-blocking read is configured by writing the start address into a **EPIRADDRx** register, the size per transaction into a **EPIRSIZEx** register, and then the count of operations into a **EPIRPSTDx** register. After each read is completed, the result is written into the NBRFIFO and the **EPIRADDRx** register is incremented by the size (1, 2, or 4).

If the NBRFIFO is filled, then the reads pause until space is made available. The NBRFIFO can be configured to interrupt the processor or trigger the  $\mu$ DMA based on fullness using the **EPIFIFOLVL** register. By using the trigger/interrupt method, the  $\mu$ DMA (or processor) can keep space available in the NBRFIFO and allow the reads to continue unimpeded.

When performing non-blocking reads, the SDRAM controller issues two additional read transactions after the burst request is terminated. The data for these additional transfers is discarded. This situation is transparent to the user other than the additional EPI bus activity and can safely be ignored.

Two non-blocking read register sets are available to allow sequencing and ping-pong use. When one completes, the other then activates. So, for example, if 20 words are to be read from 0x100 and 10 words from 0x200, the **EPIRPSTD0** register can be set up with the read from 0x100 (with a count of 20), and the **EPIRPSTD1** register can be set up with the read from 0x200 (with a count of 10). When **EPIRPSTD0** finishes (count goes to 0), the **EPIRPSTD1** register then starts its operation. The NBRFIFO has then passed 30 values. When used with the  $\mu$ DMA, it may transfer 30 values (simple sequence), or the primary/alternate model may be used to handle the first 20 in one way and the second 10 in another. It is also possible to reload the **EPIRPSTD0** register when it is finished (and the **EPIRPSTD1** register is active); thereby, keeping the interface constantly busy.

To cancel a non-blocking read, the **EPIRPSTDx** register is cleared. Care must be taken, however if the register set was active to drain away any values read into the NBRFIFO and ensure that any read in progress is allowed to complete.

To ensure that the cancel is complete, the following algorithm is used (using the **EPIRPSTD0** register for example):

```
EPIRPSTD0 = 0;
while ((EPISTAT & 0x11) == 0x10)
; // we are active and busy
// if here, then other one is active or interface no longer busy
cnt = (EPIRADDR0 – original_address) / EPIRSIZ0E; // count of values read
cnt -= values_read_so_far;
// cnt is now number left in FIFO
while (cnt-->0)
value = EPIREADFIFO; // drain
```

The above algorithm can be optimized in code; however, the important point is to wait for the cancel to complete because the external interface could have been in the process of reading a value when the cancel came in, and it must be allowed to complete.

## 10.2.2 DMA Operation

The  $\mu$ DMA can be used to efficiently transfer data to and from the NBRFIFO and the WRFIFO. The  $\mu$ DMA has one channel for write and one for read. The write channel can be configured to copy values to the WRFIFO when the WRFIFO is empty. For non-blocking reads, the start address, the size per transaction, and the count of elements must be programmed in the  $\mu$ DMA. The NBRFIFO level at which the  $\mu$ DMA triggers the read accesses must also be programmed. Note that both non-blocking read channels can be used, and they fill the NBRFIFO such that one runs to completion, then the next one starts (they do not interleave). For blocking reads, any  $\mu$ DMA channel can be used as a memory-to-memory transfer (or memory to peripheral, where some other peripheral is used). In this situation, the  $\mu$ DMA is blocked when reading, thus the  $\mu$ DMA is not able to service another channel until the read is done. As a result, the arbitration size should normally be programmed to one access at a time. See “Micro Direct Memory Access ( $\mu$ DMA)” on page 219 for more information on configuring the  $\mu$ DMA.

## 10.3 Initialization and Configuration

To enable and initialize the EPI block, the following steps are necessary:

1. Enable the EPI block using the **RCGC1** register. See page 163.
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register. See page 175. To find out which GPIO port to enable, refer to Table 24-5 on page 937.
3. Set the GPIO **AFSEL** bits for the appropriate pins. See page 304. To determine which GPIOs to configure, see Table 24-5 on page 937.
4. Set the GPIO current level and/or slew rate as specified for the mode selected. See page 306 and page 313.
5. Select the mode for the EPI block to SDRAM, HB8, or general parallel use, using the **MODE** field in the **EPI Configuration (EPICFG)** register. Set the mode-specific details (if needed) using the appropriate mode configuration **EPI xxx Configuration (EPIxxxCFG)** and **EPI xxx**

**Configuration 2 (EPIxxxCFG2)** registers. Set the **EPI Main Baud Rate (EPIBAUD)** register if the baud rate must be slower than the core clock rate.

6. Configure the address mapping using the **EPI Address Map (EPIADDRMAP)** register. The selected start address and range is dependent on the type of external device and maximum address (as appropriate). For example, for a 512-MB SDRAM, program the `ERADR` field to 0x1 for address 0x60000000 or 0x2 for address 0x80000000; and program the `ERSZ` field to 0x3 for 512 MB. If using a non-mode and no address at all, program the `EPADR` field to 0x1 for address 0xA0000000 or 0x2 for address 0xC0000000; and program the `EPSZ` field to 0x0 for 256 bytes.
7. To read or write directly, use the mapped address area (configured with the **EPIADDRMAP** register). Up to 4 or 5 writes can be performed at once without blocking. Each read is blocked until the value is retrieved.
8. To perform a non-blocking read, see “Non-blocking reads” on page 337.

The following sub-sections describe the initialization and configuration for each of the modes of operation. Care must be taken to initialize everything properly to ensure correct operation. Control of the GPIO states is also important, as changes may cause the external device to interpret pin states as actions or commands (see “Register Descriptions” on page 293). Normally, a pull-up or pull-down is needed on the board to at least control the chip-select or chip-enable as the Stellaris® GPIOs come out of reset in tri-state.

The Table 10-1 on page 339 table defines how EPI module signals should be connected to various external peripherals. The table applies when using a x16 SDRAM up to 512 MB.

**Table 10-1. EPI Signal Connections**

EPI Signal	SDRAM Signal <sup>a</sup>		Host Bus 8 Signal (MODE =0x0 in EPIHB8CFG register)	Host Bus 8 Signal (MODE =0x1 in EPIHB8CFG register)	Host Bus 8 Signal (MODE =0x3 in EPIHB8CFG register)	General-Purpose Signal (D8, A20)	General- Purpose Signal (D16, A12)	General- Purpose Signal (D24, A4)	General- Purpose Signal (D32)
EPI0	A0	D0	AD0	D0	D0	D0	D0	D0	D0
EPI1	A1	D1	AD1	D1	D1	D1	D1	D1	D1
EPI2	A2	D2	AD2	D2	D2	D2	D2	D2	D2
EPI3	A3	D3	AD3	D3	D3	D3	D3	D3	D3
EPI4	A4	D4	AD4	D4	D4	D4	D4	D4	D4
EPI5	A5	D5	AD5	D5	D5	D5	D5	D5	D5
EPI6	A6	D6	AD6	D6	D6	D6	D6	D6	D6
EPI7	A7	D7	AD7	D7	D7	D7	D7	D7	D7
EPI8	A8	D8	A8	A0	-	A0	D8	D8	D8
EPI9	A9	D9	A9	A1	-	A1	D9	D9	D9
EPI10	A10	D10	A10	A2	-	A2	D10	D10	D10
EPI11	A11	D11	A11	A3	-	A3	D11	D11	D11
EPI12	A12 <sup>b</sup>	D12	A12	A4	-	A4	D12	D12	D12
EPI13	BA0	D13	A13	A5	-	A5	D13	D13	D13
EPI14	BA1	D14	A14	A6	-	A6	D14	D14	D14
EPI15	D15		A15	A7	-	A7	D15	D15	D15

EPI Signal	SDRAM Signal <sup>a</sup>	Host Bus 8 Signal (MODE =0x0 in EPIHB8CFG register)	Host Bus 8 Signal (MODE =0x1 in EPIHB8CFG register)	Host Bus 8 Signal (MODE =0x3 in EPIHB8CFG register)	General-Purpose Signal (D8, A20)	General-Purpose Signal (D16, A12)	General-Purpose Signal (D24, A4)	General-Purpose Signal (D32)
EPI16	DQM0	A16	A8	-	A8	A1 <sup>c</sup>	D16	D16
EPI17	DQM1	A17	A9	-	A9	A2	D17	D17
EPI18	CASn	A18	A10	-	A10	A3	D18	D18
EPI19	RASn	A19	A11	-	A11	A4	D19	D19
EPI20	-	A20	A12	-	A12	A5	D20	D20
EPI21	-	A21	A13	-	A13	A6	D21	D21
EPI22	-	A22	A14	-	A14	A7	D22	D22
EPI23	-	A23	A15	-	A15	A8	D23	D23
EPI24	-	A24	A16	-	A16	A9	A2 <sup>d</sup>	D24
EPI25	-	A25	A17	-	A17	A10	A3	D25
EPI26	-	A26	A18	FEMPTY	A18	A11	A4	D26
EPI27	-	A27	A19	FFULL	A19/iRDY <sup>e</sup>	A12/iRDY <sup>e</sup>	A5/iRDY <sup>e</sup>	D27
EPI28	WEn	RDn/OEn	RDn/OEn	RDn	WR	WR	WR	D28
EPI29	CSn	WRn	WRn	WRn	RD	RD	RD	D29
EPI30	CKE	ALE <sup>f</sup>	CSn <sup>g</sup>	-	Frame	Frame	Frame	D30
EPI31	CLK	Clock	Clock	Clock	Clock	Clock	Clock	D31

a. If 2 signals are listed, connect the EPI signal to both pins.

b. Only for 256/512 Mb SDRAMs

c. A1 represents the system address bit 1 for 16-bit data access. If this signal is connected to a device that only has 16-bit data access, then EPI16 should be connected to A0. EPI[27:17] should also be connected to A[11:1] in this case.

d. A2 represents the system address bit 2 for 32-bit data access. If this signal is connected to a device that only has 24-bit data access then EPI24 should be connected to A0. EPI[27:25] should also be connected to A[3:1] in this case.

e. This signal is iRDY if the RDYEN bit in the EPIGPCFG register is set.

f. The CSCFG field in the EPIHB8CFG2 register should be configured to 0x0. This option creates an ALE pulse during the address cycle preceding the read/write cycle.

g. The CSCFG field in the EPIHB8CFG2 register should be configured to 0x1. This option creates a CSn that is active during the read/write cycle.

### 10.3.1 SDRAM mode

When activating the SDRAM mode, it is important to consider a few points:

1. Generally, it takes over 100  $\mu$ s from when the mode is activated to when the first operation is allowed. The SDRAM controller begins the SDRAM initialization sequence as soon as the mode is selected and enabled via the EPIPCFG register. It is important that the GPIOs are properly configured before the SDRAM mode is enabled, as the EPI Controller is relying on the GPIO block's ability to drive the pins immediately. As part of the initialization sequence, the LOAD MODE REGISTER command is automatically sent to the SDRAM with a value of 0x27, which sets a CAS latency of 2 and a full page burst length.
2. The INITSEQ bit in the EPI Status (EPISTAT) register can be checked to determine when the initialization sequence is complete.
3. When using a frequency range and/or refresh value other than the default value, It is important to configure the FREQ and RFSH fields in the EPI SDRAM Configuration (EPISDRAMCFG)



register shortly after activating the mode. After the 100- $\mu$ s startup time, the EPI block must be configured properly to keep the SDRAM contents stable.

4. The SLEEP bit in the **EPISDRAMCFG** register may be configured to put the SDRAM into a low-power self-refreshing state. It is important to note that the SDRAM mode must not be disabled once enabled, or else the SDRAM is no longer be clocked and the contents are lost.

The SIZE field of the **EPISDRAMCFG** register must be configured correctly based on the amount of SDRAM in the system.

The FREQ field must be configured according to the value that represents the range being used. Based on the range selected, the number of external clocks used between certain operations (for example, PRECHARGE or ACTIVATE) is determined. If a higher frequency is given than is used, then the only downside is that the peripheral is slower (uses more cycles for these delays). If a lower frequency is given, incorrect operation occurs.

The refresh count is based on the external clock speed and the number of rows per bank as well as the refresh period. The RFSH field represents how many external clock cycles remain before an AUTO-REFRESH is required. The normal formula is:

$$\text{RFSH} = (\text{tRefresh}_{\mu\text{s}} / \text{number\_rows}) / \text{ext\_clock\_period}$$

A refresh period is normally 64 ms, or 64000  $\mu$ s. The number of rows is normally 4096 or 8192. The ext\_clock\_period is a value expressed in  $\mu$ sec and is derived by dividing 1000 by the clock speed expressed in MHz. So, 50 MHz is 1000/50=20 ns, or 0.02  $\mu$ s. A typical SDRAM is 4096 rows per bank if the core clock is running at 50 MHz with an **EPIBAUD** register value of 0:

$$\text{RFSH} = (64000/4096) / 0.02 = 15.625 \mu\text{s} / 0.02 \mu\text{s} = 781.25$$

The default value in the RFSH field is 750 decimal or 0x2EE to allow for a margin of safety and providing 15  $\mu$ s per refresh. It is important to note that this number should always be smaller or equal to what is required by the above equation. For example, if running the external clock at 25 MHz (40 ns per clock period), 390 is the highest number that may be used. Note that the external clock may be 25 MHz when running the core at 25 MHz or when running the core at 50 MHz and setting the **EPIBAUD** register to 1 (divide by 2).

If a number larger than allowed is used, the SDRAM is not refreshed often enough, and data is lost.

See "External Peripheral Interface (EPI)" on page 952 for timing details for the SDRAM mode.

## 10.3.2 Host Bus Mode

Host Bus supports the traditional 8-bit interface popularized by the 8051 devices. This interface is asynchronous and uses strobe pins to control activity.

### 10.3.2.1 Control Pins

The main three strobes are ALE (Address latch enable), WRn (write), and RDn (sometimes called OEn, used for read). Note that the timings are designed for older logic and so are hold-time vs. setup-time specific. To ensure proper operation on this bus, the EPI block uses two core clocks per transition to allow significant skewing of control vs. data signals. So, for example, ALE rises one EPI clock before ADDR/DA is asserted. Likewise, ALE falls (latch point) one EPI clock before DA changes or tri-states. The same approach is used for the WRn and RDn/OEn strobes.

The ALE can be changed to CSn through the **EPI Host-Bus Configuration 2 (EPIHB8CFG2)** register. The ALE is best used for Host-Bus muxed mode in which EPI address and data pins are shared. All Host-Bus accesses have an address phase then a data phase. The ALE indicates to an

external latch to capture the address then hold until the data phase. CSn is best used for Host-Bus unmuxed mode in which EPI address and data pins are separate. The CSn indicate when the address and data phases of a read or write access is occurring.

For FIFO mode, the ALE is not used, and two input holds are optionally supported to gate input and output to what the XFIFO can handle.

### 10.3.2.2 Speed of Transactions

The COUNT field **EPIBAUD** must be configured to set the main transaction rate based on what the slave device can support (including wiring considerations). The main control transitions are normally  $\frac{1}{2}$  the baud rate (COUNT = 1) because the EPI block forces data vs. control to change on alternating clocks.

Additionally, the Host Bus mode provides read and write wait states for the data portion to support different classes of device. These wait states stretch the data period (hold the rising edge of data strobe) and may be used in all four sub-modes. The wait states are set using the WRWS and RDWS bits in the **EPI Host-Bus 8 Configuration (EPIHB8CFG)** register.

### 10.3.2.3 Sub-Modes of Host Bus 8

The EPI controller supports four variants of the host bus model using 8 bits of data in all four cases. The four sub-modes are selected using the MODE bits in the **EPIHB8CFG** register, and are:

1. Address and data are muxed (address and data share EPI[7:0] with additional address at EPI[19:8]). This scheme is used by many 8051 devices, some Microchip PIC parts, and some ATmega parts. When used for standard SRAMs, a latch must be used between the microcontroller and the SRAM. This sub-mode is provided for compatibility with existing devices that support data transfers without a latch (for example, LCD controllers or CPLDs). In general, the de-muxed sub-mode should normally be used. The ALE configuration should be used in this mode, as all Host-Bus accesses have an address phase followed by a data phase. The ALE indicates to an external latch to capture the address then hold until the data phase. The ALE configuration is controlled by configuring the CSCFG field to be 0x0 in the **EPIHB8CFG2** register. The CSn is best used for Host-Bus 8 unmuxed mode which EPI address and data pins are separate. The CSn will indicate when the address and data phases of a read or write access is occurring.
2. Address and data are separate with 8 bits of data and up to 20 bits of address (1MB). This scheme is used by more modern 8051 devices, as well as some PIC and ATmega parts. This mode is generally used with real SRAMs, many EEPROMS, and many NOR Flash memory devices. Note that there is no hardware command write support for Flash memory devices; this mode should only be used for Flash memory devices programmed at manufacturing time. If a Flash memory device must be written and does not support a direct programming model, the command mechanism must be performed in software. The CSn configuration should be used in this mode. The CSn signals indicate when the address and data phases of a read or write access is occurring. The CSn configuration is controlled by configuring the CSCFG field to be 0x1 in the **EPIHB8CFG2** register.
3. SRAM fast mode where address and data are separate. This sub-mode is used for real SRAMs which can be read more quickly by only changing the address (and not using RDn/OEn strobing).
4. FIFO mode uses 8 bits of data, removes ALE and address pins and optionally adds external XFIFO FULL/EMPTY flag inputs. This scheme is used by many devices, such as radios, communication devices (including USB2 devices), and some FPGA configurations (FIFO through block RAM). This sub-mode provides the data side of the normal Host Bus interface, but is

paced by the FIFO control signals. It is important to consider that the XFIFO FULL/EMPTY control signals may stall the interface and could have an impact on blocking read latency from the processor or  $\mu$ DMA.

See “External Peripheral Interface (EPI)” on page 952 for timing details for the Host-Bus 8 mode.

### 10.3.3 General-Purpose Mode

The **General-Purpose Mode Configuration (EPIGPCFG)** register is used to control the size of control, data, and address pins, if used. The general-purpose configuration can be used for custom interfaces with FPGAs, CPLDs, and digital data acquisition and actuator control.

It is designed for three general types of use:

- Extremely high-speed clocked interfaces to FPGAs and CPLDs. Three sizes of data and optional address are supported. Framing and clock-enable functions permit more optimized interfaces.
- General parallel GPIO. From 1 to 31 pins may be written or read, with the speed controlled by the **EPIBAUD** register baud rate (when used with the WFIFO and/or the NBRFIFO) or by the rate of accesses from software or  $\mu$ DMA.
- General custom interfaces of any speed.

The configuration allows for choice of an output clock (free-running or gated), a framing signal (with frame size), a clock-enable input (to stretch transactions), a READ and WRITE strobe, an address (of varying sizes), and data (of varying sizes). Additionally, provisions are made for separating data and address phases.

To understand the interface’s possibilities, it is important to understand the optional features:

- Use of output clock or not (controlled by the **CLKPIN** bit in the **EPIGPCFG** register). Unlocked uses include general purpose I/O and asynchronous interfaces (optionally using READ and WRITE strobes). Clocked interfaces allow for higher speeds and are much easier to connect to FPGAs and CPLDs (which usually include input clocks).
- Clock, if used, may be free running or gated (using the **CLKGATE** bit in the **EPIGPCFG** register). A free-running clock requires another method for determining when data is live, such as the frame pin or READ/WRITE strobes. A gated clock approach uses a setup time model in which the clock controls when transactions are starting and stopping. Note that a gated clock can only be used when the **EPIBAUD** register has a value other than 0 (meaning the output clock is less than the core clock). The gated clock is held low until a new transaction is started and goes high at the end of the cycle where READ/WRITE/FRAME and address (and data if write) are emitted.
- Clock-enable input (iRDY) from the external device (controlled by the **RDYEN** bit in the **EPIGPCFG** register). The clock-enable signal uses **EPI27** and may only be used with a free-running clock. **RDYEN** gates transactions, no matter what state they are in. In addition, **RDYEN** is registered internally and holds the transaction state across multiple clocks if clock-disabled. Generally, **RDYEN** should be changed before the falling edge of the external clock. If the **EPIBAUD** register is 0, an external device can stretch the current state by clearing the **RDYEN** bit.
- Frame pin (controlled by the **FRMPIN** bit in the **EPIGPCFG** register). The frame pin may be used whether the clock is output or not, and whether the clock is free running or not. It may also be used along with the clock-enable. The frame may be a pulse (one clock) or may be 50/50 split across the frame size (controlled by the **FRM50** bit in the **EPIGPCFG** register). The frame count (the size of the frame as specified by the **FRMCNT** field in the **EPIGPCFG** register) may be between

1 and 15 clocks for pulsed and between 2 and 30 clocks for 50/50. The frame pin counts transactions and not clocks; a transaction is any clock where the READ or WRITE strobe is high (if used). So, if the `FRMCNT` bit is set, then the frame pin pulses every other transaction; if 2-cycle reads and writes are used, it pulses every other address phase. `FRM50` must be used with this in mind as it may hold state for many clocks waiting for the next transaction.

- READ and WRITE strobes may be used (controlled by the `RW` bit in the **EPIGPCFG** register). For interfaces where the direction is known (in advance, related to frame size, or other means), these strobes are not needed. For most other interfaces, READ and WRITE are used so the external peripheral knows what transaction is taking place, and if any transaction is taking place. READ is used in conjunction with separating the address and data phases (2-cycle mode), as explained below.
- Separation of address/request and data phases may be used on reads and writes using the `WR2CYC` and `RD2CYC` bits in the **EPIGPCFG** register. This configuration allows the external peripheral extra time to act and is more commonly used on reads. When configured to use an address as specified by the `ASIZE` field in the **EPIGPCFG** register, the address is emitted on the READ cycle (first cycle) and data is expected to be returned on the next cycle (when READ is not asserted). If no address is used, then READ is asserted on the first cycle and data is captured on the second cycle (when READ is not asserted), allowing more setup time for data. If single-cycle reads are used, then data is expected to be available on the same cycle as READ using the specified setup time. To use single-cycle reads, the external peripheral must have either fast combinatorial logic (relative to clock period) or must be able to setup the data in advance.

For writes, the output may be in one or two cycles. In the two-cycle case, the address (if any) is emitted on the first cycle with WRITE and the data is emitted on the second cycle (with WRITE not asserted). Although split address and write data phases are not normally needed for logic reasons, it may be useful to make read and write timings match. If 2-cycle reads or writes are used, the `RW` bit is automatically set.

- Address may be emitted (controlled by the `ASIZE` field in the **EPIGPCFG** register). The address may be 4 bits (16 possible values), 12 bits (4096 possible values), or 20 bits (1 M possible values). Size of address limits size of data, for example, 4 bits of address supports 20 bits data in non-multiplex mode. Address comes from the bottom bits of the address used for the transaction by the processor or  $\mu$ DMA. The address signals may be used by the external peripheral as an address, code (command), or for other unrelated uses (such as a chip enable).
- Data may be 8 bits, 16 bits, 24 bits, or 32 bits (controlled by the `DSIZE` field in the **EPIGPCFG** register). 32-bit data cannot be used with address or clock or any other signal. 24-bit data can only be used with 4-bit address or no address. 32-bit data requires that either the `WR2CYC` bit or the `RD2CYC` bit in the **EPIGPCFG** register is set.
- When using the EPI as a GPIO interface, writes are FIFOed (up to 4 can be held at any time), and up to 32 pins are changed using the **EPIBAUD** clock rate. So, output pin control can be very precisely controlled as a function of time. By contrast, when writing to normal GPIOs, writes can only occur 8-bits at a time and take up to two clock cycles to complete. In addition, the write itself may be further delayed by the bus due to DMA or draining of a previous write. With both GPIO and EPI, reads may be performed directly, in which case the current pin states are read back. With EPI, the non-blocking interface may also be used to perform reads based on a fixed time rule via the **EPIBAUD** clock rate.

See “External Peripheral Interface (EPI)” on page 952 for timing details for the General-Purpose mode.

## 10.4 Register Map

Table 10-2 on page 345 lists the EPI registers. The offset listed is a hexadecimal increment to the register's address, relative to the base address of 0x400D.0000. Note that the EPI controller clock must be enabled before the registers can be programmed (see page 163).

**Note:** A back-to-back write followed by a read of the same register reads the value that written by the first write access, not the value from the second write access. (This situation only occurs when the processor core attempts this action, the  $\mu$ DMA does not do this.). To read back what was just written, another instruction must be generated between the write and read. Read-write does not have this issue, so use of read-write for clear of error interrupt cause is not affected.

**Table 10-2. External Peripheral Interface (EPI) Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	EPICFG	R/W	0x0000.0000	EPI Configuration	347
0x004	EPIBAUD	R/W	0x0000.0000	EPI Main Baud Rate	348
0x010	EPISDRAMCFG	R/W	0x42EE.0000	EPI SDRAM Configuration	349
0x010	EPIHB8CFG	R/W	0x0000.FF00	EPI Host-Bus 8 Configuration	351
0x010	EPIGPCFG	R/W	0x0000.FF00	EPI General-Purpose Configuration	355
0x014	EPIHB8CFG2	R/W	0x0000.0000	EPI Host-Bus 8 Configuration 2	359
0x014	EPIGPCFG2	R/W	0x0000.0000	EPI General-Purpose Configuration 2	361
0x01C	EPIADDRMAP	R/W	0x0000.0000	EPI Address Map	362
0x020	EPIRSIZE0	R/W	0x0000.0003	EPI Read Size 0	364
0x024	EPIRADDR0	R/W	0x0000.0000	EPI Read Address 0	365
0x028	EPIRPSTD0	R/W	0x0000.0000	EPI Non-Blocking Read Data 0	366
0x030	EPIRSIZE1	R/W	0x0000.0003	EPI Read Size 1	364
0x034	EPIRADDR1	R/W	0x0000.0000	EPI Read Address 1	365
0x038	EPIRPSTD1	R/W	0x0000.0000	EPI Non-Blocking Read Data 1	366
0x060	EPISTAT	R	0x0000.0000	EPI Status	368
0x06C	EPIRFIFOCNT	R	-	EPI Read FIFO Count	370
0x070	EPIREADFIFO	R	0x0000.0000	EPI Read FIFO	371
0x074	EPIREADFIFO1	R	0x0000.0000	EPI Read FIFO Alias 1	371
0x078	EPIREADFIFO2	R	0x0000.0000	EPI Read FIFO Alias 2	371
0x07C	EPIREADFIFO3	R	0x0000.0000	EPI Read FIFO Alias 3	371
0x080	EPIREADFIFO4	R	0x0000.0000	EPI Read FIFO Alias 4	371
0x084	EPIREADFIFO5	R	0x0000.0000	EPI Read FIFO Alias 5	371
0x088	EPIREADFIFO6	R	0x0000.0000	EPI Read FIFO Alias 6	371
0x08C	EPIREADFIFO7	R	0x0000.0000	EPI Read FIFO Alias 7	371

Offset	Name	Type	Reset	Description	See page
0x200	EPIFIFOLVL	R/W	0x0000.0033	EPI FIFO Level Selects	372
0x204	EPIWFIFOCNT	R	0x0000.0000	EPI Write FIFO Count	374
0x210	EPIIM	R/W	0x0000.0000	EPI Interrupt Mask	375
0x214	EPIRIS	R	0x0000.0000	EPI Raw Interrupt Status	376
0x218	EPIMIS	R	0x0000.0000	EPI Masked Interrupt Status	378
0x21C	EPIEISC	R/W1C	0x0000.0000	EPI Error Interrupt Status and Clear	379

## 10.5 Register Descriptions

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

## Register 1: EPI Configuration (EPICFG), offset 0x000

**Important:** The programming of the `MODE` field determines which configuration register is accessed for offsets 0x010 and 0x014. Any write to the EPICFG register resets the register contents at offsets 0x010 and 0x014.

The configuration register is used to enable the block, select a mode, and select the basic pin use (based on the mode). Note that attempting to program an undefined `MODE` field clears the `BLKEN` bit and disables the EPI controller.

### EPI Configuration (EPICFG)

Base 0x400D.0000

Offset 0x000

Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved												BLKEN	MODE			
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
31:5	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	BLKEN	R/W	0	Block Enable Setting this bit enables the EPI Controller.
3:0	MODE	R/W	0x0	Mode Select
	Value	Description		
	0x0	General Purpose		General-Purpose mode. Control, address, and data pins are configured using the <b>EPIGPCFG</b> and <b>EPIGPCFG2</b> registers.
	0x1	SDRAM		Supports SDR SDRAM. Control, address, and data pins are configured using the <b>EPISDRAMCFG</b> register.
	0x2	8-Bit Host-Bus (HB8)		Host-bus 8-bit interface (also known as the MCU interface). Control, address, and data pins are configured using the <b>EPIHB8CFG</b> and <b>EPIHB8CFG2</b> registers.
	0x3-0xF	Reserved		

### Register 2: EPI Main Baud Rate (EPIBAUD), offset 0x004

The main core clock is used internally to the EPI Controller. The baud rate counter can be used to divide the core clock down to control the speed on the external interface. If the mode selected emits an external clock, this register defines the clock emitted. If the mode selected does not use a clock, this register controls the speed of changes on the external interface. Care must be taken to program this register properly so that the speed of the external bus corresponds to the speed of the external peripheral and puts acceptable current load on the pins.

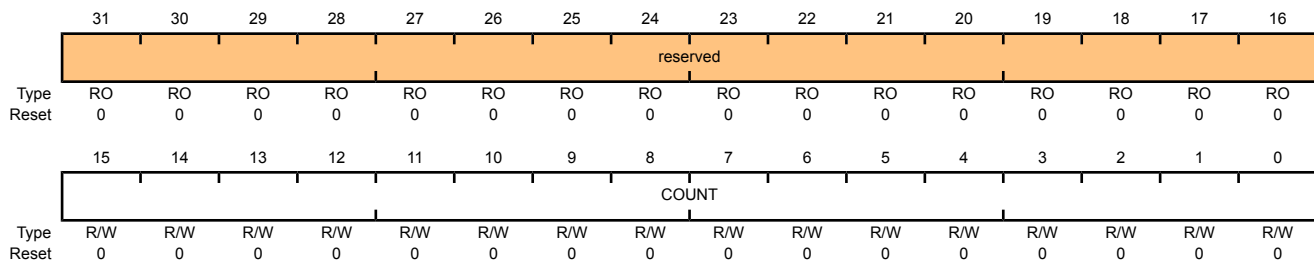
The COUNT is not a straight divider or count, but is instead calculated using the following formula:

$$\frac{1}{\left(\frac{\text{COUNT}}{2} + 1\right) \times 2}$$

So, for example, a COUNT of 0x0001 results in a clock rate of 1/2(core clock); a COUNT of 0x0002 or 0x0003 results in a clock rate of 1/4(core clock).

#### EPI Main Baud Rate (EPIBAUD)

Base 0x400D.0000  
 Offset 0x004  
 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	COUNT	R/W	0x0000	Baud Rate Counter  This bit field contains a counter used to divide the system clock by the count. The maximum frequency for the external baud clock is 40 MHz.  A count of 0 means the system clock is used as is.



### Register 3: EPI SDRAM Configuration (EPISDRAMCFG), offset 0x010

**Important:** To access this register, the `MODE` field in the `EPICFG` register must be 0x1.

The SDRAM Configuration register is used to specify several parameters for the SDRAM controller. Note that this register is reset when the `MODE` field in the `EPICFG` register is changed. If another mode is selected and the SDRAM mode is selected again, the values must be reinitialized.

The SDRAM interface designed to interface to x16 SDR SDRAMs of 64 MHz or higher, with the address and data pins overlapped (wire ORed on the board). See Table 10-1 on page 339 for pin assignments.

#### EPI SDRAM Configuration (EPISDRAMCFG)

Base 0x400D.0000

Offset 0x010

Type R/W, reset 0x42EE.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	FREQ		reserved			RFSH										
Type	R/W	R/W	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	1	0	0	0	0	1	0	1	1	1	0	1	1	1	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved						SLEEP	reserved							SIZE	
Type	RO	RO	RO	RO	RO	RO	R/W	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															
31:30	FREQ	R/W	0x1	<p>Frequency Range</p> <p>Frequency range of core clock. This field must be configured correctly to ensure proper operation. This field does not affect the refresh counting, which is configured separately using the <code>RFSH</code> field (and is based on core clock rate and number of rows per bank). The ranges are:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Low (MHz)</th> <th>High (MHz)</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>0</td> <td>15</td> </tr> <tr> <td>0x1</td> <td>15</td> <td>30</td> </tr> <tr> <td>0x2</td> <td>30</td> <td>50</td> </tr> <tr> <td>0x3</td> <td>50</td> <td>100</td> </tr> </tbody> </table>	Value	Low (MHz)	High (MHz)	0x0	0	15	0x1	15	30	0x2	30	50	0x3	50	100
Value	Low (MHz)	High (MHz)																	
0x0	0	15																	
0x1	15	30																	
0x2	30	50																	
0x3	50	100																	
29:27	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.															
26:16	RFSH	R/W	0x2EE	<p>Refresh Counter</p> <p>Refresh counter in core clocks. The reset value of 0x2EE provides a refresh period of 64 ms when using a 50 MHz clock.</p>															
15:10	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.															

Bit/Field	Name	Type	Reset	Description
9	SLEEP	R/W	0	Sleep Mode  Value Description 1 The SDRAM is put into low power state, but is self-refreshed. 0 No effect.
8:2	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	SIZE	R/W	0x0	Size of SDRAM  The value of this field affects address pins and behavior.  Value Description 0x0 64Mb (8MB) 0x1 128Mb (16MB) 0x2 256Mb (32MB) 0x3 512Mb (64MB)

## Register 4: EPI Host-Bus 8 Configuration (EPIHB8CFG), offset 0x010

**Important:** To access this register, the `MODE` field in the `EPICFG` register must be 0x2.

The Host Bus 8 sub-configuration register is activated when the HB8 mode is selected. The HB8 mode supports muxed address/data (overlay of lower 8 address and all 8 data pins), separated address/data, and address-less FIFO mode. Note that this register is reset when the `MODE` field in the `EPICFG` register is changed. If another mode is selected and the SDRAM mode is selected again, the values must be reinitialized.

It is intended to support SRAMs, Flash memory (read), FIFOs, CPLDs/FPGAs, and devices with an MCU/HostBus slave or 8-bit FIFO interface support.

When activated, certain pins are assigned as follows:

- `EPI31` is assigned to clock
- `EPI30` is assigned to ALE/CS<sub>n</sub> (not used when using an external FIFO)
- `EPI29` is assigned to WR<sub>n</sub> (or WR if `WRHIGH` is set)
- `EPI28` is assigned to RD<sub>n</sub>/OEn
- `EPI27` down to `EPI8` are assigned to address for all but FIFO sub-mode
- `EPI27` is assigned to FFULL (XFIFO full) when in FIFO sub-mode and `XFFEN` is set
- `EPI26` is assigned to FEMPTY (XFIFO empty) when in FIFO sub-mode and `XFEEN` is set
- `EPI7` down to `EPI0` are assigned to data (D[7:0]) in all sub-modes, and address low (A[7:0]) in muxed AD sub-mode

See Table 10-1 on page 339 for more on pin assignments.

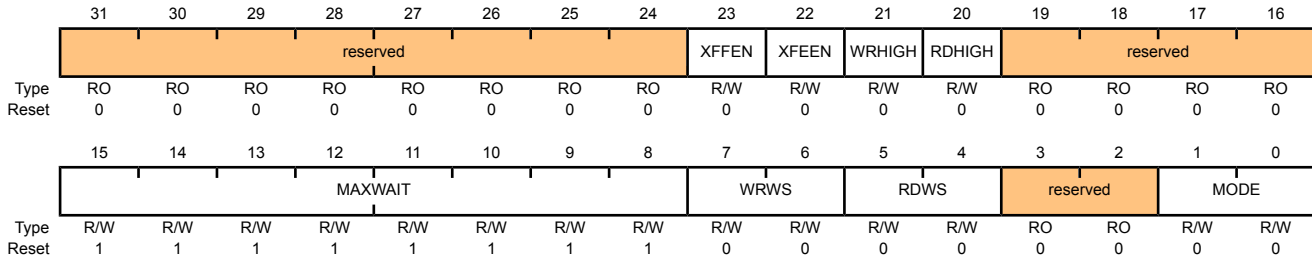
If less address pins are required, the corresponding GPIO's `AFSEL` (page 304) should not be enabled (so the EPI controller does not drive those pins, and they are available as standard GPIOs).

There is no direct chip enable (CE) model. Instead, CE can be handled in one of three ways:

1. Manually control via GPIOs.
2. Associate one or more upper address pins to CE. Because CE is normally CEn, lower addresses are not used. For example, if pins `EPI27` and `EPI26` are used for Device 1 and 0 respectively, then address 0x6800.0000 accesses Device 0 (Device 1 has its CEn high), and 0x6400.0000 accesses Device 1 (Device 0 has its CEn high). The pull-up behavior on the corresponding GPIOs must be properly configured to ensure that the pins are disabled when the interface is not in use.
3. With certain SRAMs, the ALE can be used as CEn because the address remains stable after the ALE strobe. The subsequent WR<sub>n</sub> or RD<sub>n</sub> signals write or read when ALE is low thus providing CEn functionality.

EPI Host-Bus 8 Configuration (EPIHB8CFG)

Base 0x400D.0000  
 Offset 0x010  
 Type R/W, reset 0x0000.FF00



Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23	XFFEN	R/W	0	External FIFO FULL Enable  Value Description 1 An external FIFO full signal can be used to control write cycles. If this bit is set and the external FIFO full signal is high, XFIFO writes are stalled. 0 No effect.
22	XFEEN	R/W	0	External FIFO EMPTY Enable  Value Description 1 An external FIFO empty signal can be used to control read cycles. If this bit is set and the external FIFO empty signal is high, XFIFO reads are stalled. 0 No effect.
21	WRHIGH	R/W	0	WRITE Strobe Polarity  Value Description 1 The WRITE strobe is WRn (active low). 0 The WRITE strobe is WR (active high).
20	RDHIGH	R/W	0	READ Strobe Polarity  Value Description 1 The READ strobe is RDn (active low). 0 The READ strobe is RD (active high).
19:16	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.

Bit/Field	Name	Type	Reset	Description										
15:8	MAXWAIT	R/W	0xFF	<p>Maximum Wait</p> <p>This field defines the maximum number of external clocks to wait while an external FIFO ready signal is holding off a transaction (FFULL and FEMPTY).</p> <p>When this field is clear, the transaction is held off forever.</p>										
7:6	WRWS	R/W	0x0	<p>Write Wait States</p> <p>This field adds wait states to the data phase (the address phase is not affected). The effect is to delay the rising edge of WRn (or the falling edge of WR).</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>No wait states</td> </tr> <tr> <td>0x1</td> <td>1 wait state</td> </tr> <tr> <td>0x2</td> <td>2 wait states</td> </tr> <tr> <td>0x3</td> <td>3 wait states</td> </tr> </tbody> </table> <p>This field is used in conjunction with the <b>EPIBAUD</b> register.</p>	Value	Description	0x0	No wait states	0x1	1 wait state	0x2	2 wait states	0x3	3 wait states
Value	Description													
0x0	No wait states													
0x1	1 wait state													
0x2	2 wait states													
0x3	3 wait states													
5:4	RDWS	R/W	0x0	<p>Read Wait States</p> <p>This field adds wait states to the data phase (the address phase is not affected). The effect is to delay the rising edge of RDn/Oen (or the falling edge of RD).</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>No wait states</td> </tr> <tr> <td>0x1</td> <td>1 wait state</td> </tr> <tr> <td>0x2</td> <td>2 wait states</td> </tr> <tr> <td>0x3</td> <td>3 wait states</td> </tr> </tbody> </table> <p>This field is used in conjunction with the <b>EPIBAUD</b> register.</p>	Value	Description	0x0	No wait states	0x1	1 wait state	0x2	2 wait states	0x3	3 wait states
Value	Description													
0x0	No wait states													
0x1	1 wait state													
0x2	2 wait states													
0x3	3 wait states													
3:2	reserved	RO	0x0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>										

Bit/Field	Name	Type	Reset	Description
1:0	MODE	R/W	0x0	<p>Host Bus Sub-Mode</p> <p>This field determines which of four Host Bus 8 sub-modes to use. Sub-mode use is determined by the connected external peripheral.</p> <p>Value Description</p> <p>0x0 ADMUX – AD[7:0] Data and Address are muxed on EPI[7:0] and additional address is at EPI[27:8].</p> <p>0x1 ADNONMUX – D[7:0] Data and address are separate. D[7:0] are on EPI[7:0] and A[19:0] are on EPI[27:8].</p> <p>0x2 SRAM This mode is the same as ADNONMUX, but uses address switch for multiple reads vs. OEn.</p> <p>0x3 XFIFO – D[7:0] This mode adds XFIFO controls with sense of XFIFO full and XFIFO empty. This mode uses no address or ALE.</p>

## Register 5: EPI General-Purpose Configuration (EPIGPCFG), offset 0x010

**Important:** To access this register, the `MODE` field in the `EPICFG` register must be 0x0.

The General-Purpose configuration register is used to control the size of control, data, and address pins, if used. This mode can be used for custom interfaces with FPGAs, CPLDs, and for digital data acquisition and actuator control. Note that this register is reset when the `MODE` field in the `EPICFG` register is changed. If another mode is selected and the SDRAM mode is selected again, the register the values must be reinitialized.

This mode is designed for 3 general types of use:

- Extremely high-speed clocked interfaces to FPGAs and CPLDs, with 3 sizes of data and optional address. Framing and clock-enable permit more optimized interfaces.
- General parallel GPIO. From 1 to 31 pins may be written or read, with the speed controlled by setting the baud rate in the `EPIBAUD` register (when used with the NBRFIFO and/or the WFIFO) or by rate of accesses from software or  $\mu$ DMA.
- General custom interfaces of any speed.

The configuration allows for choice of an output clock (free running or gated), a framing signal (with frame size), a clock-enable input (to stretch transactions), READ and WRITE strobes, address of varying sizes, and data of varying sizes. Additionally, provisions are made for splitting address and data phases on the external interface.

### EPI General-Purpose Configuration (EPIGPCFG)

Base 0x400D.0000

Offset 0x010

Type R/W, reset 0x0000.FF00

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CLKPIN	CLKGATE	reserved	RDYEN	FRMPIN	FRM50	FRMCNT			RW	reserved	WR2CYC	RD2CYC	reserved		
Type	R/W	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RO	R/W	R/W	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
	MAXWAIT							reserved		ASIZE		reserved		DSIZE		
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO	R/W	R/W	RO	RO	R/W	R/W
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	CLKPIN	R/W	0	Clock Pin

#### Value Description

- 1 `EPI31` functions as a clock output.
- 0 No clock output.

The clock is generated from the `EPIBAUD` register (as is the core clock which is divided down from it).

Bit/Field	Name	Type	Reset	Description
30	CLKGATE	R/W	0	<p>Clock Gated</p> <p>Value Description</p> <p>1 The clock is output only when there is data to write or read (current transaction); otherwise the clock is held low.</p> <p>0 The clock is free running.</p> <p>Note that <code>EPI27</code> is an <code>iRDY</code> signal if <code>RDYEN</code> is set. <code>CLKGATE</code> is ignored if <code>CLKPIN</code> is 0 or if the <b>EPIBAUD</b> register is cleared.</p>
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	RDYEN	R/W	0	<p>Ready Enable</p> <p>Value Description</p> <p>1 The external peripheral drives an <code>iRDY</code> signal into pin <code>EPI27</code>.</p> <p>0 The external peripheral does not drive an <code>iRDY</code> signal and is assumed to be ready always.</p> <p>The ready enable signal may only be used with a free-running clock (<code>CLKGATE=0</code>).</p> <p>The external <code>iRDY</code> signal is sampled on the rising edge of the clock. Setup and hold times must be met to ensure registration on the next rising clock edge.</p> <p>This bit is ignored if <code>CLKPIN</code> is 0 or <code>CLKGATE</code> is 1.</p>
27	FRMPIN	R/W	0	<p>Framing Pin</p> <p>Value Description</p> <p>1 A framing signal is output on <code>EPI30</code>.</p> <p>0 No framing signal.</p> <p>Framing has no impact on data itself, but forms a context for the external peripheral. When used with a free-running clock, <code>FRAME</code> forms the valid signal. When used with a gated clock, it is usually used to form a frame size.</p>
26	FRM50	R/W	0	<p>50/50 Frame</p> <p>Value Description</p> <p>1 The <code>FRAME</code> signal is output as 50/50 duty cycle using count (see <code>FRMCNT</code>).</p> <p>0 The <code>FRAME</code> signal is output as a single pulse, and then held low for the count.</p> <p>This bit is ignored if <code>FRMPIN</code> is 0.</p>



Bit/Field	Name	Type	Reset	Description
25:22	FRMCNT	R/W	0x0	<p>Frame Count</p> <p>This field specifies the size of the frame in clocks. The frame counter is used to determine the frame size. The count is FRMCNT+1. So, a FRMCNT of 0 forms a pure transaction valid signal (held high during transactions, low otherwise).</p> <p>A FRMCNT of 0 with FRM50 set inverts the FRAME signal on each transaction. A FRMCNT of 1 means FRAME is inverted every other transaction; a value of 15 means every sixteenth transaction.</p> <p>If FRM50 is set, the frame is held high for FRMCNT+1 transactions, then held low for that many transactions, and so on.</p> <p>If FRM50 is clear, the frame is pulsed high for one clock and then low for FRMCNT clocks.</p> <p>This field is ignored if FRMPIN is 0.</p>
21	RW	R/W	0	<p>Read and Write</p> <p>Value Description</p> <p>1 READ and WRITE strobes are asserted on EPI29 and EPI28. READ is asserted high on the rising edge of the clock when a read is being performed. WRITE is asserted high on the rising edge of the clock when a write is being performed</p> <p>0 READ and WRITE strobes are not output.</p> <p>This bit is forced to 1 when RD2CYC and/or WR2CYC is 1.</p>
20	reserved	RO	0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>
19	WR2CYC	R/W	0	<p>2-Cycle Writes</p> <p>Value Description</p> <p>1 Writes are two cycles long, with address on one cycle (with the WRITE strobe asserted) and data written on the following cycle (with WRITE strobe de-asserted). The next address (if any) is in the cycle following.</p> <p>0 Data is output on the same cycle as the address.</p> <p>When this bit is set, then the RW bit is forced to be set.</p>
18	RD2CYC	R/W	0	<p>2-Cycle Reads</p> <p>Value Description</p> <p>1 Reads are two cycles, with address on one cycle (with the READ strobe asserted) and data captured on the following cycle (with READ strobe de-asserted). The next address (if any) is in the cycle following.</p> <p>0 Data is captured on the cycle with READ strobe asserted.</p> <p>When this bit is set, then the RW bit is forced to be set.</p>
17:16	reserved	RO	0x0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>

Bit/Field	Name	Type	Reset	Description										
15:8	MAXWAIT	R/W	0xFF	<p>Maximum Wait</p> <p>This field defines the maximum number of external clocks to wait while an external clock-enable (see <code>RDYEN</code>) is holding off a transaction. If this field is 0, the transaction is held forever. If the maximum wait of 255 clocks (<code>MAXWAIT=0xFF</code>) is exceeded, an error interrupt occurs and the transaction is aborted/ignored.</p>										
7:6	reserved	RO	0x0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>										
5:4	ASIZE	R/W	0x0	<p>Address Bus Size</p> <p>This field defines the size of the address bus (starting at <code>EPI8</code>, <code>EPI16</code>, or <code>EPI24</code>, depending on size). Subsets of these numbers can be created by disabling the <code>AFSEL</code> for the corresponding GPIOs. Also, if <code>RDYEN</code> is 1, then the address sizes are 1 smaller (3, 11, 19).</p> <p>The values are:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>No address</td> </tr> <tr> <td>0x1</td> <td>4 Bits Wide (<code>EPI24</code> to <code>EPI27</code>)</td> </tr> <tr> <td>0x2</td> <td>12 Bits Wide (<code>EPI16</code> to <code>EPI27</code>). This size cannot be used with 24-bit data.</td> </tr> <tr> <td>0x3</td> <td>20 Bits Wide (<code>EPI8</code> to <code>EPI27</code>). This size cannot be used with data sizes other than 8.</td> </tr> </tbody> </table>	Value	Description	0x0	No address	0x1	4 Bits Wide ( <code>EPI24</code> to <code>EPI27</code> )	0x2	12 Bits Wide ( <code>EPI16</code> to <code>EPI27</code> ). This size cannot be used with 24-bit data.	0x3	20 Bits Wide ( <code>EPI8</code> to <code>EPI27</code> ). This size cannot be used with data sizes other than 8.
Value	Description													
0x0	No address													
0x1	4 Bits Wide ( <code>EPI24</code> to <code>EPI27</code> )													
0x2	12 Bits Wide ( <code>EPI16</code> to <code>EPI27</code> ). This size cannot be used with 24-bit data.													
0x3	20 Bits Wide ( <code>EPI8</code> to <code>EPI27</code> ). This size cannot be used with data sizes other than 8.													
3:2	reserved	RO	0x0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>										
1:0	DSIZE	R/W	0x0	<p>Size of Data Bus</p> <p>This field defines the size of the data bus (starting at <code>EPI0</code>). Subsets of these numbers can be created by disabling the <code>AFSEL</code> for the corresponding GPIOs. Note that size 32 may not be used with clock, frame, address, or other control.</p> <p>The values are:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>8 Bits Wide (<code>EPI0</code> to <code>EPI7</code>)</td> </tr> <tr> <td>0x1</td> <td>16 Bits Wide (<code>EPI0</code> to <code>EPI15</code>)</td> </tr> <tr> <td>0x2</td> <td>24 Bits Wide (<code>EPI0</code> to <code>EPI23</code>)</td> </tr> <tr> <td>0x3</td> <td>32 Bits Wide (<code>EPI0</code> to <code>EPI31</code>). This size may not be used with a clock. This value is normally used for acquisition input and actuator control as well as other general-purpose uses that require 32 bits per direction.</td> </tr> </tbody> </table>	Value	Description	0x0	8 Bits Wide ( <code>EPI0</code> to <code>EPI7</code> )	0x1	16 Bits Wide ( <code>EPI0</code> to <code>EPI15</code> )	0x2	24 Bits Wide ( <code>EPI0</code> to <code>EPI23</code> )	0x3	32 Bits Wide ( <code>EPI0</code> to <code>EPI31</code> ). This size may not be used with a clock. This value is normally used for acquisition input and actuator control as well as other general-purpose uses that require 32 bits per direction.
Value	Description													
0x0	8 Bits Wide ( <code>EPI0</code> to <code>EPI7</code> )													
0x1	16 Bits Wide ( <code>EPI0</code> to <code>EPI15</code> )													
0x2	24 Bits Wide ( <code>EPI0</code> to <code>EPI23</code> )													
0x3	32 Bits Wide ( <code>EPI0</code> to <code>EPI31</code> ). This size may not be used with a clock. This value is normally used for acquisition input and actuator control as well as other general-purpose uses that require 32 bits per direction.													

## Register 6: EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2), offset 0x014

Tempest RevB

**Important:** To access this register, the `MODE` field in the `EPICFG` register must be 0x1.

This register is used to configure operation while in Host-Bus 8 mode. Note that this register is reset when the `MODE` field in the `EPICFG` register is changed. If another mode is selected and the Host-Bus 8 mode is selected again, the values must be reinitialized.

### EPI Host-Bus 8 Configuration 2 (EPIHB8CFG2)

Base 0x400D.0000  
 Offset 0x014  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	WORD	reserved						reserved									
Type	R/W	RO	RO	RO	RO	RO	R/W	R/W	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	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	WORD	R/W	0x0	Word Access Mode  By default, the EPI controller uses data bits [7:0] for Host-Bus 8 accesses. When using Word Access mode, the EPI controller can automatically route bytes of data onto the correct byte lanes such that data can be stored in bits [31:8].  Value Description 0 Word Access mode is disabled. 1 Word Access mode is enabled.
30:26	reserved	RO	0x000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description								
25:24	CSCFG	R/W	0x0	<p>Chip Select Configuration</p> <p>This field controls the chip select options, including an ALE format and a chip select format.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td> <p>ALE Configuration</p> <p><i>EPI30</i> is used as an address latch (ALE). When using this mode, the address and data should be muxed (<i>HB8MODE</i> field in the <b>CPIHB8CFG</b> register should be configured to 0x0). If needed, the address can be latched by external logic.</p> </td> </tr> <tr> <td>0x1</td> <td> <p>CSn Configuration</p> <p><i>EPI30</i> is used as a Chip Select (CSn). When using this mode, the address and data should not be muxed (<i>HB8MODE</i> field in the <b>CPIHB8CFG</b> register should be configured to 0x1). In this mode, the WR signal (<i>EPI29</i>) and the RD signal (<i>EPI28</i>) are used to latch the address when CSn is low.</p> </td> </tr> <tr> <td>0x2-0x3</td> <td>reserved</td> </tr> </tbody> </table>	Value	Description	0x0	<p>ALE Configuration</p> <p><i>EPI30</i> is used as an address latch (ALE). When using this mode, the address and data should be muxed (<i>HB8MODE</i> field in the <b>CPIHB8CFG</b> register should be configured to 0x0). If needed, the address can be latched by external logic.</p>	0x1	<p>CSn Configuration</p> <p><i>EPI30</i> is used as a Chip Select (CSn). When using this mode, the address and data should not be muxed (<i>HB8MODE</i> field in the <b>CPIHB8CFG</b> register should be configured to 0x1). In this mode, the WR signal (<i>EPI29</i>) and the RD signal (<i>EPI28</i>) are used to latch the address when CSn is low.</p>	0x2-0x3	reserved
Value	Description											
0x0	<p>ALE Configuration</p> <p><i>EPI30</i> is used as an address latch (ALE). When using this mode, the address and data should be muxed (<i>HB8MODE</i> field in the <b>CPIHB8CFG</b> register should be configured to 0x0). If needed, the address can be latched by external logic.</p>											
0x1	<p>CSn Configuration</p> <p><i>EPI30</i> is used as a Chip Select (CSn). When using this mode, the address and data should not be muxed (<i>HB8MODE</i> field in the <b>CPIHB8CFG</b> register should be configured to 0x1). In this mode, the WR signal (<i>EPI29</i>) and the RD signal (<i>EPI28</i>) are used to latch the address when CSn is low.</p>											
0x2-0x3	reserved											
25:0	reserved	RO	0x000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								

**Register 7: EPI General-Purpose Configuration 2 (EPIGPCFG2), offset 0x014**

Tempest RevB

**Important:** To access this register, the `MODE` field in the `EPICFG` register must be 0x1.

This register is used to configure operation while in General-Purpose sub-mode. Note that this register is reset when the `MODE` field in the `EPICFG` register is changed. If another mode is selected and the General-Purpose mode is selected again, the values must be reinitialized.

## EPI General-Purpose Configuration 2 (EPIGPCFG2)

Base 0x400D.0000

Offset 0x014

Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WORD	reserved														
Type	R/W	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	WORD	R/W	0x0	<p>Word Access Mode</p> <p>By default, the EPI controller uses data bits [7:0] for Host-Bus 8 accesses. When using Word Access mode, the EPI controller can automatically route bytes of data onto the correct byte lanes such that data can be stored in bits [31:8].</p> <p>Value Description</p> <table border="0"> <tr> <td>0</td> <td>Word Access mode is disabled.</td> </tr> <tr> <td>1</td> <td>Word Access mode is enabled.</td> </tr> </table>	0	Word Access mode is disabled.	1	Word Access mode is enabled.
0	Word Access mode is disabled.							
1	Word Access mode is enabled.							
30:0	reserved	RO	0x000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.				

### Register 8: EPI Address Map (EPIADDRMAP), offset 0x01C

This register enables address mapping. The EPI controller can directly address memory and peripherals. In addition, the EPI controller supports address mapping to allow indirect accesses in the External RAM and External Peripheral areas. Note that use of either one does not affect how the EPI Controller behaves, but care must be taken not to overlap memory regions.

#### EPI Address Map (EPIADDRMAP)

Base 0x400D.0000  
 Offset 0x01C  
 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								EPSZ		EPADR		ERSZ		ERADR	
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:6	EPSZ	R/W	0x0	External Peripheral Size  This field selects the size of the external peripheral. If the size of the external peripheral is larger, a bus fault occurs. If the size of the external peripheral is smaller, it wraps (upper address bits unused):  Value Description 0x0 0x100 (256) 0x1 0x1.0000 (64 KB) 0x2 0x100.0000 (16 MB) 0x3 0x2000.0000 (512 MB)
5:4	EPADR	R/W	0x0	External Peripheral Address  This field selects address mapping for the external peripheral area:  Value Description 0x0 Not mapped 0x1 At 0xA000.0000 0x2 At 0xC000.0000 0x3 reserved

Bit/Field	Name	Type	Reset	Description										
3:2	ERSZ	R/W	0x0	<p>External RAM Size</p> <p>This field selects the size of mapped RAM. If the size of the external memory is larger, a bus fault occurs. If the size of the external memory is smaller, it wraps (upper address bits unused):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>0x100 (256)</td> </tr> <tr> <td>0x1</td> <td>0x1.0000 (64KB)</td> </tr> <tr> <td>0x2</td> <td>0x100.0000 (16MB)</td> </tr> <tr> <td>0x3</td> <td>0x2000.0000 (512MB)</td> </tr> </tbody> </table>	Value	Description	0x0	0x100 (256)	0x1	0x1.0000 (64KB)	0x2	0x100.0000 (16MB)	0x3	0x2000.0000 (512MB)
Value	Description													
0x0	0x100 (256)													
0x1	0x1.0000 (64KB)													
0x2	0x100.0000 (16MB)													
0x3	0x2000.0000 (512MB)													
1:0	ERADR	R/W	0x0	<p>External RAM Address</p> <p>Selects address mapping for external RAM area:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Not mapped</td> </tr> <tr> <td>0x1</td> <td>At 0x6000.0000</td> </tr> <tr> <td>0x2</td> <td>At 0x8000.0000</td> </tr> <tr> <td>0x3</td> <td>reserved</td> </tr> </tbody> </table>	Value	Description	0x0	Not mapped	0x1	At 0x6000.0000	0x2	At 0x8000.0000	0x3	reserved
Value	Description													
0x0	Not mapped													
0x1	At 0x6000.0000													
0x2	At 0x8000.0000													
0x3	reserved													

**Register 9: EPI Read Size 0 (EPIRSIZE0), offset 0x020**

**Register 10: EPI Read Size 1 (EPIRSIZE1), offset 0x030**

This register selects the size of transactions when performing non-blocking reads with the **EPIRPSTD** registers. This size affects how the external address is incremented.

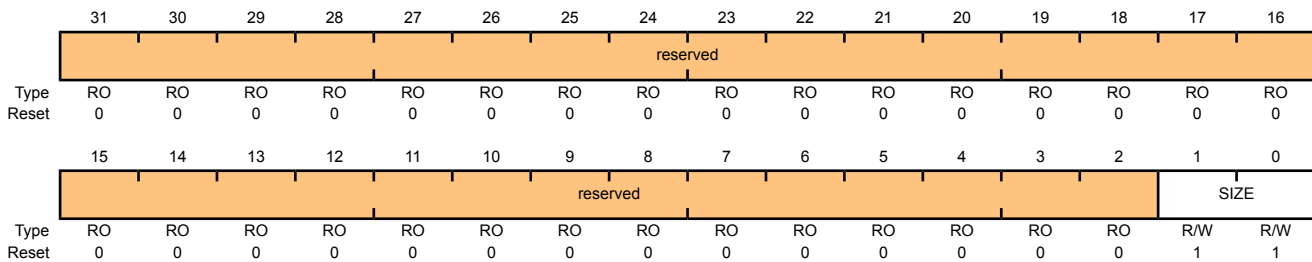
The **SIZE** field must match the external data width as configured in the **EPIHB8CFG** or **EPIGPCFG** register (see Table 10-1 on page 339).

SDRAM mode uses a 16-bit data interface. If **SIZE** is 0x1, data is returned on the least significant bits (D[7:0]), and the remaining bits D[31:8] are all zeros, therefore the data on bits D[15:8] is lost.. If **SIZE** is 0x2, data is returned on the least significant bits (D[15:0]), and the remaining bits D[31:16] are all zeros.

Note that changing this register while a read is active has an unpredictable effect.

**EPI Read Size 0 (EPIRSIZE0)**

Base 0x400D.0000  
 Offset 0x020  
 Type R/W, reset 0x0000.0003



Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1:0	SIZE	R/W	0x3	Current Size
				Value Description
				0x0 reserved
				0x1 Byte (8 bits)
				0x2 Half-word (16 bits)
				0x3 Word (32 bits)



**Register 11: EPI Read Address 0 (EPIRADDR0), offset 0x024****Register 12: EPI Read Address 1 (EPIRADDR1), offset 0x034**

This register holds the current address value. When performing non-blocking reads via the **EPIRSTD** registers, this register's value forms the address (when used by the mode). That is, when a **EPIRSTD** register is written with a non-0 value, this register is used as the first address. After each read, it is incremented by the size specified by the corresponding **EPIRSIZE** register. Thus at the end of a read, this register contains the next address for the next read. For example, if the last read was 0x20, and the size is word, then the register contains 0x24. When a non-blocking read is cancelled, this register contains the next address that would have been read had it not been cancelled. For example, if reading by bytes and 0x103 had been read but not 0x104, this register contains 0x104. In this manner, the system can determine the number of values in the NBRFIFO to drain.

Note that changing this register while a read is active has an unpredictable effect due to race condition.

**EPI Read Address 0 (EPIRADDR0)**

Base 0x400D.0000

Offset 0x024

Type R/W, reset 0x0000.0000

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

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.
28:0	ADDR	R/W	0x000.0000	Current Address Next address to read.

**Register 13: EPI Non-Blocking Read Data 0 (EPIRPSTD0), offset 0x028**

**Register 14: EPI Non-Blocking Read Data 1 (EPIRPSTD1), offset 0x038**

This register sets up a non-blocking read via the external interface. A non-blocking read is started by writing to this register with the count (other than 0). Clearing this register terminates an active non-blocking read as well as cancelling any that are pending. This register should always be cleared before writing a value other than 0; failure to do so can cause improper operation.

The first address is based on the corresponding **EPIRADDR** register. The address register is incremented by the size specified by the **EPIRSIZE** register after each read. If the size is less than a word, only the least significant bits of data are filled into the NBRFIFO; the most significant bits are cleared.

Note that all three registers may be written using one STM instruction, such as with a structure copy in C/C++.

The data may be read from the **EPIREADFIFO** register after the read cycle is completed. The interrupt mechanism is normally used to trigger the FIFO reads via ISR or  $\mu$ DMA.

If the countdown has not reached 0 and the NBRFIFO is full, the external interface waits until a NBRFIFO entry becomes available to continue.

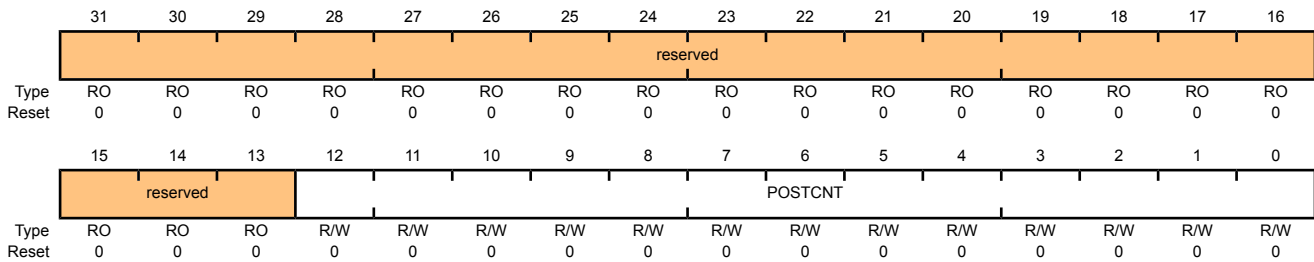
Note: if a blocking read or write is performed through the address mapped area (at 0x6000.0000 through 0xCFFF.FFFF), any current non-blocking read is paused (at the next safe boundary), and the blocking request is inserted. After completion of any blocking reads or writes, the non-blocking reads continue from where they were paused.

The other way to read data is via the address mapped locations (see the **EPIADDRMAP** register), but this method is blocking (core or  $\mu$ DMA waits until result is returned).

To cancel a non-blocking read, clear this register. To make sure that all values read are drained from the NBRFIFO, the **EPISTAT** register must be consulted to be certain that bits **NBRBUSY** and **ACTIVE** are cleared. One of these registers should not be cleared until either the other **EPIRPSTDx** register becomes active or the external interface is not busy. At that point, the corresponding **EPIRADDR** register indicates how many values were read.

EPI Non-Blocking Read Data 0 (EPIRPSTD0)

Base 0x400D.0000  
 Offset 0x028  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:13	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
12:0	POSTCNT	R/W	0x000	<p>Post Count</p> <p>A write of a non-zero value starts a read operation for that count. Note that it is the software's responsibility to handle address wraparound.</p> <p>Reading this register provides the current count.</p> <p>A write of 0 cancels a non-blocking read (whether active now or pending).</p> <p>Prior to writing a non-zero value, this register must first be cleared.</p>

### Register 15: EPI Status (EPISTAT), offset 0x060

This register indicates which non-blocking read register is currently active; it also indicates whether the external interface is busy performing a write or non-blocking read (it cannot be performing a blocking read, as the bus would be blocked and as a result, this register could not be accessed).

This register is useful for determining which non-blocking read register is active when both are loaded with values and when implementing sequencing or sharing.

This register is also useful when canceling non-blocking reads, as it shows how many values were read by the canceled side.

#### EPI Status (EPISTAT)

Base 0x400D.0000  
 Offset 0x060  
 Type R, 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						CELOW	XFFULL	XFEMPTY	INITSEQ	WBUSY	NBRBUSY	reserved			ACTIVE
Type	RO	RO	RO	RO	RO	RO	R	R	R	R	R	R	RO	RO	RO	R
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	CELOW	R	0	<p>Clock Enable Low</p> <p>This bit provides information on the clock status when in general-purpose mode and the RDYEN bit is set.</p> <p>Value Description</p> <ul style="list-style-type: none"> <li>1 The external device is gating the clock (iRDY is low). Attempts to read or write in this situation are stalled until the clock is enabled or the counter times-out on MAXWAIT.</li> <li>0 The external device is not gating the clock.</li> </ul>
8	XFFULL	R	0	<p>External FIFO Full</p> <p>This bit provides information on the XFIFO when in the FIFO sub-mode of the Host Bus 8 mode with the XFFEN bit set in the EPIHB8CFG register. The EPI26 signal reflects the status of this bit.</p> <p>Value Description</p> <ul style="list-style-type: none"> <li>1 The XFIFO is signaling as full (the FIFO full signal is high). Attempts to write in this case are stalled until the XFIFO full signal goes low or the counter times-out on MAXWAIT.</li> <li>0 The external device is not gating the clock.</li> </ul>

Bit/Field	Name	Type	Reset	Description
7	XFEMPTY	R	0	<p>External FIFO Empty</p> <p>This bit provides information on the XFIFO when in the FIFO sub-mode of the Host Bus 8 mode with the <i>XFEEEN</i> bit set in the <b>EPIHB8CFG</b> register. The <i>EPI27</i> signal reflects the status of this bit.</p> <p>Value Description</p> <p>1 The XFIFO is signaling as empty (the FIFO empty signal is high).</p> <p>Attempts to read in this case are stalled until the XFIFO empty signal goes low or the counter times-out on <i>MAXWAIT</i>.</p> <p>0 The external device is not gating the clock.</p>
6	INITSEQ	R	0	<p>Initialization Sequence</p> <p>Value Description</p> <p>1 The SDRAM interface is running through the wakeup period (greater than 100 <math>\mu</math>s).</p> <p>If an attempt is made to read or write the SDRAM during this period, the access is held off until the wakeup period is complete.</p> <p>0 The SDRAM interface is not in the wakeup period.</p>
5	WBUSY	R	0	<p>Write Busy</p> <p>Value Description</p> <p>1 The external interface is performing a write.</p> <p>0 The external interface is not performing a write.</p>
4	NBRBUSY	R	0	<p>Non-Blocking Read Busy</p> <p>Value Description</p> <p>1 The external interface is performing a non-blocking read, or if the non-blocking read is paused due to a write.</p> <p>0 The external interface is not performing a non-blocking read.</p>
3:1	reserved	RO	0x0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>
0	ACTIVE	R	0	<p>Register Active</p> <p>Value Description</p> <p>1 The <b>EPIRPSTD1</b> register is active.</p> <p>0 If <i>NBRBUSY</i> is set, the <b>EPIRPSTD0</b> register is active.</p> <p>If the <i>NBRBUSY</i> bit is clear, then neither <b>EPIRPSTDx</b> register is active.</p>

### Register 16: EPI Read FIFO Count (EPIRFIFOCNT), offset 0x06C

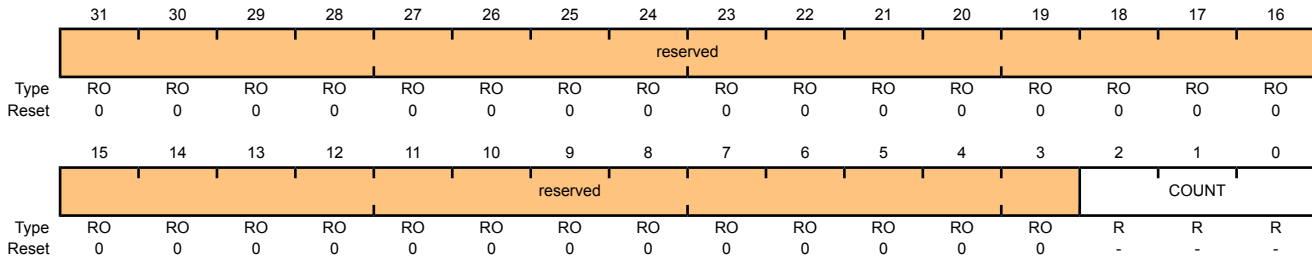
This register returns the number of values in the NBRFIFO (the data in the NBRFIFO can be read via the **EPIREADFIFO** register). A race is possible, but that only means that more values may come in after this register has been read.

#### EPI Read FIFO Count (EPIRFIFOCNT)

Base 0x400D.0000

Offset 0x06C

Type R, reset -



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	COUNT	R	-	FIFO Count Number of filled entries in the NBRFIFO.

**Register 17: EPI Read FIFO (EPIREADFIFO), offset 0x070**

**Register 18: EPI Read FIFO Alias 1 (EPIREADFIFO1), offset 0x074**

**Register 19: EPI Read FIFO Alias 2 (EPIREADFIFO2), offset 0x078**

**Register 20: EPI Read FIFO Alias 3 (EPIREADFIFO3), offset 0x07C**

**Register 21: EPI Read FIFO Alias 4 (EPIREADFIFO4), offset 0x080**

**Register 22: EPI Read FIFO Alias 5 (EPIREADFIFO5), offset 0x084**

**Register 23: EPI Read FIFO Alias 6 (EPIREADFIFO6), offset 0x088**

**Register 24: EPI Read FIFO Alias 7 (EPIREADFIFO7), offset 0x08C**

This register returns the contents of the NBRFIFO or 0 if the NBRFIFO is empty. Each read returns the data that is at the top of the NBRFIFO, and then empties that value from the NBRFIFO. The alias registers can be used with the LDMIA instruction for more efficient operation (for up to 8 registers). See *ARM® Cortex™-M3 Technical Reference Manual* for more information on the LDMIA instruction.

#### EPI Read FIFO (EPIREADFIFO)

Base 0x400D.0000

Offset 0x070

Type R, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:0	DATA	R	0x0000.0000	Reads Data

This field contains the data that is at the top of the NBRFIFO. After being read, the NBRFIFO entry is removed.

### Register 25: EPI FIFO Level Selects (EPIFIFOLVL), offset 0x200

This register allows selection of the FIFO levels which trigger an interrupt to the core or, more efficiently, a DMA request to the  $\mu$ DMA. The NBRFIFO select triggers on fullness such that it triggers on match or above (more full). The WFIFO triggers on emptiness such that it triggers on match or below (less entries).

It should be noted that the FIFO triggers are not identical to other such FIFOs in Stellaris<sup>®</sup> peripherals. In particular, empty and full triggers are provided to avoid wait states when using blocking operations.

The settings in this register are only meaningful if the  $\mu$ DMA is active or the interrupt is enabled.

Additionally, this register allows protection against writes stalling and notification of performing blocking reads which stall for extra time due to preceding writes. The two functions behave in a non-orthogonal way because read and write are not orthogonal.

The write error bit configures the system such that an attempted write to an already full WFIFO abandons the write and signals an error interrupt to prevent accidental latencies due to stalling writes.

The read error bit configures the system such that after a read has been stalled due to any preceding writes in the WFIFO, the error interrupt is generated. Note that the excess stall is not prevented, but an interrupt is generated after the fact to notify that it has happened.

#### EPI FIFO Level Selects (EPIFIFOLVL)

Base 0x400D.0000  
Offset 0x200  
Type R/W, reset 0x0000.0033

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

Bit/Field	Name	Type	Reset	Description
31:18	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.
17	WFERR	R/W	0	Write Full Error
				Value Description
				1 This bit enables the Write Full error interrupt ( <b>WTFULL</b> in the <b>EPIIC</b> register) to be generated when a write is attempted and the WFIFO is full. The write stalls until a WFIFO entry becomes available.
				0 The Write Full error interrupt is disabled. Writes are stalled when the WFIFO is full until a space becomes available but an error is not generated. Note that the Cortex-M3 write buffer may hide that stall if no other memory transactions are attempted during that time.



Bit/Field	Name	Type	Reset	Description
16	RSERR	R/W	0	<p>Read Stall Error</p> <p>Value Description</p> <p>1 This bit enables the Read Stalled error interrupt (<math>R_{STALL}</math> in the <b>EPIIC</b> register) to be generated when a read is attempted and the WFIFO is not empty. The read is still stalled during the time the WFIFO drains, but this error notifies the application that this excess delay has occurred.</p> <p>0 The Read Stalled error interrupt is disabled. Reads behave as normal and are stalled until any preceding writes have completed and the read has returned a result.</p> <p>Note that the configuration of this bit has no effect on non-blocking reads.</p>
15:7	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.
6:4	WRFIFO	R/W	0x3	<p>Write FIFO</p> <p>This field configures the trigger point for the WFIFO.</p> <p>Value Description</p> <p>0x0 Trigger when there are 1 to 4 spaces available in the WFIFO.</p> <p>0x1 reserved</p> <p>0x2 Trigger when there are 1 to 3 spaces available in the WFIFO.</p> <p>0x3 Trigger when there are 1 to 2 spaces available in the WFIFO.</p> <p>0x4 Trigger when there is 1 space available in the WFIFO.</p> <p>0x5-0x7 reserved</p>
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	RDFIFO	R/W	0x3	<p>Read FIFO</p> <p>This field configures the trigger point for the NBRFIFO.</p> <p>Value Description</p> <p>0x0 reserved</p> <p>0x1 Trigger when there are 1 or more entries in the NBRFIFO.</p> <p>0x2 Trigger when there are 2 or more entries in the NBRFIFO.</p> <p>0x3 Trigger when there are 4 or more entries in the NBRFIFO.</p> <p>0x4 Trigger when there are 6 or more entries in the NBRFIFO.</p> <p>0x5 Trigger when there are 7 or more entries in the NBRFIFO.</p> <p>0x6 Trigger when there are 8 entries in the NBRFIFO.</p> <p>0x7 reserved</p>

### Register 26: EPI Write FIFO Count (EPIWFIFOCNT), offset 0x204

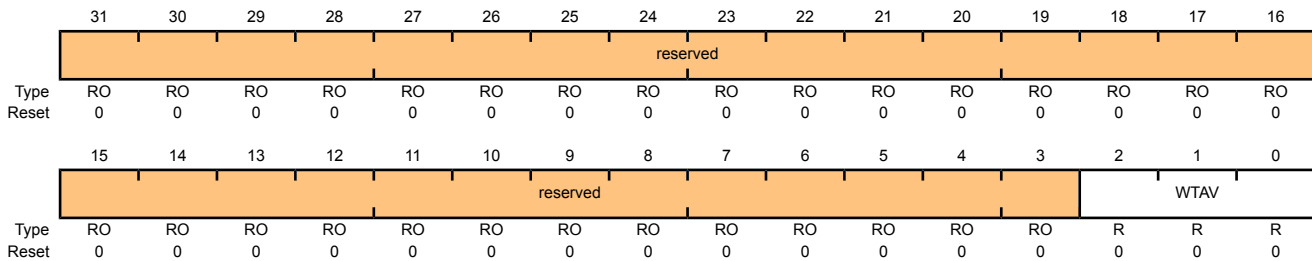
This register contains the number of slots currently available in the WFIFO. This register may be used for polled writes to avoid stalling and for blocking reads to avoid excess stalling (due to undrained writes). An example use for writes may be:

```
for (idx = 0; idx < cnt; idx++) {
while (EPIWFIFOCNT == 0) ;
*ext_ram = *mydata++;
}
```

The above code ensures that writes to the address mapped location do not occur unless the WFIFO has room. Although polling makes the code wait (spinning in the loop), it does not prevent interrupts being serviced due to bus stalling.

#### EPI Write FIFO Count (EPIWFIFOCNT)

Base 0x400D.0000  
 Offset 0x204  
 Type R, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	WTAV	R	0x0	Available Write Transactions  The number of write transactions available in the WFIFO.  When clear, a write is stalled waiting for a slot to become free (from a preceding write completing).

## Register 27: EPI Interrupt Mask (EPIIM), offset 0x210

This register is the interrupt mask set or clear register. For each interrupt source (read, write, and error), a mask value of 1 allows the interrupt source to trigger an interrupt to the interrupt controller; a mask value of 0 prevents the interrupt source from triggering an interrupt.

Note that interrupt masking has no effect on  $\mu$ DMA, which operates off the raw source of the read and write interrupts.

### EPI Interrupt Mask (EPIIM)

Base 0x400D.0000

Offset 0x210

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													WRIM	RDIM	ERRIM	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	WRIM	R/W	0	Write Interrupt Mask
				Value Description
				1 <b>WRRIS</b> in the <b>EPIRIS</b> register is not masked and can trigger an interrupt to the interrupt controller.
				0 <b>WRRIS</b> in the <b>EPIRIS</b> register is masked and does not cause an interrupt.
1	RDIM	R/W	0	Read Interrupt Mask
				Value Description
				1 <b>RDRIS</b> in the <b>EPIRIS</b> register is not masked and can trigger an interrupt to the interrupt controller.
				0 <b>RDRIS</b> in the <b>EPIRIS</b> register is masked and does not cause an interrupt.
0	ERRIM	R/W	0	Error Interrupt Mask
				Value Description
				1 <b>ERRIS</b> in the <b>EPIRIS</b> register is not masked and can trigger an interrupt to the interrupt controller.
				0 <b>ERRIS</b> in the <b>EPIRIS</b> register is masked and does not cause an interrupt.

### Register 28: EPI Raw Interrupt Status (EPIRIS), offset 0x214

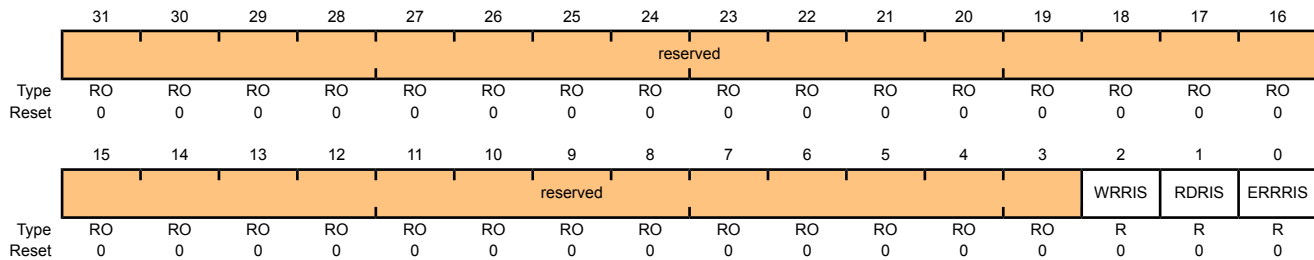
This register is the raw interrupt status register. On a read, it gives the current state of each interrupt source. A write has no effect.

Note that raw status for read and write is set or cleared based on FIFO fullness as controlled by **EPIFIFOLVL**.

Raw status for error is held until the error is cleared by writing to the **EPIIC** register.

#### EPI Raw Interrupt Status (EPIRIS)

Base 0x400D.0000  
 Offset 0x214  
 Type R, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	WRRIS	R	0	Write Raw Interrupt Status  Value Description 1 The number of available entries in the WFIFO is within the range specified by the trigger level (the <code>WRFIFO</code> field in the <b>EPIFIFOLVL</b> register). 0 The number of available entries in the WFIFO is above the range specified by the trigger level.  This bit is cleared when the level in the WFIFO is above the trigger point programmed by the <code>WRFIFO</code> field.
1	RDRIS	R	0	Read Raw Interrupt Status  Value Description 1 The number of valid entries in the NBRFIFO is within the range specified by the trigger level (the <code>RDFIFO</code> field in the <b>EPIFIFOLVL</b> register). 0 The number of valid entries in the NBRFIFO is below the range specified by the trigger level.  This bit is cleared when the level in the NBRFIFO is below the trigger point programmed by the <code>RDFIFO</code> field.

Bit/Field	Name	Type	Reset	Description				
0	ERRRIS	R	0	<p>Error Raw Interrupt Status</p> <p>The error interrupt occurs in the following situations:</p> <ul style="list-style-type: none"> <li>■ <b>WFIFO Full.</b> For a full WFIFO to generate an error interrupt, the <code>WFERR</code> bit in the <b>EPIFIFOLVL</b> register must be set.</li> <li>■ <b>Read Stalled.</b> For a stalled read to generate an error interrupt, the <code>RSERR</code> bit in the <b>EPIFIFOLVL</b> register must be set.</li> <li>■ <b>Timeout.</b> If the <code>MAXWAIT</code> field in the <b>EPIGPCFG</b> register is configured to a value other than 0, a timeout error occurs when <code>iRDY</code> or <code>XFIFO</code> not-ready signals hold a transaction for more than the count in <code>MAXWAIT</code>.</li> </ul> <p>Value Description</p> <table border="0"> <tr> <td>1</td> <td>A WFIFO Full, a Read Stalled, or a Timeout error has occurred.</td> </tr> <tr> <td>0</td> <td>An error has not occurred.</td> </tr> </table> <p>To determine which error occurred, read the status of the <b>EPI Error Interrupt Status and Clear (EPIEISC)</b> register. This bit is cleared by writing a 1 to the bit in the <b>EPIEISC</b> register that caused the interrupt.</p>	1	A WFIFO Full, a Read Stalled, or a Timeout error has occurred.	0	An error has not occurred.
1	A WFIFO Full, a Read Stalled, or a Timeout error has occurred.							
0	An error has not occurred.							

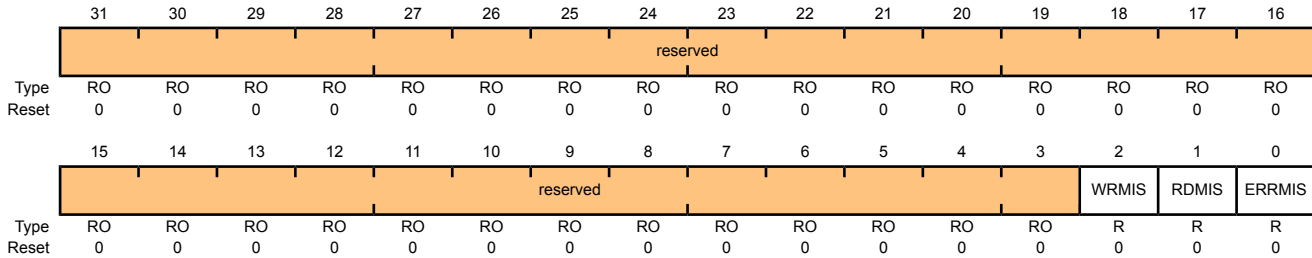
### Register 29: EPI Masked Interrupt Status (EPIMIS), offset 0x218

This register is the masked interrupt status register. On read, it gives the current state of each interrupt source (read, write, and error) after being masked via the **EPIIM** register. A write has no effect.

The values returned are the ANDing of the **EPIIM** and **EPIRIS** registers. If a bit is set in this register, the interrupt is sent to the interrupt controller.

#### EPI Masked Interrupt Status (EPIMIS)

Base 0x400D.0000  
Offset 0x218  
Type R, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	WRMIS	R	0	Write Masked Interrupt Status  Value Description 1 The number of available entries in the WFIFO is within the range specified by the trigger level (the <b>WRFIFO</b> field in the <b>EPIFIFOLVL</b> register) and the <b>WRIM</b> bit in the <b>EPIIM</b> register is set, triggering an interrupt to the interrupt controller. 0 The number of available entries in the WFIFO is above the range specified by the trigger level or the interrupt is masked.
1	RDMIS	R	0	Read Masked Interrupt Status  Value Description 1 The number of valid entries in the NBRFIFO is within the range specified by the trigger level (the <b>RDFIFO</b> field in the <b>EPIFIFOLVL</b> register) and the <b>RDIM</b> bit in the <b>EPIIM</b> register is set, triggering an interrupt to the interrupt controller. 0 The number of valid entries in the NBRFIFO is below the range specified by the trigger level or the interrupt is masked.
0	ERRMIS	R	0	Error Masked Interrupt Status  Value Description 1 A WFIFO Full, a Read Stalled, or a Timeout error has occurred and the <b>ERIM</b> bit in the <b>EPIIM</b> register is set, triggering an interrupt to the interrupt controller. 0 An error has not occurred or the interrupt is masked.

### Register 30: EPI Error Interrupt Status and Clear (EPIEISC), offset 0x21C

This register is used to clear a pending error interrupt. If any of these bits are set, the `ERRRIS` bit in the `EPIRIS` register is set, and an EPI controller error is sent to the interrupt controller if the `ERIM` bit in the `EPIIM` register is set. Clearing any defined bit has no effect; setting a bit clears the error source and the raw error returns to 0. Note that writing to this register and reading back immediately (pipelined by the processor) returns the old register contents. One cycle is needed between write and read.

#### EPI Error Interrupt Status and Clear (EPIEISC)

Base 0x400D.0000  
 Offset 0x21C  
 Type R/W1C, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	WTFULL	R/W1C	0	Write FIFO Full Error  Value Description 1 The <code>WFERR</code> bit is enabled and a write is stalled due to the <code>WFIFO</code> being full. 0 The <code>WFERR</code> bit is not enabled or no writes are stalled.  Writing a 1 to this bit clears it and the <code>WFERR</code> bit in the <code>EPIFIFOLVL</code> register.
1	RSTALL	R/W1C	0	Read Stalled Error  Value Description 1 The <code>RSERR</code> bit is enabled and a pending read is stalled due to writes in the <code>WFIFO</code> . 0 The <code>RSERR</code> bit is not enabled pr no pending reads are stalled.  Writing a 1 to this bit clears it and the <code>RSERR</code> bit in the <code>EPIFIFOLVL</code> register.

Bit/Field	Name	Type	Reset	Description				
0	TOUT	R/W1C	0	<p>Timeout Error</p> <p>This bit is the timeout error source. The timeout error occurs when the iRDY or XFIFO not-ready signals hold a transaction for more than the count in MAXWAIT (when not 0).</p> <p>Value Description</p> <table><tbody><tr><td>1</td><td>A timeout error has occurred.</td></tr><tr><td>0</td><td>No timeout error has occurred.</td></tr></tbody></table> <p>Writing a 1 to bit this clears it.</p>	1	A timeout error has occurred.	0	No timeout error has occurred.
1	A timeout error has occurred.							
0	No timeout error has occurred.							



## 11 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 (Timer 0, Timer 1, Timer 2, and Timer 3). Each GPTM block provides two 16-bit timers/counters (referred to as Timer A and Timer B) 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  $\mu$ DMA transfers.

In addition, timers can be used to trigger analog-to-digital conversions (ADC). The ADC trigger signals from all of the general-purpose timers are ORed together before reaching the ADC module, so only one timer should be used to trigger ADC events.

The GPT Module is one timing resource available on the Stellaris® microcontrollers. Other timer resources include the System Timer (SysTick) (see “System Timer (SysTick)” on page 65) and the PWM timer in the PWM module (see “PWM Timer” on page 830).

The General-Purpose Timers provide the following features:

- Count up or down
- 16- or 32-bit programmable one-shot timer
- 16- or 32-bit programmable periodic timer
- 16-bit general-purpose timer with an 8-bit prescaler
- 32-bit Real-Time Clock (RTC) when using an external 32.768-KHz clock as the input
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events
- ADC event trigger
- User-enabled stalling when the controller asserts CPU Halt flag during debug (excluding RTC mode)
- 16-bit input-edge count- or time-capture modes
- 16-bit PWM mode with software-programmable output inversion of the PWM signal
- Ability to determine the elapsed time between the assertion of the timer interrupt and entry into the interrupt service routine.
- Efficient transfers using Micro Direct Memory Access Controller ( $\mu$ DMA)
  - Dedicated channel for each timer
  - Burst request generated on timer interrupt

### 11.1 Block Diagram

**Note:** In Figure 11-1 on page 382, the specific CCP pins available depend on the Stellaris® device. See Table 11-1 on page 382 for the available CCPs.

Figure 11-1. GPTM Module Block Diagram

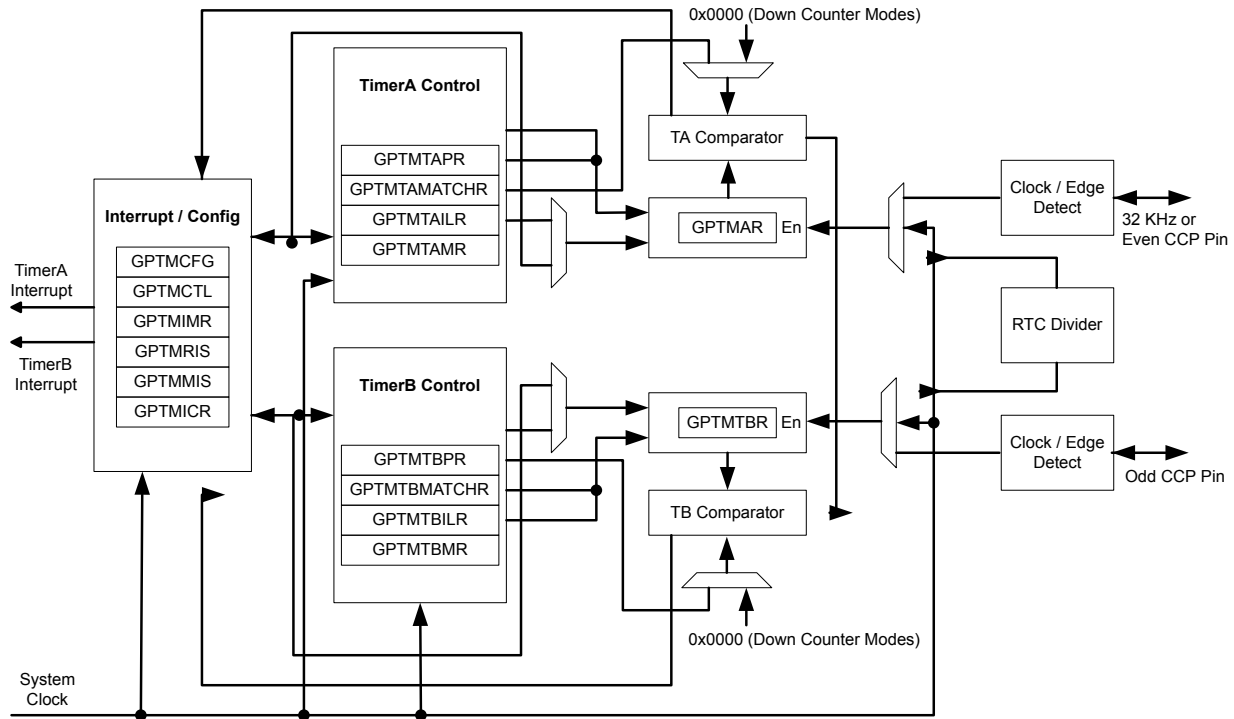


Table 11-1. Available CCP Pins

Timer	16-Bit Up/Down Counter	Even CCP Pin	Odd CCP Pin
Timer 0	Timer A	CCP0	-
	Timer B	-	CCP1
Timer 1	Timer A	CCP2	-
	Timer B	-	CCP3
Timer 2	Timer A	CCP4	-
	Timer B	-	CCP5
Timer 3	Timer A	CCP6	-
	Timer B	-	CCP7

## 11.2 Functional Description

The main components of each GPTM block are two free-running 16-bit up/down counters (referred to as Timer A and Timer B), two 16-bit match registers, 2 16-bit shadow 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 394), the **GPTM Timer A Mode (GPTMTAMR)** register (see page 395), and the **GPTM Timer B Mode (GPTMTBMR)** register (see page 397). When in one of the 32-bit modes, the timer can only act as a 32-bit timer. However, when configured in 16-bit mode, the GPTM can have its two 16-bit timers configured in any combination of the 16-bit modes.

### 11.2.1 GPTM Reset Conditions

After reset has been applied to the GPTM module, the module is in an inactive state, and all control registers are cleared and in their default states. Counters Timer A and Timer B are initialized to 0xFFFF, along with their corresponding load registers: the **GPTM Timer A Interval Load (GPTMTAILR)** register (see page 410) and the **GPTM Timer B Interval Load (GPTMTBILR)** register (see page 411) and shadow registers: the **GPTM Timer A Value (GPTMTAV)** register (see page 418) and the **GPTM Timer B Value (GPTMTBV)** register (see page 419). The prescale counters are initialized to 0x00: the **GPTM Timer A Prescale (GPTMTAPR)** register (see page 414) and the **GPTM Timer B Prescale (GPTMTBPR)** register (see page 415).

### 11.2.2 32-Bit Timer Operating Modes

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

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

- **GPTM Timer A Interval Load (GPTMTAILR)** register [15:0], see page 410
- **GPTM Timer B Interval Load (GPTMTBILR)** register [15:0], see page 411
- **GPTM Timer A (GPTMTAR)** register [15:0], see page 416
- **GPTM Timer B (GPTMTBR)** register [15:0], see page 417
- **GPTM Timer A Value (GPTMTAV)** register [15:0], see page 418
- **GPTM Timer B Value (GPTMTBV)** register [15:0], see page 419

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]
```

A read access to **GPTMTAV** returns the value:

```
GPTMTBV[15:0]:GPTMTAV[15:0]
```

#### 11.2.2.1 32-Bit One-Shot/Periodic Timer Mode

In 32-bit one-shot and periodic timer modes, the concatenated versions of the Timer A and Timer B registers are configured as a 32-bit up or down counter. The selection of one-shot or periodic mode is determined by the value written to the **TAMR** field of the **GPTM Timer A Mode (GPTMTAMR)** register (see page 395), and there is no need to write to the **GPTM Timer B Mode (GPTMTBMR)** register.

When software sets the **TAEN** bit in the **GPTM Control (GPTMCTL)** register (see page 399), the timer begins counting up or down from its preloaded value. Alternatively, if the **TAWOT** bit is set in the **GPTMTAMR** register, once the **TAEN** bit is set, the timer waits for the trigger from the previous timer to begin counting. This mode allows the timer modules to be daisy chained such that a single

timer can initiate multiple timing events. Care must be taken not to set the `TAWOT` bit in the **GPTMTAMR** register of GP Timer Module 0.

Once the time-out event (0x0000.0000 when counting down, 0xFFFF.FFFF when counting up) 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. If the `TnSNAPS` bit in the **GPTMTnMR** register is set, the actual free-running value of the timer at the time-out event is loaded into the **GPTMTAR** register. In this manner, software can determine the time elapsed from the interrupt assertion to the ISR entry.

In addition to reloading the count value, the GPTM generates interrupts and triggers when it reaches the time-out event. The GPTM sets the `TATORIS` bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register (see page 404), and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register (see page 408). If the time-out interrupt is enabled in the **GPTM Interrupt Mask (GPTIMR)** register (see page 402), the GPTM also sets the `TATOMIS` bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register (see page 406). By setting the `TAMIE` bit in the **GPTMTAMR** register, an interrupt can also be generated when the Timer A value equals the value loaded into the **GPTM Timer A Match (GPTMTAMATCH)** register. This interrupt has the same status, masking, and clearing functions as the time-out interrupt. The ADC trigger is enabled by setting the `TAOTE` bit in **GPTMCTL**. The  $\mu$ DMA trigger is enabled by configuring and enabling the appropriate  $\mu$ DMA channel. See “Channel Configuration” on page 223.

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

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

### 11.2.2.2 32-Bit Real-Time Clock Timer Mode

In Real-Time Clock (RTC) mode, the concatenated versions of the Timer A and Timer B 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 Timer A Match (GPTMTAMATCHR)** register (see page 412) by the controller.

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

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

In addition to generating interrupts, a  $\mu$ DMA trigger can be generated. The  $\mu$ DMA trigger is enabled by configuring and enabling the appropriate  $\mu$ DMA channel. See “Channel Configuration” on page 223.

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

### 11.2.3 16-Bit Timer Operating Modes

The GPTM is placed into global 16-bit mode by writing a value of 0x4 to the **GPTM Configuration (GPTMCFG)** register (see page 394). This section describes each of the GPTM 16-bit modes of

operation. Timer A and Timer B have identical modes, so a single description is given using an *n* to reference both.

### 11.2.3.1 16-Bit One-Shot/Periodic Timer Mode

In 16-bit one-shot and periodic timer modes, the timer is configured as a 16-bit up or 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 Timer n Prescale (GPTMTnPR)** register.

When software sets the *TnEN* bit in the **GPTMCTL** register, the timer begins counting up or down from its preloaded value. Alternatively, if the *TnWOT* bit is set in the **GPTMTnMR** register, once the *TnEN* bit is set, the timer waits for the external trigger to begin counting. This mode allows the timer modules to be daisy chained such that a single timer can initiate multiple timing events.

Once the time-out event (0x0000 when counting down, 0xFFFF when counting up) 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. If the *TnSNAPS* bit in the **GPTMTnMR** register is set, the actual free-running value of the timer at the time-out event is loaded into the **GPTMTAR** register. In this manner, software can determine the time elapsed from the interrupt assertion to the ISR entry.

In addition to reloading the count value, the timer generates interrupts and triggers when it reaches the time-out event. 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. By setting the *TnMIE* bit in the **GPTMTnMR** register, an interrupt can also be generated when the timer value equals the value loaded into the **GPTM Timer n Match (GPTMTnMATCH)** register. This interrupt has the same status, masking, and clearing functions as the time-out interrupt. The ADC trigger is enabled by setting the *TnOTE* bit in the **GPTMCTL** register. The  $\mu$ DMA trigger is enabled by configuring and enabling the appropriate  $\mu$ DMA channel. See “Channel Configuration” on page 223.

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 an 80-MHz clock with  $T_c=12.5$  ns (clock period).

**Table 11-2. 16-Bit Timer With Prescaler Configurations**

Prescale	#Clock ( $T_c$ ) <sup>a</sup>	Max Time	Units
00000000	1	0.8192	mS
00000001	2	1.6385	mS
00000010	3	2.4576	mS
-----	--	--	--
11111100	254	208.0768	mS
11111110	255	208.896	mS
11111111	256	209.7152	mS

a.  $T_c$  is the clock period.

### 11.2.3.2 16-Bit Input Edge-Count Mode

**Note:** For rising-edge detection, the input signal must be High for at least two system clock periods following the rising edge. Similarly, for falling-edge detection, the input signal must be Low for at least two system clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is 1/4 of the system frequency.

In Edge-Count mode, 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. In this mode, the timer is capable of capturing three types of events: rising edge, falling edge, or both. To place the timer in Edge-Count mode, the  $T_nCMR$  bit of the **GPTMTnMR** register must be set to 0. The type of edge that the timer counts is determined by the  $T_nEVENT$  fields of the **GPTMCTL** register. During initialization, the **GPTM Timer n 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. The optional prescaler is loaded into the **GPTM Timer n Prescale (GPTMTnPR)** register.

When software writes the  $T_nEN$  bit in the **GPTM Control (GPTMCTL)** register, the timer is enabled for event capture. Each input event on the CCP pin decrements the counter by 1 until the event count matches **GPTMTnMATCHR**. When the counts match, the GPTM asserts the  $C_nMRIS$  bit in the **GPTMRIS** register (and the  $C_nMMIS$  bit, if the interrupt is not masked).

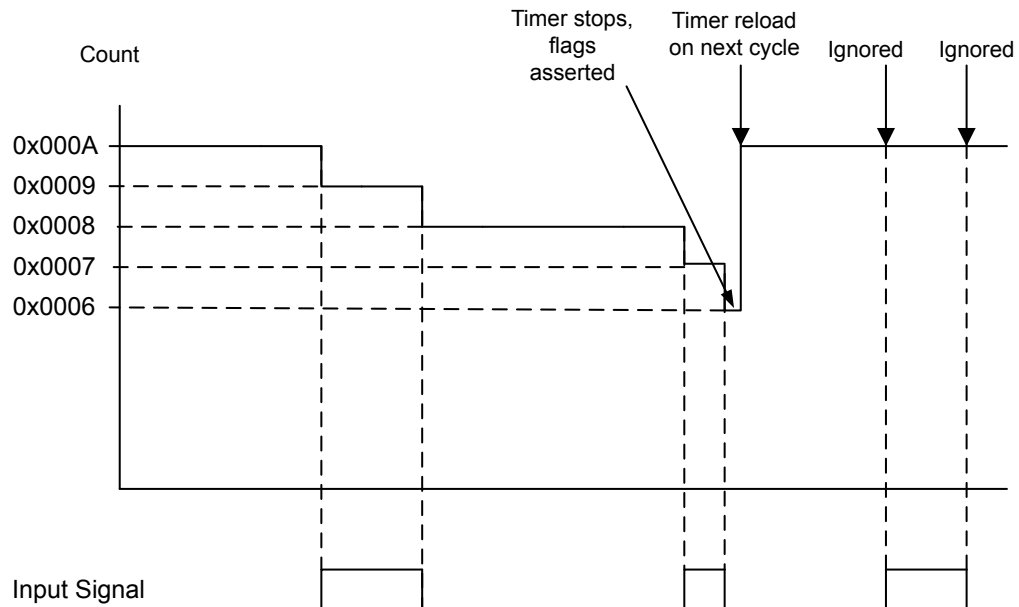
In addition to generating interrupts, a  $\mu$ DMA trigger can be generated. The  $\mu$ DMA trigger is enabled by configuring and enabling the appropriate  $\mu$ DMA channel. See “Channel Configuration” on page 223.

The counter is then reloaded using the value in **GPTMTnILR**, and stopped because the GPTM automatically clears the  $T_nEN$  bit in the **GPTMCTL** register. Once the event count has been reached, all further events are ignored until  $T_nEN$  is re-enabled by software. The **GPTMTnV** contains the free-running timer value and can be read to determine the time that elapsed between the interrupt assertion and the entry into the ISR.

Figure 11-2 on page 387 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  $T_nEN$  bit after the current count matches the value in the **GPTMnMR** register.

Figure 11-2. 16-Bit Input Edge-Count Mode Example



### 11.2.3.3 16-Bit Input Edge-Time Mode

**Note:** For rising-edge detection, the input signal must be High for at least two system clock periods following the rising edge. Similarly, for falling edge detection, the input signal must be Low for at least two system clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is 1/4 of the system frequency.

In Edge-Time mode, the timer is configured as a 16-bit free-running down-counter with an optional 8-bit prescaler that effectively extends the counting range of the timer to 24 bits. In this mode, the timer is initialized to the value loaded in the **GPTMTnILR** register (or 0xFFFF at reset). This mode allows for event capture of either rising or falling edges, but not both. The timer is placed into Edge-Time mode by setting the **TnCMR** bit in the **GPTMTnMR** register, and the type of event that the timer captures is determined by the **TnEVENT** fields of the **GPTMCnTL** register. The optional prescaler is loaded into the **GPTM Timer n Prescale (GPTMTnPR)** 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). The **GPTMTnV** is the free-running value of the timer and can be read to determine the time that elapsed between the interrupt assertion and the entry into the ISR.

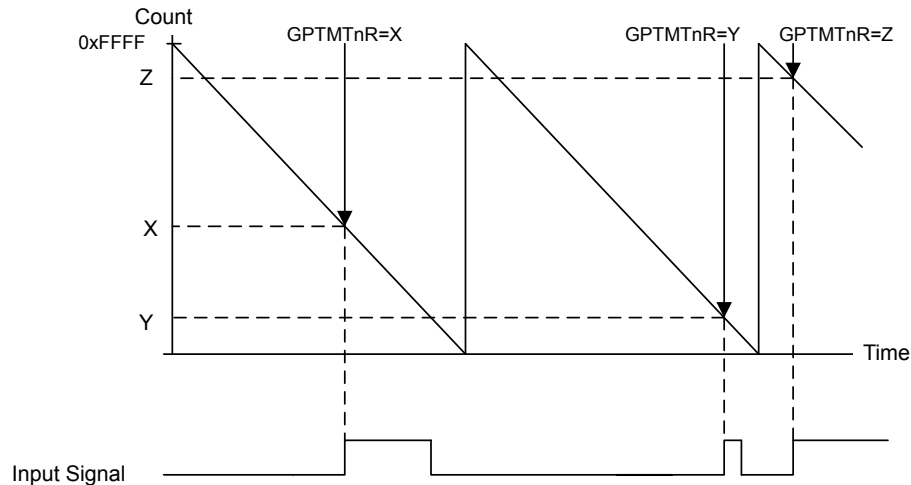
In addition to generating interrupts, a  $\mu$ DMA trigger can be generated. The  $\mu$ DMA trigger is enabled by configuring and enabling the appropriate  $\mu$ DMA channel. See “Channel Configuration” on page 223.

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

Figure 11-3 on page 388 shows how input edge timing mode works. In the diagram, it is assumed that the start value of the timer is the default value of 0xFFFF, and the timer is configured to capture rising edge events.

Each time a rising edge event is detected, the current count value is loaded into the **GPTMTnR** register, and is held there until another rising edge is detected (at which point the new count value is loaded into **GPTMTnR**).

**Figure 11-3. 16-Bit Input Edge-Time Mode Example**



#### 11.2.3.4 16-Bit PWM Mode

**Note:** The prescaler is not available in 16-Bit PWM mode.

The GPTM supports a simple PWM generation mode. In PWM mode, the timer is configured as a down-counter with a start value (and thus period) defined by **GPTMTnILR**. PWM mode is enabled with the **GPTMTnMR** register by setting the  $TnAMS$  bit to 0x1, the  $TnCMR$  bit to 0x0, and the  $TnMR$  field to 0x2.

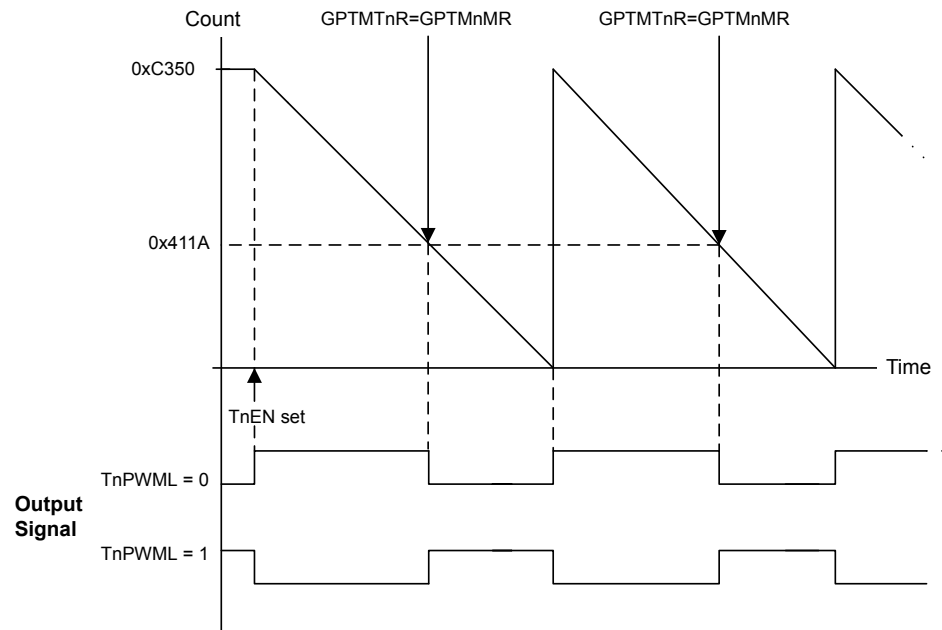
When software writes the  $TnEN$  bit in the **GPTMCTL** register, the counter begins counting down until it reaches the 0x0000 state. On the next counter cycle, the counter reloads its start value from **GPTMTnILR** and continues counting until disabled by software clearing the  $TnEN$  bit in the **GPTMCTL** register. No interrupts or status bits are asserted in PWM mode.

The output PWM signal asserts when the counter is at the value of the **GPTMTnILR** register (its start state), and is deasserted when the counter value equals the value in the **GPTM Timer n Match Register (GPTMnMATCHR)**. Software has the capability of inverting the output PWM signal by setting the  $TnPWML$  bit in the **GPTMCTL** register.

Figure 11-4 on page 389 shows how to generate an output PWM with a 1-ms period and a 66% duty cycle assuming a 50-MHz input clock and  $TnPWML = 0$  (duty cycle would be 33% for the  $TnPWML = 1$  configuration). For this example, the start value is **GPTMnILR**=0xC350 and the match value is **GPTMnMR**=0x411A.



Figure 11-4. 16-Bit PWM Mode Example



## 11.2.4 DMA Operation

The timers each have a dedicated  $\mu$ DMA channel and can provide a request signal to the  $\mu$ DMA controller. The request signal is a burst type, and will occur whenever a timer raw interrupt condition occurs. The arbitration size of the  $\mu$ DMA transfer should be set to the amount of data that should be transferred whenever a timer event occurs.

For example, to transfer 256 items, 8 items at a time every 10 ms, configure a timer to generate a periodic timeout at 10 ms. Configure the  $\mu$ DMA transfer for a total of 256 items, with a burst size of 8 items. Each time the timer times out, the  $\mu$ DMA controller will transfer 8 items, until all 256 items have been transferred.

No other special steps are needed to enable Timers for  $\mu$ DMA operation. Refer to “Micro Direct Memory Access ( $\mu$ DMA)” on page 219 for more details about programming the  $\mu$ DMA controller.

## 11.3 Initialization and Configuration

To use the general-purpose timers, the peripheral clock must be enabled by setting the `TIMER0`, `TIMER1`, `TIMER2`, and `TIMER3` bits in the `RCGC1` register. See page 163. If using any CCP pins, the clock to the appropriate GPIO module must be enabled via the `RCGC2` register in the System Control module. See page 175. To find out which GPIO port to enable, refer to Table 24-5 on page 937.

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

### 11.3.1 32-Bit One-Shot/Periodic Timer Mode

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

1. Ensure the timer is disabled (the `TAEN` bit in the **GPTMCTL** register is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x0.
3. Configure the `TAMR` field in the **GPTM Timer A Mode Register (GPTMTAMR)**:
  - a. Write a value of 0x1 for One-Shot mode.
  - b. Write a value of 0x2 for Periodic mode.
4. Optionally configure the `TASNAPS`, `TAWOT`, `TAMTE`, and `TACDIR` bits in the **GPTMTAMR** register to select whether to capture the value of the free-running timer at time-out, use an external trigger to start counting, configure an additional trigger or interrupt, and count up or down.
5. Load the start value into the **GPTM Timer A Interval Load Register (GPTMTAILR)**.
6. If interrupts are required, set the appropriate bits in the **GPTM Interrupt Mask Register (GPTMIMR)**.
7. Set the `TAEN` bit in the **GPTMCTL** register to enable the timer and start counting.
8. Poll 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 appropriate bit of the **GPTM Interrupt Clear Register (GPTMICR)**.

In One-Shot mode, the timer stops counting after step 7 on page 390. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode does not stop counting after it times out.

### 11.3.2 32-Bit Real-Time Clock (RTC) Mode

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

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

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

### 11.3.3 16-Bit One-Shot/Periodic Timer Mode

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

1. Ensure the timer is disabled (the `TnEN` bit is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x4.

3. Set the  $TnMR$  field in the **GPTM Timer Mode (GPTMTnMR)** register:
  - a. Write a value of 0x1 for One-Shot mode.
  - b. Write a value of 0x2 for Periodic mode.
4. Optionally configure the  $TnSNAPS$ ,  $TnWOT$ ,  $TnMTE$  and  $TnCDIR$  bits in the **GPTMTnMR** register to select whether to capture the value of the free-running timer at time-out, use an external trigger to start counting, configure an additional trigger or interrupt, and count up or down.
5. If a prescaler is to be used, write the prescale value to the **GPTM Timer n Prescale Register (GPTMTnPR)**.
6. Load the start value into the **GPTM Timer Interval Load Register (GPTMTnILR)**.
7. If interrupts are required, set the appropriate bit in the **GPTM Interrupt Mask Register (GPTMIMR)**.
8. Set the  $TnEN$  bit in the **GPTM Control Register (GPTMCTL)** to enable the timer and start counting.
9. Poll 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 appropriate bit of the **GPTM Interrupt Clear Register (GPTMICR)**.

In One-Shot mode, the timer stops counting after step 8 on page 391. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode does not stop counting after it times out.

### 11.3.4 16-Bit Input Edge-Count Mode

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

1. Ensure the timer is disabled (the  $TnEN$  bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x4.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the  $TnCnMR$  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. If a prescaler is to be used, write the prescale value to the **GPTM Timer n Prescale Register (GPTMTnPR)**.
6. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** register.
7. Load the desired event count into the **GPTM Timer n Match (GPTMTnMATCHR)** register.
8. If interrupts are required, set the  $CnMIM$  bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
9. Set the  $TnEN$  bit in the **GPTMCTL** register to enable the timer and begin waiting for edge events.
10. 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 391 through step 9 on page 391.

### 11.3.5 16-Bit Input Edge Timing Mode

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

1. Ensure the timer is disabled (the  $TnEN$  bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x4.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the  $TnCMR$  field to 0x1 and the  $TnMR$  field to 0x3.
4. Configure the type of event that the timer captures by writing the  $TnEVENT$  field of the **GPTM Control (GPTMCTL)** register.
5. If a prescaler is to be used, write the prescale value to the **GPTM Timer n Prescale Register (GPTMTnPR)**.
6. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** register.
7. If interrupts are required, set the  $CnEIM$  bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
8. Set the  $TnEN$  bit in the **GPTM Control (GPTMCTL)** register to enable the timer and start counting.
9. 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 Timer n (GPTMTnR)** register.

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

### 11.3.6 16-Bit PWM Mode

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

1. Ensure the timer is disabled (the  $TnEN$  bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x4.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, set the  $TnAMS$  bit to 0x1, the  $TnCMR$  bit to 0x0, and the  $TnMR$  field to 0x2.
4. Configure the output state of the PWM signal (whether or not it is inverted) in the  $TnEVENT$  field of the **GPTM Control (GPTMCTL)** register.
5. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** register.
6. Load the **GPTM Timer n Match (GPTMTnMATCHR)** register with the desired value.
7. Set the  $TnEN$  bit in the **GPTM Control (GPTMCTL)** register to enable the timer and begin generation of the output PWM signal.

In PWM Timing mode, the timer continues running after the PWM signal has been generated. The PWM period can be adjusted at any time by writing the **GPTMTnILR** register, and the change takes effect at the next cycle after the write.

## 11.4 Register Map

Table 11-3 on page 393 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

Note that the GP Timer module clock must be enabled before the registers can be programmed (see page 163).

**Table 11-3. Timers Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	GPTMCFG	R/W	0x0000.0000	GPTM Configuration	394
0x004	GPTMTAMR	R/W	0x0000.0000	GPTM Timer A Mode	395
0x008	GPTMTBMR	R/W	0x0000.0000	GPTM Timer B Mode	397
0x00C	GPTMCTL	R/W	0x0000.0000	GPTM Control	399
0x018	GPTMIMR	R/W	0x0000.0000	GPTM Interrupt Mask	402
0x01C	GPTMRIS	RO	0x0000.0000	GPTM Raw Interrupt Status	404
0x020	GPTMMIS	RO	0x0000.0000	GPTM Masked Interrupt Status	406
0x024	GPTMICR	W1C	0x0000.0000	GPTM Interrupt Clear	408
0x028	GPTMTAILR	R/W	0xFFFF.FFFF	GPTM Timer A Interval Load	410
0x02C	GPTMTBILR	R/W	0x0000.FFFF	GPTM Timer B Interval Load	411
0x030	GPTMTAMATCHR	R/W	0xFFFF.FFFF	GPTM Timer A Match	412
0x034	GPTMTBMATCHR	R/W	0x0000.FFFF	GPTM Timer B Match	413
0x038	GPTMTAPR	R/W	0x0000.0000	GPTM Timer A Prescale	414
0x03C	GPTMTBPR	R/W	0x0000.0000	GPTM Timer B Prescale	415
0x048	GPTMTAR	RO	0xFFFF.FFFF	GPTM Timer A	416
0x04C	GPTMTBR	RO	0x0000.FFFF	GPTM Timer B	417
0x050	GPTMTAV	RO	0xFFFF.FFFF	GPTM Timer A Value	418
0x054	GPTMTBV	RO	0x0000.FFFF	GPTM Timer B Value	419

## 11.5 Register Descriptions

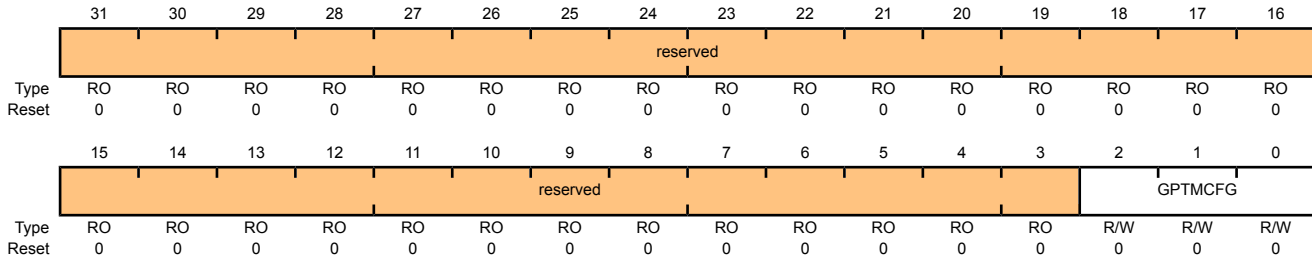
The remainder of this section lists and describes the GPTM registers, in numerical order by address offset.

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

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

#### GPTM Configuration (GPTMCFG)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x000  
 Type R/W, reset 0x0000.0000



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

The GPTMCFG values are defined as follows:

Value	Description
0x0	32-bit timer configuration.
0x1	32-bit real-time clock (RTC) counter configuration.
0x2-0x3	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 Timer A 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.

In 16-bit timer configuration, **TAMR** controls the 16-bit timer modes for Timer A. In 32-bit timer configuration, this register controls the mode, and the contents of **GPTMTBMR** are ignored.

### GPTM Timer A Mode (GPTMTAMR)

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

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TASNAPS	R/W	0	GPTM Timer A Snap-Shot Mode  Value Description 0 Snap-shot mode is disabled. 1 If Timer A is configured in the periodic mode, the actual free-running value of Timer A is loaded at the time-out event into the <b>GPTM Timer A (GPTMTAR)</b> register.
6	TAWOT	R/W	0	GPTM Timer A Wait-on-Trigger  Value Description 0 Timer A begins counting as soon as it is enabled. 1 If Timer A is enabled ( <b>TAEN</b> is set in the <b>GPTMCTL</b> register), Timer A does not begin counting until it receives a trigger from the timer in the previous position in the daisy chain. This function is valid for both one-shot and periodic modes.  This bit must be clear for GP Timer Module 0, Timer A.

Bit/Field	Name	Type	Reset	Description
5	TAMIE	R/W	0	<p>GPTM Timer A Match Interrupt Enable</p> <p>Value Description</p> <p>0 The match interrupt is disabled.</p> <p>1 An interrupt is generated when the match value in the <b>GPTMTAMATCHR</b> register is reached in the one-shot and periodic modes.</p>
4	TACDIR	R/W	0	<p>GPTM Timer A Count Direction</p> <p>Value Description</p> <p>0 The timer counts down.</p> <p>1 When in one-shot or periodic mode, the timer counts up. When counting up, the timer starts from a value of 0x0000.</p>
3	TAAMS	R/W	0	<p>GPTM Timer A Alternate Mode Select</p> <p>The <b>TAAMS</b> values are defined as follows:</p> <p>Value Description</p> <p>0 Capture mode is enabled.</p> <p>1 PWM mode is enabled.</p> <p><b>Note:</b> To enable PWM mode, you must also clear the <b>TACMR</b> bit and set the <b>TAMR</b> field to 0x2.</p>
2	TACMR	R/W	0	<p>GPTM Timer A Capture Mode</p> <p>The <b>TACMR</b> values are defined as follows:</p> <p>Value Description</p> <p>0 Edge-Count mode</p> <p>1 Edge-Time mode</p>
1:0	TAMR	R/W	0x0	<p>GPTM Timer A Mode</p> <p>The <b>TAMR</b> values are defined as follows:</p> <p>Value Description</p> <p>0x0 Reserved</p> <p>0x1 One-Shot Timer mode</p> <p>0x2 Periodic Timer mode</p> <p>0x3 Capture mode</p> <p>The Timer mode is based on the timer configuration defined by bits 2:0 in the <b>GPTMCFG</b> register (16-or 32-bit).</p>



### Register 3: GPTM Timer B 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.

In 16-bit timer configuration, these bits control the 16-bit timer modes for Timer B. In 32-bit timer configuration, this register's contents are ignored, and **GPTMTAMR** is used.

#### GPTM Timer B 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

	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								TBSNAPS	TBWOT	TBMIE	TBCDIR	TBAMS	TBCMR	TBMR	
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TBSNAPS	R/W	0	GPTM Timer B Snap-Shot Mode  Value Description 0 Snap-shot mode is disabled. 1 If Timer B is configured in the periodic mode, the actual free-running value of Timer B is loaded at the time-out event into the <b>GPTM Timer B (GPTMTBR)</b> register.
6	TBWOT	R/W	0	GPTM Timer B Wait-on-Trigger  Value Description 0 Timer B begins counting as soon as it is enabled. 1 If Timer B is enabled ( <b>TBEN</b> is set in the <b>GPTMCTL</b> register), Timer B does not begin counting until it receives an it receives a trigger from the timer in the previous position in the daisy chain. This function is valid for both one-shot and periodic modes.

Bit/Field	Name	Type	Reset	Description
5	TBMIE	R/W	0	<p>GPTM Timer B Match Interrupt Enable</p> <p>Value Description</p> <p>0 The match interrupt is disabled.</p> <p>1 An interrupt is generated when the match value in the <b>GPTMTBMATCHR</b> register is reached in the one-shot and periodic modes.</p>
4	TBCDIR	R/W	0	<p>GPTM Timer B Count Direction</p> <p>Value Description</p> <p>0 The timer counts down.</p> <p>1 When in one-shot or periodic mode, the timer counts up. When counting up, the timer starts from a value of 0x0000.</p>
3	TBAMS	R/W	0	<p>GPTM Timer B Alternate Mode Select</p> <p>The <b>TBAMS</b> values are defined as follows:</p> <p>Value Description</p> <p>0 Capture mode is enabled.</p> <p>1 PWM mode is enabled.</p> <p><b>Note:</b> To enable PWM mode, you must also clear the <b>TBCMR</b> bit and set the <b>TBMR</b> field to 0x2.</p>
2	TBCMR	R/W	0	<p>GPTM Timer B Capture Mode</p> <p>The <b>TBCMR</b> values are defined as follows:</p> <p>Value Description</p> <p>0 Edge-Count mode</p> <p>1 Edge-Time mode</p>
1:0	TBMR	R/W	0x0	<p>GPTM Timer B Mode</p> <p>The <b>TBMR</b> values are defined as follows:</p> <p>Value Description</p> <p>0x0 Reserved</p> <p>0x1 One-Shot Timer mode</p> <p>0x2 Periodic Timer mode</p> <p>0x3 Capture mode</p> <p>The timer mode is based on the timer configuration defined by bits 2:0 in the <b>GPTMCFG</b> register.</p>

## Register 4: GPTM Control (GPTMCTL), offset 0x00C

This register is used alongside the **GPTMCFG** and **GMTMTnMR** registers to fine-tune the timer configuration, and to enable other features such as timer stall and the output trigger. The output trigger can be used to initiate transfers on the ADC module.

### GPTM Control (GPTMCTL)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x00C  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	TBPWML	TBOTE	reserved	TBEVENT	TBSTALL	TBEN	reserved	TAPWML	TAOTE	RTCEN	TAEVENT	TASTALL	TAEN		
Type	RO	R/W	R/W	RO	R/W	R/W	R/W	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description						
31:15	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
14	TBPWML	R/W	0	GPTM Timer B PWM Output Level  The <b>TBPWML</b> values are defined as follows:  <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>Output is unaffected.</td> </tr> <tr> <td>1</td> <td>Output is inverted.</td> </tr> </table>	Value	Description	0	Output is unaffected.	1	Output is inverted.
Value	Description									
0	Output is unaffected.									
1	Output is inverted.									
13	TBOTE	R/W	0	GPTM Timer B Output Trigger Enable  The <b>TBOTE</b> values are defined as follows:  <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>The output Timer B ADC trigger is disabled.</td> </tr> <tr> <td>1</td> <td>The output Timer B ADC trigger is enabled.</td> </tr> </table> <p>In addition, the ADC must be enabled and the timer selected as a trigger source with the <b>EMn</b> bit in the <b>ADCEMUX</b> register (see page 470).</p>	Value	Description	0	The output Timer B ADC trigger is disabled.	1	The output Timer B ADC trigger is enabled.
Value	Description									
0	The output Timer B ADC trigger is disabled.									
1	The output Timer B ADC trigger is enabled.									
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						

Bit/Field	Name	Type	Reset	Description										
11:10	TBEVENT	R/W	0x0	<p>GPTM Timer B Event Mode</p> <p>The TBEVENT values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Positive edge</td> </tr> <tr> <td>0x1</td> <td>Negative edge</td> </tr> <tr> <td>0x2</td> <td>Reserved</td> </tr> <tr> <td>0x3</td> <td>Both edges</td> </tr> </tbody> </table>	Value	Description	0x0	Positive edge	0x1	Negative edge	0x2	Reserved	0x3	Both edges
Value	Description													
0x0	Positive edge													
0x1	Negative edge													
0x2	Reserved													
0x3	Both edges													
9	TBSTALL	R/W	0	<p>GPTM Timer B Stall Enable</p> <p>The TBSTALL values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Timer B stalling is disabled.</td> </tr> <tr> <td>1</td> <td>Timer B stalling is enabled.</td> </tr> </tbody> </table>	Value	Description	0	Timer B stalling is disabled.	1	Timer B stalling is enabled.				
Value	Description													
0	Timer B stalling is disabled.													
1	Timer B stalling is enabled.													
8	TBEN	R/W	0	<p>GPTM Timer B Enable</p> <p>The TBEN values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Timer B is disabled.</td> </tr> <tr> <td>1</td> <td>Timer B is enabled and begins counting or the capture logic is enabled based on the <b>GPTMCFG</b> register.</td> </tr> </tbody> </table>	Value	Description	0	Timer B is disabled.	1	Timer B is enabled and begins counting or the capture logic is enabled based on the <b>GPTMCFG</b> register.				
Value	Description													
0	Timer B is disabled.													
1	Timer B is enabled and begins counting or the capture logic is enabled based on the <b>GPTMCFG</b> register.													
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
6	TAPWML	R/W	0	<p>GPTM Timer A PWM Output Level</p> <p>The TAPWML values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Output is unaffected.</td> </tr> <tr> <td>1</td> <td>Output is inverted.</td> </tr> </tbody> </table>	Value	Description	0	Output is unaffected.	1	Output is inverted.				
Value	Description													
0	Output is unaffected.													
1	Output is inverted.													
5	TAOTE	R/W	0	<p>GPTM Timer A Output Trigger Enable</p> <p>The TAOTE values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The output Timer A ADC trigger is disabled.</td> </tr> <tr> <td>1</td> <td>The output Timer A ADC trigger is enabled.</td> </tr> </tbody> </table> <p>In addition, the ADC must be enabled and the timer selected as a trigger source with the EMn bit in the <b>ADCEMUX</b> register (see page 470).</p>	Value	Description	0	The output Timer A ADC trigger is disabled.	1	The output Timer A ADC trigger is enabled.				
Value	Description													
0	The output Timer A ADC trigger is disabled.													
1	The output Timer A ADC trigger is enabled.													

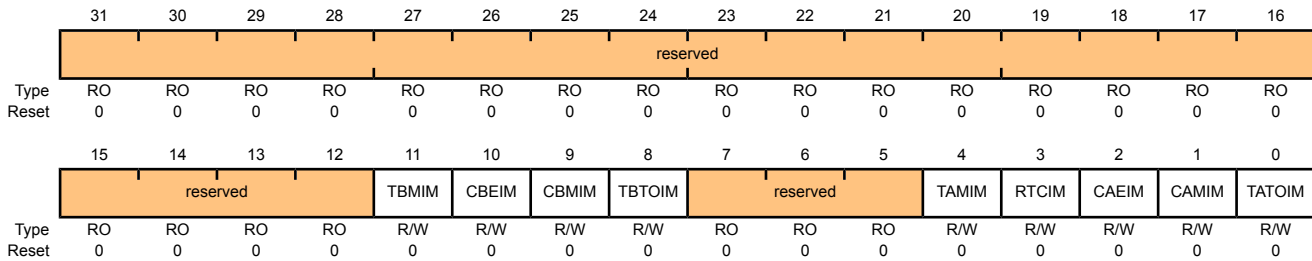
Bit/Field	Name	Type	Reset	Description										
4	RTCEN	R/W	0	<p>GPTM RTC Enable</p> <p>The <code>RTCEN</code> values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>RTC counting is disabled.</td> </tr> <tr> <td>1</td> <td>RTC counting is enabled.</td> </tr> </tbody> </table>	Value	Description	0	RTC counting is disabled.	1	RTC counting is enabled.				
Value	Description													
0	RTC counting is disabled.													
1	RTC counting is enabled.													
3:2	TAEVENT	R/W	0x0	<p>GPTM Timer A Event Mode</p> <p>The <code>TAEVENT</code> values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Positive edge</td> </tr> <tr> <td>0x1</td> <td>Negative edge</td> </tr> <tr> <td>0x2</td> <td>Reserved</td> </tr> <tr> <td>0x3</td> <td>Both edges</td> </tr> </tbody> </table>	Value	Description	0x0	Positive edge	0x1	Negative edge	0x2	Reserved	0x3	Both edges
Value	Description													
0x0	Positive edge													
0x1	Negative edge													
0x2	Reserved													
0x3	Both edges													
1	TASTALL	R/W	0	<p>GPTM Timer A Stall Enable</p> <p>The <code>TASTALL</code> values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Timer A stalling is disabled.</td> </tr> <tr> <td>1</td> <td>Timer A stalling is enabled.</td> </tr> </tbody> </table>	Value	Description	0	Timer A stalling is disabled.	1	Timer A stalling is enabled.				
Value	Description													
0	Timer A stalling is disabled.													
1	Timer A stalling is enabled.													
0	TAEN	R/W	0	<p>GPTM Timer A Enable</p> <p>The <code>TAEN</code> values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Timer A is disabled.</td> </tr> <tr> <td>1</td> <td>Timer A is enabled and begins counting or the capture logic is enabled based on the <code>GPTMCFG</code> register.</td> </tr> </tbody> </table>	Value	Description	0	Timer A is disabled.	1	Timer A is enabled and begins counting or the capture logic is enabled based on the <code>GPTMCFG</code> register.				
Value	Description													
0	Timer A is disabled.													
1	Timer A is enabled and begins counting or the capture logic is enabled based on the <code>GPTMCFG</code> register.													

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

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

#### GPTM Interrupt Mask (GPTMIMR)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x018  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:12	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	TBMIM	R/W	0	GPTM Timer B Mode Match Interrupt Mask The TBMIM values are defined as follows:  Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
10	CBEIM	R/W	0	GPTM Capture B 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 Capture B Match Interrupt Mask The CBMIM values are defined as follows:  Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.

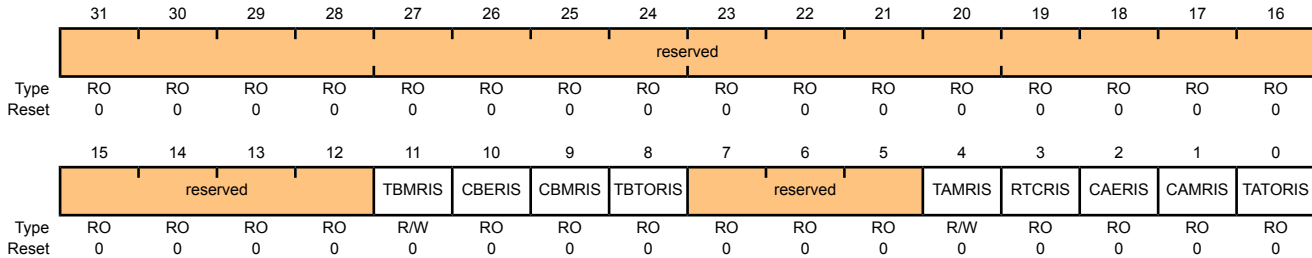
Bit/Field	Name	Type	Reset	Description
8	TBTOIM	R/W	0	GPTM Timer B Time-Out Interrupt Mask The TBTOIM values are defined as follows:  Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
7:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	TAMIM	R/W	0	GPTM Timer A Mode Match Interrupt Mask The TAMIM values are defined as follows:  Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
3	RTCIM	R/W	0	GPTM RTC Interrupt Mask The RTCIM values are defined as follows:  Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
2	CAEIM	R/W	0	GPTM Capture A Event Interrupt Mask The CAEIM values are defined as follows:  Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
1	CAMIM	R/W	0	GPTM Capture A Match Interrupt Mask The CAMIM values are defined as follows:  Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
0	TATOIM	R/W	0	GPTM Timer A Time-Out Interrupt Mask The TATOIM values are defined as follows:  Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.

## Register 6: GPTM Raw Interrupt Status (GPTMRIS), offset 0x01C

This register shows the state of the GPTM's internal interrupt signal. These bits are set whether or not the interrupt is masked in the **GPTMIMR** register. Each bit can be cleared by writing a 1 to its corresponding bit in **GPTMICR**.

### GPTM Raw Interrupt Status (GPTMRIS)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x01C  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:12	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	TBMRIS	R/W	0	GPTM Timer B Mode Match Raw Interrupt  This is the Timer B mode match interrupt status prior to masking.  When the <b>TBMIE</b> bit is set in the <b>GPTMTBMR</b> register, an interrupt is generated when the match value in the <b>GPTMTBMATCHR</b> register is reached when in the one-shot and periodic modes.
10	CBERIS	RO	0	GPTM Capture B Event Raw Interrupt  This is the Capture B event interrupt status prior to masking.
9	CBMRIS	RO	0	GPTM Capture B Match Raw Interrupt  This is the Capture B match interrupt status prior to masking.
8	TBTORIS	RO	0	GPTM Timer B Time-Out Raw Interrupt  This is the Timer B time-out interrupt status prior to masking.
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	TAMRIS	R/W	0	GPTM Timer A Mode Match Raw Interrupt  This is the Timer A mode match interrupt status prior to masking.  When the <b>TAMIE</b> bit is set in the <b>GPTMTAMR</b> register, an interrupt is generated when the match value in the <b>GPTMTAMATCHR</b> register is reached when in the one-shot and periodic modes.
3	RTCRIS	RO	0	GPTM RTC Raw Interrupt  This is the RTC event interrupt status prior to masking.



Bit/Field	Name	Type	Reset	Description
2	CAERIS	RO	0	GPTM Capture A Event Raw Interrupt This is the Capture A event interrupt status prior to masking.
1	CAMRIS	RO	0	GPTM Capture A Match Raw Interrupt This is the Capture A match interrupt status prior to masking.
0	TATORIS	RO	0	GPTM Timer A Time-Out Raw Interrupt This the Timer A time-out interrupt status prior to masking.

### Register 7: GPTM Masked Interrupt Status (GPTMMIS), offset 0x020

This register show the state of the GPTM's controller-level interrupt. If an interrupt is unmasked in **GPTMIMR**, and there is an event that causes the interrupt to be asserted, the corresponding bit is set in this register. All bits are cleared by writing a 1 to the corresponding bit in **GPTMICR**.

#### GPTM Masked Interrupt Status (GPTMMIS)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x020  
 Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved				TBMMIS	CBEMIS	CBMMIS	TBTOMIS	reserved				TAMMIS	RTCMIS	CAEMIS	CAMMIS	TATOMIS
Type	RO	RO	RO	RO	R/W	RO	RO	RO	RO	RO	RO	R/W	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bit/Field	Name	Type	Reset	Description
31:12	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	TBMMIS	R/W	0	GPTM Timer B Mode Match Masked Interrupt This is the Timer B Mode Match interrupt status after masking.
10	CBEMIS	RO	0	GPTM Capture B Event Masked Interrupt This is the Capture B event interrupt status after masking.
9	CBMMIS	RO	0	GPTM Capture B Match Masked Interrupt This is the Capture B match interrupt status after masking.
8	TBTOMIS	RO	0	GPTM Timer B Time-Out Masked Interrupt This is the Timer B time-out interrupt status after masking.
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	TAMMIS	R/W	0	GPTM Timer A Mode Match Masked Interrupt This is the Timer A Mode Match interrupt status after masking.
3	RTCMIS	RO	0	GPTM RTC Masked Interrupt This is the RTC event interrupt status after masking.
2	CAEMIS	RO	0	GPTM Capture A Event Masked Interrupt This is the Capture A event interrupt status after masking.
1	CAMMIS	RO	0	GPTM Capture A Match Masked Interrupt This is the Capture A match interrupt status after masking.

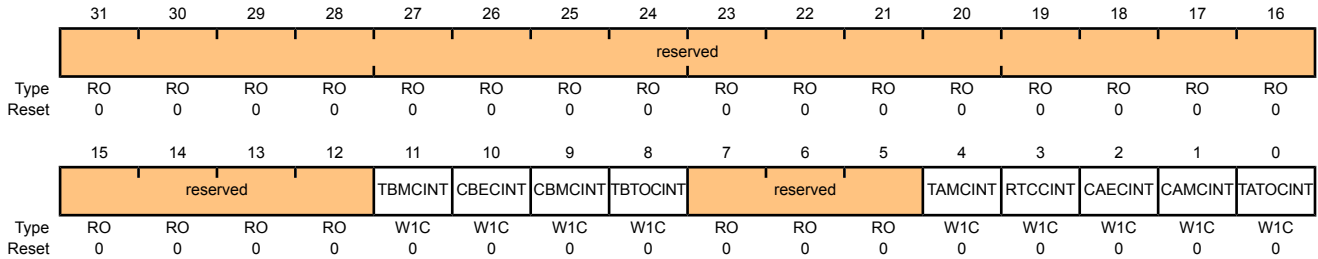
Bit/Field	Name	Type	Reset	Description
0	TATOMIS	RO	0	GPTM Timer A Time-Out Masked Interrupt This is the Timer A time-out interrupt status after masking.

### 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
31:12	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	TBMCINT	W1C	0	GPTM Timer B Mode Match Interrupt Clear  Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
10	CBECINT	W1C	0	GPTM Capture B 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 Capture B Match Interrupt Clear  The CBMCINT values are defined as follows:  Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.

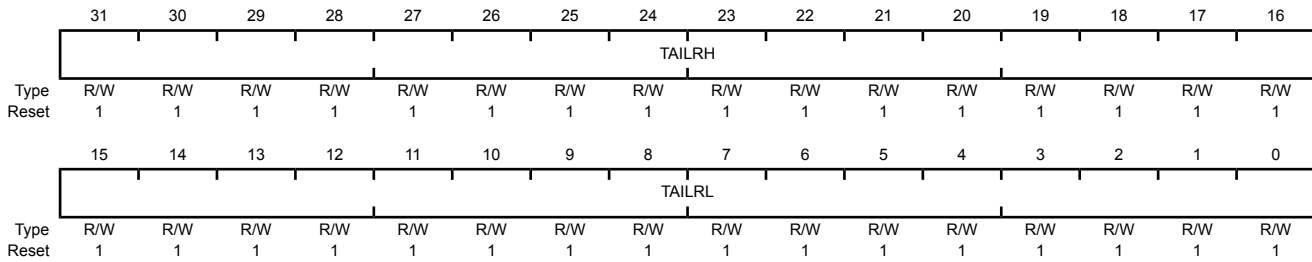
Bit/Field	Name	Type	Reset	Description						
8	TBTOCINT	W1C	0	<p>GPTM Timer B Time-Out Interrupt Clear</p> <p>The TBTOCINT values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The interrupt is unaffected.</td> </tr> <tr> <td>1</td> <td>The interrupt is cleared.</td> </tr> </tbody> </table>	Value	Description	0	The interrupt is unaffected.	1	The interrupt is cleared.
Value	Description									
0	The interrupt is unaffected.									
1	The interrupt is cleared.									
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
4	TAMCINT	W1C	0	<p>GPTM Timer A Mode Match Interrupt Clear</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The interrupt is unaffected.</td> </tr> <tr> <td>1</td> <td>The interrupt is cleared.</td> </tr> </tbody> </table>	Value	Description	0	The interrupt is unaffected.	1	The interrupt is cleared.
Value	Description									
0	The interrupt is unaffected.									
1	The interrupt is cleared.									
3	RTCCINT	W1C	0	<p>GPTM RTC Interrupt Clear</p> <p>The RTCCINT values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The interrupt is unaffected.</td> </tr> <tr> <td>1</td> <td>The interrupt is cleared.</td> </tr> </tbody> </table>	Value	Description	0	The interrupt is unaffected.	1	The interrupt is cleared.
Value	Description									
0	The interrupt is unaffected.									
1	The interrupt is cleared.									
2	CAECINT	W1C	0	<p>GPTM Capture A Event Interrupt Clear</p> <p>The CAECINT values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The interrupt is unaffected.</td> </tr> <tr> <td>1</td> <td>The interrupt is cleared.</td> </tr> </tbody> </table>	Value	Description	0	The interrupt is unaffected.	1	The interrupt is cleared.
Value	Description									
0	The interrupt is unaffected.									
1	The interrupt is cleared.									
1	CAMCINT	W1C	0	<p>GPTM Capture A Match Interrupt Clear</p> <p>The CAMCINT values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The interrupt is unaffected.</td> </tr> <tr> <td>1</td> <td>The interrupt is cleared.</td> </tr> </tbody> </table>	Value	Description	0	The interrupt is unaffected.	1	The interrupt is cleared.
Value	Description									
0	The interrupt is unaffected.									
1	The interrupt is cleared.									
0	TATOCINT	W1C	0	<p>GPTM Timer A Time-Out Raw Interrupt</p> <p>The TATOCINT values are defined as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The interrupt is unaffected.</td> </tr> <tr> <td>1</td> <td>The interrupt is cleared.</td> </tr> </tbody> </table>	Value	Description	0	The interrupt is unaffected.	1	The interrupt is cleared.
Value	Description									
0	The interrupt is unaffected.									
1	The interrupt is cleared.									

### Register 9: GPTM Timer A 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 Timer B 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 Timer A 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 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:16	TAILRH	R/W	0xFFFF	GPTM Timer A Interval Load Register High  When configured for 32-bit mode via the <b>GPTMCFG</b> register, the <b>GPTM Timer B 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 Timer A Interval Load Register Low  For both 16- and 32-bit modes, writing this field loads the counter for Timer A. A read returns the current value of <b>GPTMTAILR</b> .

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

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

### GPTM Timer B Interval Load (GPTMTBILR)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x02C  
 Type R/W, reset 0x0000.FFFF

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TBILRL	R/W	0xFFFF	GPTM Timer B Interval Load Register

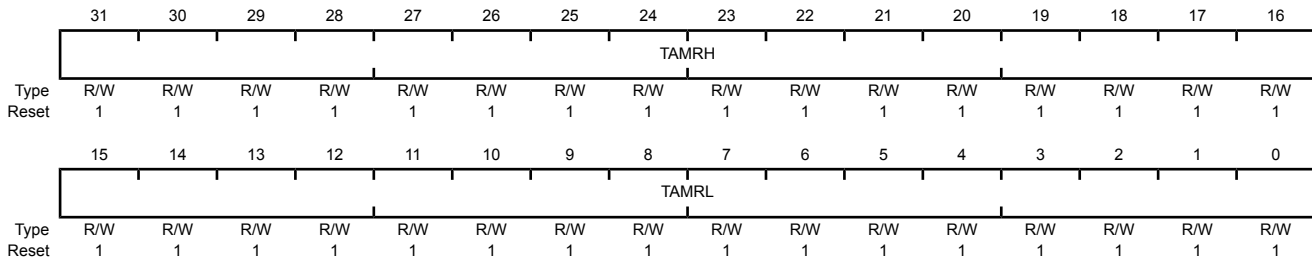
When the GPTM is not configured as a 32-bit timer, a write to this field updates **GPTMTBILR**. In 32-bit mode, writes are ignored, and reads return the current value of **GPTMTBILR**.

### Register 11: GPTM Timer A Match (GPTMTAMATCHR), offset 0x030

This register is loaded with a match value. Interrupts can be generated when the timer value is equal to the value in this register in one-shot or periodic mode. In 16-bit Edge-Count mode, this register along with **GPTMTAILR**, determines how many edge events are counted. The total number of edge events counted is equal to the value in **GPTMTAILR** minus this value.

#### GPTM Timer A Match (GPTMTAMATCHR)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x030  
 Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:16	TAMRH	R/W	0xFFFF	GPTM Timer A Match Register High  When the timer is configured for 32-bit 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 Timer A Match Register Low  When the timer is configured for 32-bit mode via the <b>GPTMCFG</b> register, this value is compared to the lower half of <b>GPTMTAR</b> , to determine match events.  When the timer is configured for 16-bit mode via the <b>GPTMCFG</b> register, this value is compared to <b>GPTMTAR</b> to determine match events.  When configured for 16-bit 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.  When configured for PWM mode, this value along with <b>GPTMTAILR</b> , determines the duty cycle of the output PWM signal.

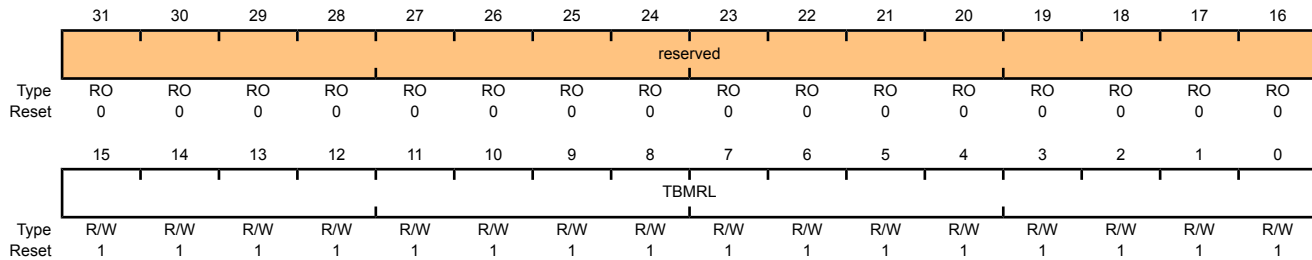


## Register 12: GPTM Timer B Match (GPTMTBMATCHR), offset 0x034

This register is loaded with a match value. Interrupts can be generated when the timer value is equal to the value in this register in one-shot or periodic mode. In 16-bit Edge-Count mode, this register along with **GPTMTAILR**, determines how many edge events are counted. The total number of edge events counted is equal to the value in **GPTMTAILR** minus this value.

### GPTM Timer B Match (GPTMTBMATCHR)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x034  
 Type R/W, reset 0x0000.FFFF



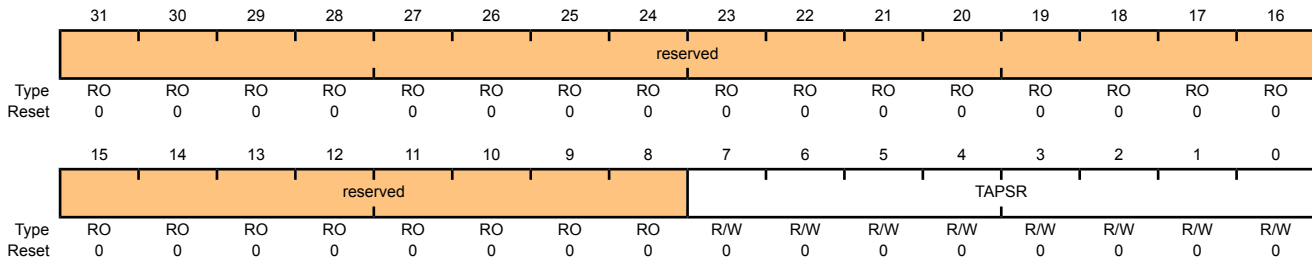
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TBMRL	R/W	0xFFFF	GPTM Timer B Match Register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

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

This register allows software to extend the range of the 16-bit timers.

#### GPTM Timer A Prescale (GPTMTAPR)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x038  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TAPSR	R/W	0x00	GPTM Timer A Prescale  The register loads this value on a write. A read returns the current value of the register.  Refer to Table 11-2 on page 385 for more details and an example.

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

This register allows software to extend the range of the 16-bit timers.

### GPTM Timer B Prescale (GPTMTBPR)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x03C  
 Type R/W, reset 0x0000.0000

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

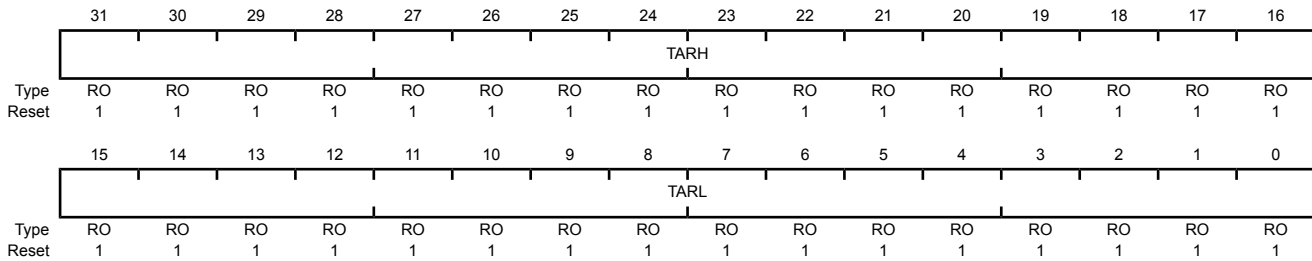
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TBPSR	R/W	0x00	GPTM Timer B Prescale  The register loads this value on a write. A read returns the current value of this register.  Refer to Table 11-2 on page 385 for more details and an example.

### Register 15: GPTM Timer A (GPTMTAR), offset 0x048

This register shows the current value of the Timer A 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 Timer A (GPTMTAR)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x048  
 Type RO, reset 0xFFFF.FFFF



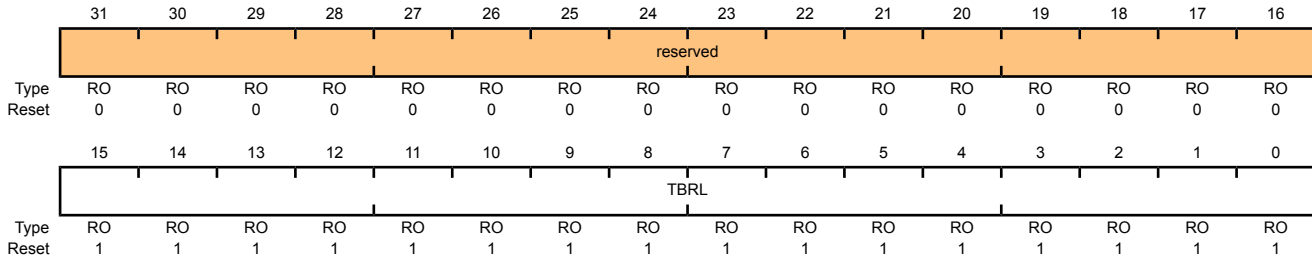
Bit/Field	Name	Type	Reset	Description
31:16	TARH	RO	0xFFFF	GPTM Timer A Register High  If the <b>GPTMCFG</b> is in a 32-bit mode, Timer B value is read. If the <b>GPTMCFG</b> is in a 16-bit mode, this is read as zero.
15:0	TARL	RO	0xFFFF	GPTM Timer A Register Low  A read returns the current value of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

### Register 16: GPTM Timer B (GPTMTBR), offset 0x04C

This register shows the current value of the Timer B 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 Timer B (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 Timer B

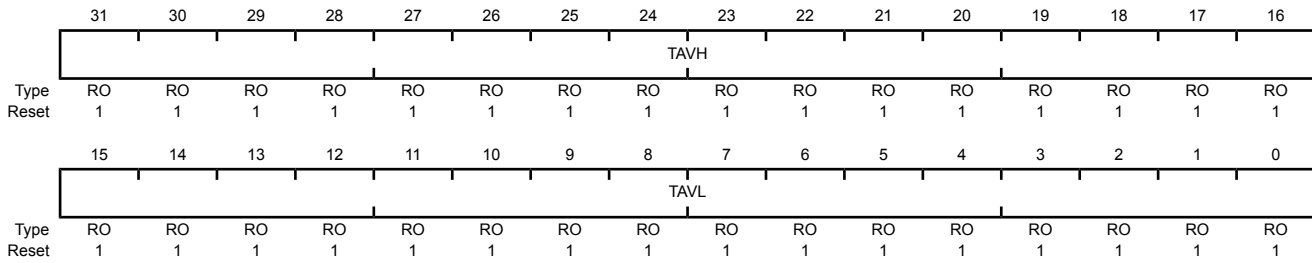
A read returns the current value of the **GPTM Timer B Count Register**, except in Input Edge-Count mode, when it returns the timestamp from the last edge event.

### Register 17: GPTM Timer A Value (GPTMTAV), offset 0x050

This register shows the current, free-running value of Timer A in all modes. Software can use this value to determine the time elapsed between an interrupt and the ISR entry.

#### GPTM Timer A Value (GPTMTAV)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x050  
 Type RO, reset 0xFFFF.FFFF



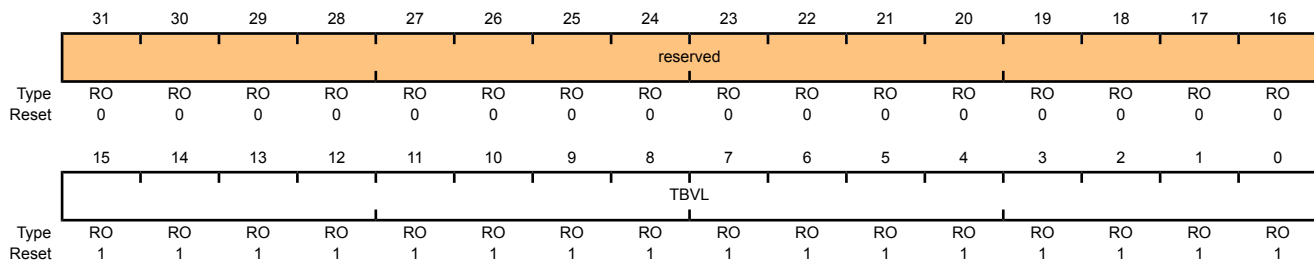
Bit/Field	Name	Type	Reset	Description
31:16	TAVH	RO	0xFFFF	GPTM Timer A Value High  If the <b>GPTMCFG</b> is configured for 32-bit mode, the Timer B value is read. If the <b>GPTMCFG</b> is configured for 16-bit mode, this is read as zero.
15:0	TAVL	RO	0xFFFF	GPTM Timer A Register Low  A read returns the current value of Timer A.

## Register 18: GPTM Timer B Value (GPTMTBV), offset 0x054

This register shows the current, free-running value of Timer B in all modes. Software can use this value to determine the time elapsed between an interrupt and the ISR entry.

### GPTM Timer B Value (GPTMTBV)

Timer0 base: 0x4003.0000  
 Timer1 base: 0x4003.1000  
 Timer2 base: 0x4003.2000  
 Timer3 base: 0x4003.3000  
 Offset 0x054  
 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	TBVL	RO	0xFFFF	GPTM Timer B Register A read returns the current value of Timer B.

## 12 Watchdog Timer

A watchdog timer can generate nonmaskable interrupts (NMIs) or a reset when a time-out value is reached. The watchdog timer is used to regain control when a system has failed due to a software error or due to the failure of an external device to respond in the expected way. The LM3S5791 microcontroller has two Watchdog Timer Modules, one module is clocked by the system clock (Watchdog Timer 0) and the other is clocked by the PIOSC (Watchdog Timer 1). The two modules are identical except that WDT1 is in a different clock domain, and therefore requires synchronizers. As a result, WDT1 has a bit defined in the **Watchdog Timer Control (WDTCTL)** register to indicate when a write to a WDT1 register is complete. Software can use this bit to ensure that the previous access has completed before starting the next access.

The Stellaris<sup>®</sup> Watchdog Timer module has the following features:

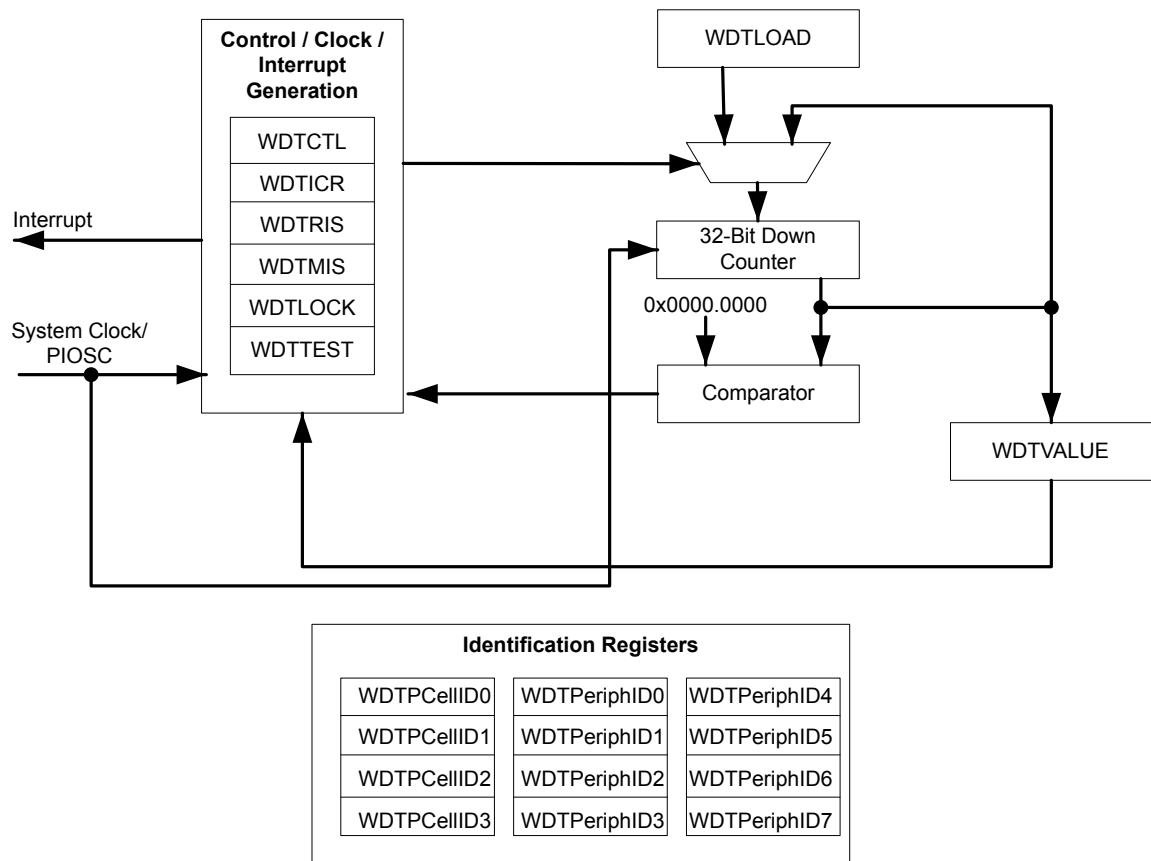
- 32-bit down counter with a programmable load register
- Separate watchdog clock with an enable
- Programmable interrupt generation logic with interrupt masking
- Lock register protection from runaway software
- Reset generation logic with an enable/disable
- User-enabled stalling when the controller asserts the CPU Halt flag during debug

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



## 12.1 Block Diagram

Figure 12-1. WDT Module Block Diagram



## 12.2 Functional Description

The Watchdog Timer module generates the first time-out signal when the 32-bit counter reaches the zero state after being enabled; enabling the counter also enables the watchdog timer interrupt. After the first time-out event, the 32-bit counter is re-loaded with the value of the **Watchdog Timer Load (WDTLOAD)** register, and the timer resumes counting down from that value. Once the Watchdog Timer has been configured, the **Watchdog Timer Lock (WDTLOCK)** register is written, which prevents the timer configuration from being inadvertently altered by software.

If the timer counts down to its zero state again before the first time-out interrupt is cleared, and the reset signal has been enabled by setting the `RESEN` bit in the **WDTCTL** register, the Watchdog timer asserts its reset signal to the system. If the interrupt is cleared before the 32-bit counter reaches its second time-out, the 32-bit counter is loaded with the value in the **WDTLOAD** register, and counting resumes from that value.

If **WDTLOAD** is written with a new value while the Watchdog Timer counter is counting, then the counter is loaded with the new value and continues counting.

Writing to **WDTLOAD** does not clear an active interrupt. An interrupt must be specifically cleared by writing to the **Watchdog Interrupt Clear (WDTICR)** register.

The Watchdog module interrupt and reset generation can be enabled or disabled as required. When the interrupt is re-enabled, the 32-bit counter is preloaded with the load register value and not its last state.

### 12.2.1 Register Access Timing

Because the Watchdog Timer 1 module has an independent clocking domain, its registers must be written with a timing gap between accesses. Software must guarantee that this delay is inserted between back-to-back writes to WDT1 registers or between a write followed by a read to the registers. The timing for back-to-back reads from the WDT1 module has no restrictions. The **WRC** bit in the **Watchdog Control (WDTCTL)** register for WDT1 indicates that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll **WDTCTL** for **WRC=1** prior to accessing another register. Note that WDT0 does not have this restriction as it runs off the system clock.

## 12.3 Initialization and Configuration

To use the WDT, its peripheral clock must be enabled by setting the **WDT** bit in the **RCGC0** register. See page 154.

The Watchdog Timer is configured using the following sequence:

1. Load the **WDTLOAD** register with the desired timer load value.
2. If WDT1, wait for the **WRC** bit in the **WDTCTL** register to be set.
3. If the Watchdog is configured to trigger system resets, set the **RESEN** bit in the **WDTCTL** register.
4. If WDT1, wait for the **WRC** bit in the **WDTCTL** register to be set.
5. Set the **INTEN** bit in the **WDTCTL** register to enable the Watchdog and lock the control register.

If software requires that all of the watchdog registers are locked, the Watchdog Timer module can be fully locked by writing any value to the **WDTLOCK** register. To unlock the Watchdog Timer, write a value of 0x1ACC.E551.

## 12.4 Register Map

Table 12-1 on page 423 lists the Watchdog registers. The offset listed is a hexadecimal increment to the register's address, relative to the Watchdog Timer base address:

- WDT0: 0x4000.0000
- WDT1: 0x4000.1000

Note that the Watchdog Timer module clock must be enabled before the registers can be programmed (see page 154).

**Table 12-1. Watchdog Timer Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	WDTLOAD	R/W	0xFFFF.FFFF	Watchdog Load	424
0x004	WDTVALUE	RO	0xFFFF.FFFF	Watchdog Value	425
0x008	WDTCTL	R/W	0x0000.0000 for WDT0, 0x8000.0000 for WDT1	Watchdog Control	426
0x00C	WDTICR	WO	-	Watchdog Interrupt Clear	428
0x010	WDTRIS	RO	0x0000.0000	Watchdog Raw Interrupt Status	429
0x014	WDTMIS	RO	0x0000.0000	Watchdog Masked Interrupt Status	430
0x418	WDTTEST	R/W	0x0000.0000	Watchdog Test	431
0xC00	WDTLOCK	R/W	0x0000.0000	Watchdog Lock	432
0xFD0	WDTPeriphID4	RO	0x0000.0000	Watchdog Peripheral Identification 4	433
0xFD4	WDTPeriphID5	RO	0x0000.0000	Watchdog Peripheral Identification 5	434
0xFD8	WDTPeriphID6	RO	0x0000.0000	Watchdog Peripheral Identification 6	435
0xFDC	WDTPeriphID7	RO	0x0000.0000	Watchdog Peripheral Identification 7	436
0xFE0	WDTPeriphID0	RO	0x0000.0005	Watchdog Peripheral Identification 0	437
0xFE4	WDTPeriphID1	RO	0x0000.0018	Watchdog Peripheral Identification 1	438
0xFE8	WDTPeriphID2	RO	0x0000.0018	Watchdog Peripheral Identification 2	439
0xFEC	WDTPeriphID3	RO	0x0000.0001	Watchdog Peripheral Identification 3	440
0xFF0	WDTPrimeCellID0	RO	0x0000.000D	Watchdog PrimeCell Identification 0	441
0xFF4	WDTPrimeCellID1	RO	0x0000.00F0	Watchdog PrimeCell Identification 1	442
0xFF8	WDTPrimeCellID2	RO	0x0000.0006	Watchdog PrimeCell Identification 2	443
0xFFC	WDTPrimeCellID3	RO	0x0000.00B1	Watchdog PrimeCell Identification 3	444

## 12.5 Register Descriptions

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

### Register 1: Watchdog Load (WDTLOAD), offset 0x000

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

#### Watchdog Load (WDTLOAD)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0x000

Type R/W, reset 0xFFFF.FFFF

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

Bit/Field	Name	Type	Reset	Description
31:0	WDTLOAD	R/W	0xFFFF.FFFF	Watchdog Load Value

## Register 2: Watchdog Value (WDTVALUE), offset 0x004

This register contains the current count value of the timer.

### Watchdog Value (WDTVALUE)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0x004

Type RO, reset 0xFFFF.FFFF

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

Bit/Field	Name	Type	Reset	Description
31:0	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.

**Important:** Because the Watchdog Timer 1 module has an independent clocking domain, its registers must be written with a timing gap between accesses. Software must guarantee that this delay is inserted between back-to-back writes to WDT1 registers or between a write followed by a read to the registers. The timing for back-to-back reads from the WDT1 module has no restrictions. The `WRC` bit in the **Watchdog Control (WDTCTL)** register for WDT1 indicates that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll **WDTCTL** for `WRC=1` prior to accessing another register. Note that WDT0 does not have this restriction as it runs off the system clock and therefore does not have a `WRC` bit.

#### Watchdog Control (WDTCTL)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0x008

Type R/W, reset 0x0000.0000 for WDT0, 0x8000.0000 for WDT1

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

The `WRC` values are defined as follows:

Value	Description
-------	-------------

0	A write access to one of the WDT1 registers is in progress.
---	---

1	The write access has completed, and WDT1 registers can be read or written.
---	--

**Note:** This bit is reserved for WDT0 and has a reset value of 0.

30:2	reserved	RO	0x000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
------	----------	----	-----------	---

---

Bit/Field	Name	Type	Reset	Description
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.

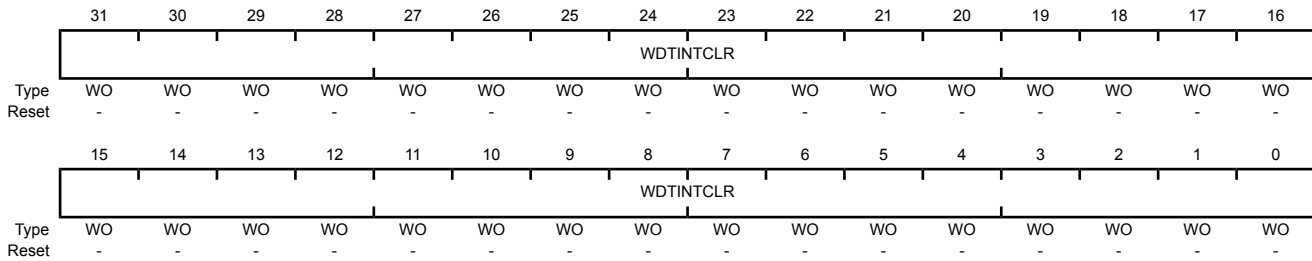
#### Watchdog Interrupt Clear (WDTICR)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

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)

WDT0 base: 0x4000.0000  
 WDT1 base: 0x4000.1000  
 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															WDTRIS
Type	RO	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

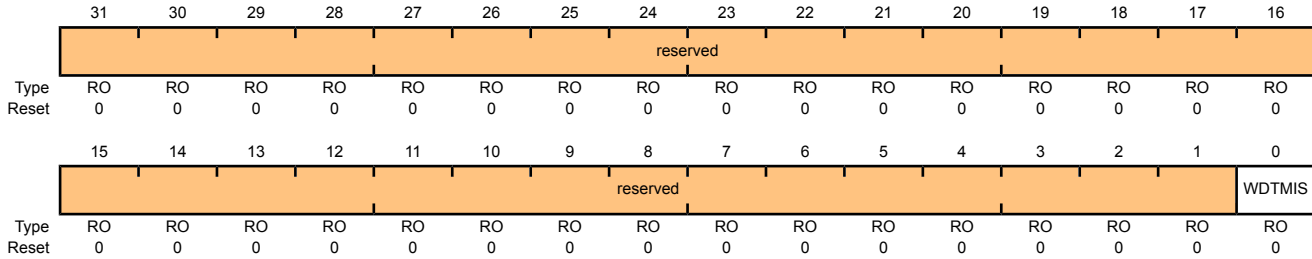
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	WDTRIS	RO	0	Watchdog Raw Interrupt Status Gives the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

### 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)

WDT0 base: 0x4000.0000  
 WDT1 base: 0x4000.1000  
 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 Watchdog 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)**

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0x418

Type R/W, reset 0x0000.0000

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

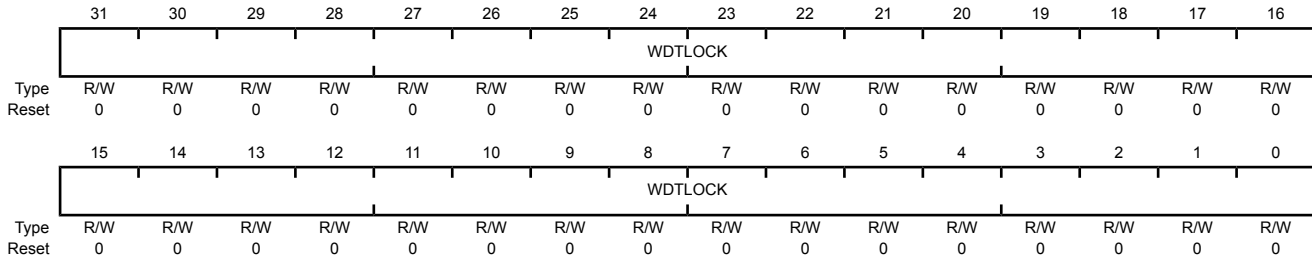
Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	STALL	R/W	0	Watchdog Stall Enable  When set to 1, 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)

WDT0 base: 0x4000.0000  
 WDT1 base: 0x4000.1000  
 Offset 0xC00  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	WDTLOCK	R/W	0x0000	Watchdog Lock

A write of the value 0x1ACC.E551 unlocks the watchdog registers for write access. A write of any other value reapplies the lock, preventing any register updates.

A read of this register returns the following values:

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)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFD0

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

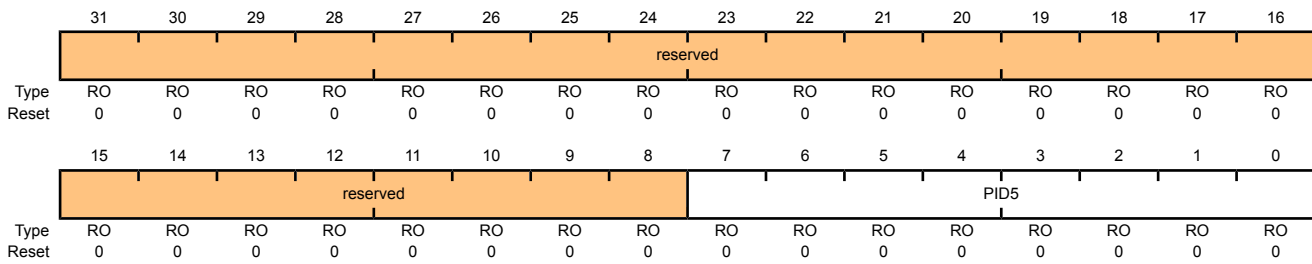
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	WDT Peripheral ID Register [7:0]

## Register 10: Watchdog Peripheral Identification 5 (WDTPeriphID5), offset 0xFD4

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

### Watchdog Peripheral Identification 5 (WDTPeriphID5)

WDT0 base: 0x4000.0000  
 WDT1 base: 0x4000.1000  
 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)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFD8

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID6							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

## Register 12: Watchdog Peripheral Identification 7 (WDTPeriphID7), offset 0xFDC

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

### Watchdog Peripheral Identification 7 (WDTPeriphID7)

WDT0 base: 0x4000.0000  
 WDT1 base: 0x4000.1000  
 Offset 0xFDC  
 Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID7							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	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)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFE0

Type RO, reset 0x0000.0005

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

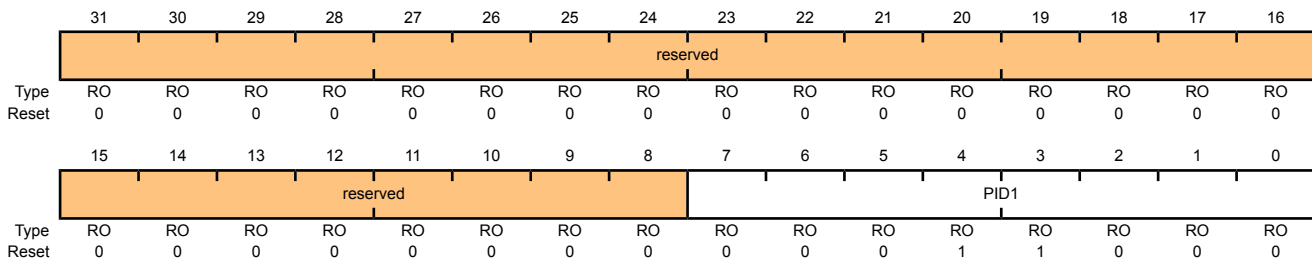
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x05	Watchdog Peripheral ID Register [7:0]

### Register 14: Watchdog Peripheral Identification 1 (WDTPeriphID1), offset 0xFE4

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

#### Watchdog Peripheral Identification 1 (WDTPeriphID1)

WDT0 base: 0x4000.0000  
 WDT1 base: 0x4000.1000  
 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)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFE8

Type RO, reset 0x0000.0018

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

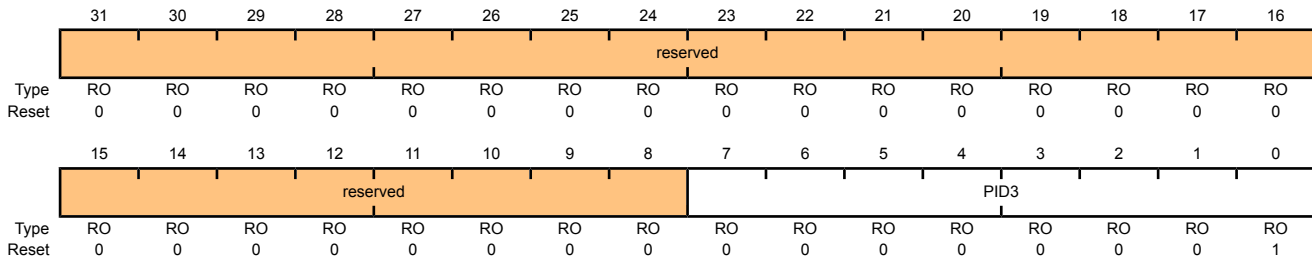
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	Watchdog Peripheral ID Register [23:16]

### Register 16: Watchdog Peripheral Identification 3 (WDTPeriphID3), offset 0xFEC

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

#### Watchdog Peripheral Identification 3 (WDTPeriphID3)

WDT0 base: 0x4000.0000  
 WDT1 base: 0x4000.1000  
 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)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFF0

Type RO, reset 0x0000.000D

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved								CID0								
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

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

### Register 18: Watchdog PrimeCell Identification 1 (WDTPCellID1), offset 0xFF4

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

#### Watchdog PrimeCell Identification 1 (WDTPCellID1)

WDT0 base: 0x4000.0000  
 WDT1 base: 0x4000.1000  
 Offset 0xFF4  
 Type RO, reset 0x0000.00F0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

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

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFF8

Type RO, reset 0x0000.0006

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

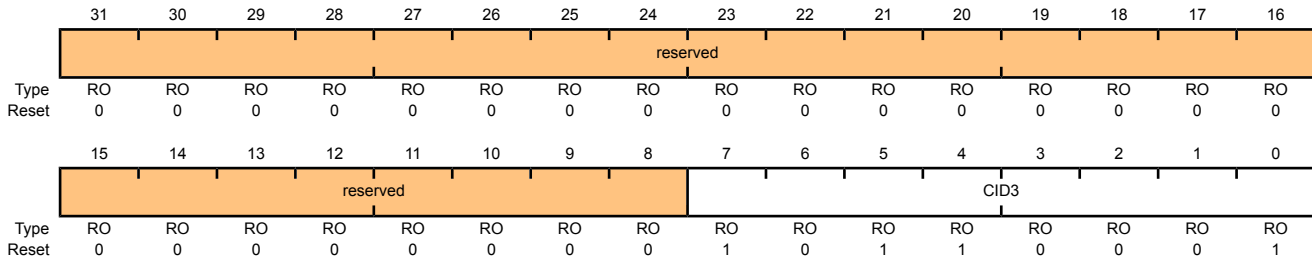
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x06	Watchdog PrimeCell ID Register [23:16]

### 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)

WDT0 base: 0x4000.0000  
 WDT1 base: 0x4000.1000  
 Offset 0xFFC  
 Type RO, reset 0x0000.00B1



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	Watchdog PrimeCell ID Register [31:24]



## 13 Analog-to-Digital Converter (ADC)

An analog-to-digital converter (ADC) is a peripheral that converts a continuous analog voltage to a discrete digital number. Two identical converter units are included, which share sixteen input channels. The two converter units may be sampled in the same processor clock or out of phase with each other.

The Stellaris<sup>®</sup> ADC module features 10-bit conversion resolution and supports sixteen input channels, plus an internal temperature sensor. The ADC module contains four programmable sequencers allowing the sampling of multiple analog input sources without controller intervention. Each sample sequencer provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequencer priority. A digital comparator function is included which allows the conversion value to be diverted to a digital comparator module. The digital comparator module provides digital comparator. The comparator module measures the ADC conversion value against two user-defined values to determine the operational range of the signal.

The Stellaris<sup>®</sup> ADC module provides the following features:

- Sixteen analog input channels
- Single-ended and differential-input configurations
- On-chip internal temperature sensor
- Sample rate of one million samples/second
- Flexible, configurable analog-to-digital conversion
- Four programmable sample conversion sequencers 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
- Digital comparison unit providing digital comparator
- Converter uses an internal 3-V reference or an external reference
- Power and ground for the analog circuitry is separate from the digital power and ground
- Efficient transfers using Micro Direct Memory Access Controller ( $\mu$ DMA)
  - Dedicated channel for each sample sequencer
  - Burst request asserted when interrupt is triggered

### 13.1 Block Diagram

The Stellaris® microcontroller contains two identical Analog-to-Digital Converter units. These two modules, ADC0 and ADC1, share the same sixteen analog input channels. Each ADC module operates independently and can therefore execute different sample sequences, sample any of the analog input channels at any time, and generate different interrupts and triggers. Figure 13-1 on page 446 shows how the two modules are connected to analog inputs and the system bus.

Figure 13-1. Implementation of Two ADC Blocks

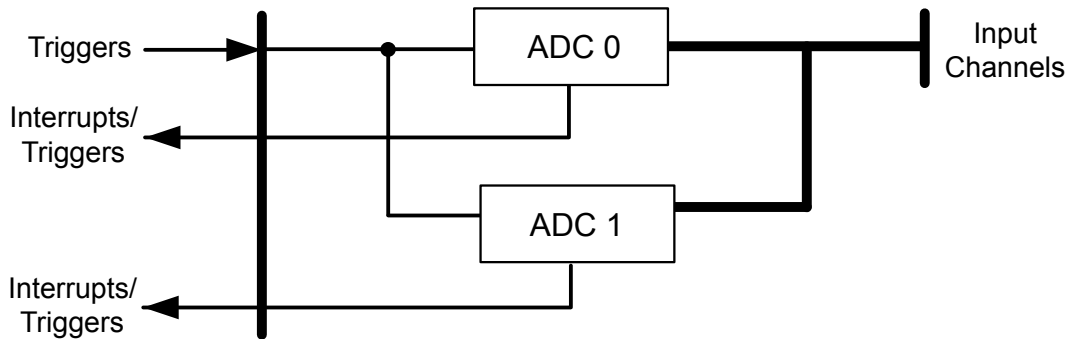
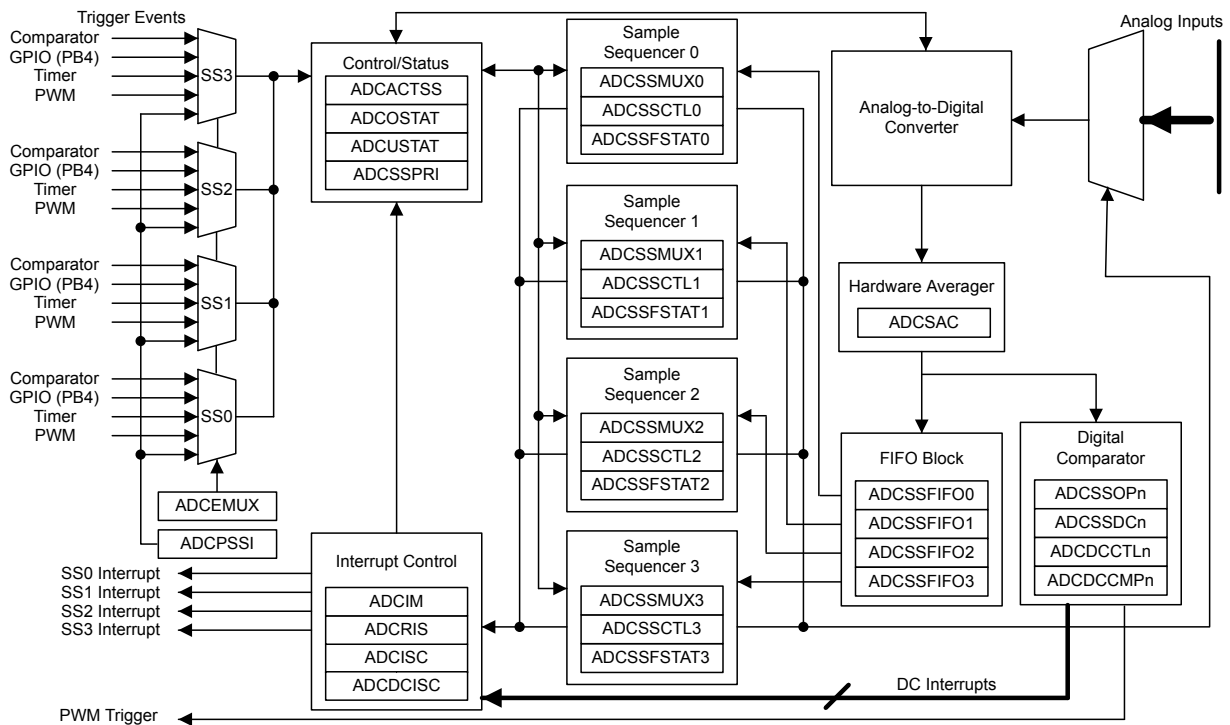


Figure 13-2 on page 446 provides details on the internal configuration of the ADC controls and data registers.

Figure 13-2. ADC Module Block Diagram



## 13.2 Functional Description

The Stellaris<sup>®</sup> ADC collects sample data by using a programmable sequence-based approach instead of the traditional single or double-sampling approaches found on many ADC modules. Each *sample sequence* is a fully programmed series of consecutive (back-to-back) samples, allowing the ADC to collect data from multiple input sources without having to be re-configured or serviced by the processor. 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. The  $\mu$ DMA can be used to more efficiently move data from the sample sequencers without CPU intervention.

### 13.2.1 Sample Sequencers

The sampling control and data capture is handled by the sample sequencers. All of the sequencers are identical in implementation except for the number of samples that can be captured and the depth of the FIFO. Table 13-1 on page 447 shows the maximum number of samples that each sequencer can capture and its corresponding FIFO depth. In this implementation, each FIFO entry is a 32-bit word, with the lower 10 bits containing the conversion result.

**Table 13-1. Samples and FIFO Depth of Sequencers**

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

For a given sample sequence, each sample is defined by two 4-bit nibbles in the **ADC Sample Sequence Input Multiplexer Select (ADCSSMUXn)** and **ADC Sample Sequence Control (ADCSSCTLn)** registers, where "n" corresponds to the sequence number. The **ADCSSMUXn** nibbles select the input pin, while the **ADCSSCTLn** nibbles contain the sample control bits corresponding to parameters such as temperature sensor selection, interrupt enable, end of sequence, and differential input mode. Sample sequencers are enabled by setting the respective **ASENn** bit in the **ADC Active Sample Sequencer (ADCACTSS)** register and should be configured before being enabled. Sampling is then initiated by setting the **SSn** bit in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register. In addition, sample sequences may be initiated on multiple ADC modules simultaneously using the **GSYNC** and **SYNCWAIT** bits in the **ADCPSSI** register during the configuration of each ADC module. For more information on using these bits, refer to page 477.

When configuring a sample sequence, multiple uses of the same input pin within the same sequence is allowed. In the **ADCSSCTLn** register, the **IE<sub>n</sub>** 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 (ADCSSFSTAT<sub>n</sub>)** registers along with **FULL** and **EMPTY** status flags. Overflow and underflow conditions are monitored using the **ADCOSTAT** and **ADCUSTAT** registers.

## 13.2.2 Module Control

Outside of the sample sequencers, the remainder of the control logic is responsible for tasks such as:

- Interrupt generation
- Sequence prioritization
- Trigger configuration
- Comparator 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.

### 13.2.2.1 Interrupts

The register configurations of the sample sequencers and digital comparators dictate which events generate raw interrupts, but do not have control over whether the interrupt is actually sent to the interrupt controller. The ADC module's interrupt signals are controlled by the state of the `MASK` bits in the **ADC Interrupt Mask (ADCIM)** register. Interrupt status can be viewed at two locations: the **ADC Raw Interrupt Status (ADCRIS)** register, which shows the raw status of the various interrupt signals; and the **ADC Interrupt Status and Clear (ADCISC)** register, which shows active interrupts that are enabled by the **ADCIM** register. Sequencer interrupts are cleared by writing a 1 to the corresponding `IN` bit in **ADCISC**. Digital comparator interrupts are cleared by writing a 1 to the **ADC Digital Comparator Interrupt Status and Clear (ADCDCISC)** register.

### 13.2.2.2 DMA Operation

The ADC module provides a request signal to the  $\mu$ DMA controller for each sample sequencer. Each sample sequencer has a dedicated  $\mu$ DMA channel. The request signal is a burst type and is asserted whenever an interrupt is enabled in a sample sequence (`IE` bit in the **ADCSSCTLn** register is set). Single requests are not supported.

The arbitration size of the  $\mu$ DMA transfer must be a power of 2, and the associated `IE` bits in the **ADDSSCTLn** register must be set. For example, if the  $\mu$ DMA channel of SS0 has an arbitration size of four, the `IE3` bit (4th sample) and the `IE7` bit (8th sample) must be set. Thus the  $\mu$ DMA request occurs every time 4 samples have been acquired. No other special steps are needed to enable the ADC module for  $\mu$ DMA operation.

Refer to the “Micro Direct Memory Access ( $\mu$ DMA)” on page 219 for more details about programming the  $\mu$ DMA controller.

### 13.2.2.3 Prioritization

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

### 13.2.2.4 Sampling Events

Sample triggering for each sample sequencer is defined in the **ADC Event Multiplexer Select (ADCEMUX)** register. Trigger sources include processor (default), analog comparators, an external signal on GPIO `PB4`, a GP Timer, PWM2, and continuous sampling.

Care must be taken when using the continuous sampling trigger. If a sequencer's priority is too high, it is possible to starve other lower priority sequencers.

### 13.2.2.5 External Voltage Reference

An external reference voltage may be provided to serve as the maximum conversion value reference. The  $V_{REF}$  bit in the **ADC Control (ADCCTL)** register specifies whether to use the internal or external reference. The useful range for the external voltage reference is  $2.4\text{ V} - V_{DDA}$ . Ground is always used as the reference level for the minimum conversion value. Care must be taken to supply a reference voltage of acceptable quality.

### 13.2.3 Hardware Sample Averaging Circuit

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

By default the averaging circuit is off, and all data from the converter passes through to the sequencer FIFO. The averaging hardware is controlled by the **ADC Sample Averaging Control (ADCSAC)** register (see page 479). A single averaging circuit has been implemented, thus all input channels receive the same amount of averaging whether they are single-ended or differential.

### 13.2.4 Analog-to-Digital Converter

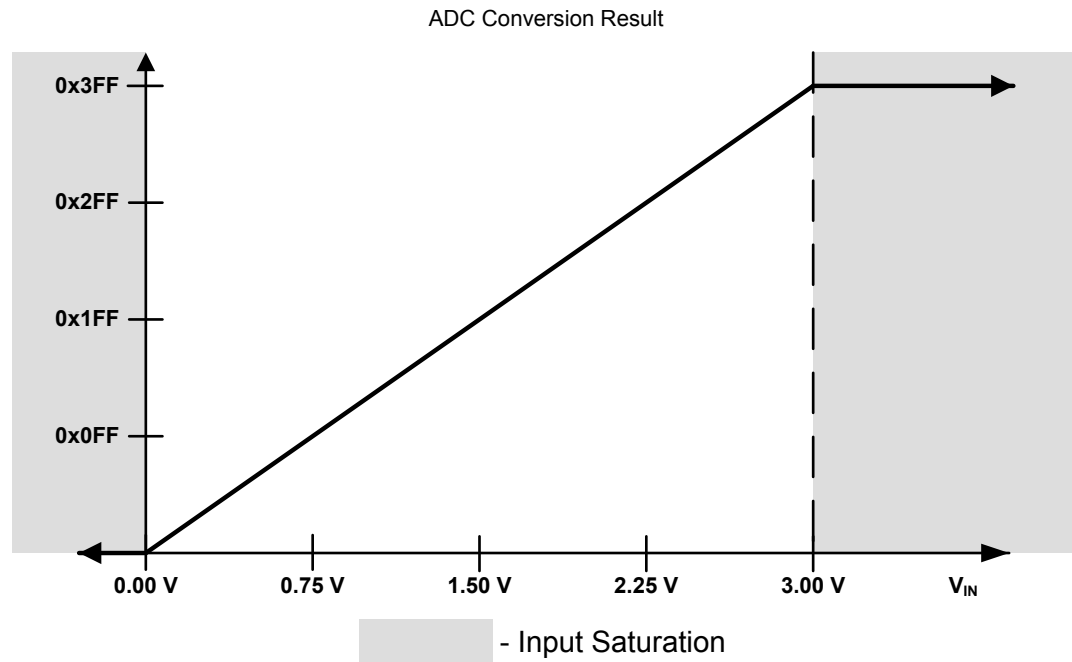
The Analog-to-Digital Converter (ADC) module uses a Successive Approximation Register (SAR) architecture to deliver a 10-bit, low-power, high-precision conversion value. The successive-approximation algorithm uses a current mode D/A converter to achieve lower settling time, resulting in higher conversion speeds for the A/D converter. In addition, built-in sample-and-hold circuitry with offset-calibration circuitry improves conversion accuracy. The sample-and-hold circuitry is open for the first time period of a conversion as specified by  $T_{ADCCA}$  and requires that the ADC be run from the PLL or a 16-MHz clock source.

The ADC operates from the 3.3-V analog and 1.2-V digital power supply. Integrated shutdown modes are available to reduce power consumption when ADC conversions are not required. The analog inputs are connected to the ADC through custom pads and specially balanced input paths to minimize the distortion on the inputs. Detailed information on the ADC power supplies and analog inputs can be found in "Analog-to-Digital Converter" on page 960.

#### 13.2.4.1 Internal Voltage Reference

The band-gap circuitry generates an internal 3.0 V reference that can be used by the ADC to produce a conversion value from the selected analog input. The range of this conversion value is from 0x000 to 0x3FF. In single-ended-input mode, the 0x000 value corresponds to an analog input voltage of 0.0 V; the 0x3FF value corresponds to an analog input voltage of 3.0 V. This configuration results in a resolution of approximately 2.9 mV per ADC code. While the analog input pads can handle voltages beyond this range, the ADC conversions saturate in under-voltage and over-voltage cases. Figure 13-3 on page 450 shows the ADC conversion function of the analog inputs.

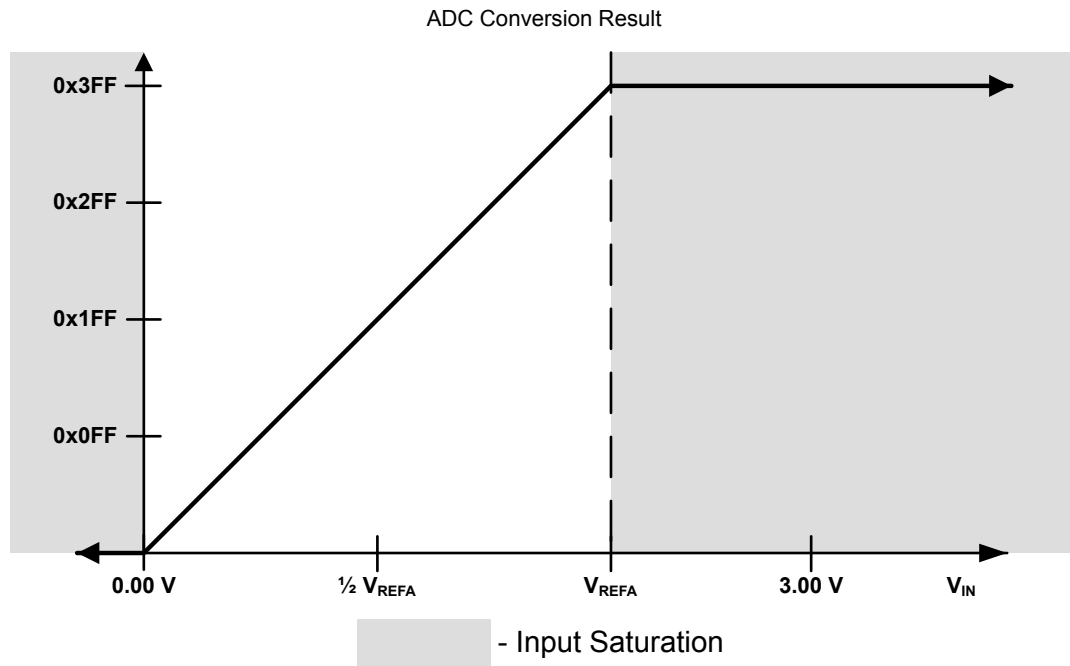
Figure 13-3. Internal Voltage Conversion Result



#### 13.2.4.2 External Voltage Reference

The ADC can use an external voltage reference to produce the conversion value from the selected analog input by setting the  $V_{REF}$  bit in the **ADC Control (ADCCTL)** register. While the range of the conversion value remains the same (0x000 to 0x3FF), the analog voltage associated with the 0x3FF value corresponds to the value of the external voltage reference, resulting in a smaller voltage resolution per ADC code. Analog input voltages above the external voltage reference saturate to 0x3FF while those below 0.0 V continue to saturate at 0x000. Figure 13-4 on page 451 shows the ADC conversion function of the analog inputs when using an external voltage reference.

Figure 13-4. External Voltage Conversion Result



The external voltage reference can be useful in circuits where the maximum analog input voltage is significantly lower than 3.0 V (3.3 V). In this case, the maximum value of the analog input can be used as the external voltage reference. The result is conversions covering the entire 10-bit range and a smaller voltage resolution per bit. There is a physical limit to how far the external voltage reference can be lowered before the quantization resolution of the ADC is exceeded and accuracy starts to be affected. “Analog-to-Digital Converter” on page 960 provides detailed information on the tradeoffs between voltage reference and conversion accuracy.

### 13.2.5 Differential Sampling

In addition to traditional single-ended sampling, the ADC module supports differential sampling of two analog input channels. To enable differential sampling, software must set the  $D_n$  bit in the **ADCSSCTL0n** register in a step's configuration nibble.

When a sequence step is configured for differential sampling, the input pair to sample must be configured in the **ADCSSMUXn** register. Differential pair 0 samples analog inputs 0 and 1; differential pair 1 samples analog inputs 2 and 3; and so on (see Table 13-2 on page 451). The ADC does not support other differential pairings such as analog input 0 with analog input 3.

Table 13-2. Differential Sampling Pairs

Differential Pair	Analog Inputs
0	0 and 1
1	2 and 3
2	4 and 5
3	6 and 7
4	8 and 9

Differential Pair	Analog Inputs
5	10 and 11
6	12 and 13
7	14 and 15

The voltage sampled in differential mode is the difference between the odd and even channels:

$\Delta V$  (differential voltage) =  $V_{IN\_EVEN}$  (even channels) –  $V_{IN\_ODD}$  (odd channels), therefore:

- If  $\Delta V = 0$ , then the conversion result = 0x1FF
- If  $\Delta V > 0$ , then the conversion result > 0x1FF (range is 0x1FF–0x3FF)
- If  $\Delta V < 0$ , then the conversion result < 0x1FF (range is 0–0x1FF)

The differential pairs assign polarities to the analog inputs: the even-numbered input is always positive, and the odd-numbered input is always negative. In order for a valid conversion result to appear, the negative input must be in the range of  $\pm 1.5$  V of the positive input. If an analog input is greater than 3 V or less than 0 V (the valid range for analog inputs), the input voltage is clipped, meaning it appears as either 3 V or 0 V, respectively, to the ADC.

Figure 13-5 on page 452 shows an example of the negative input centered at 1.5 V. In this configuration, the differential range spans from -1.5 V to 1.5 V. Figure 13-6 on page 453 shows an example where the negative input is centered at -0.75 V, meaning inputs on the positive input saturate past a differential voltage of -0.75 V since the input voltage is less than 0 V. Figure 13-7 on page 453 shows an example of the negative input centered at 2.25 V, where inputs on the positive channel saturate past a differential voltage of 0.75 V since the input voltage would be greater than 3 V.

**Figure 13-5. Differential Sampling Range,  $V_{IN\_ODD} = 1.5$  V**

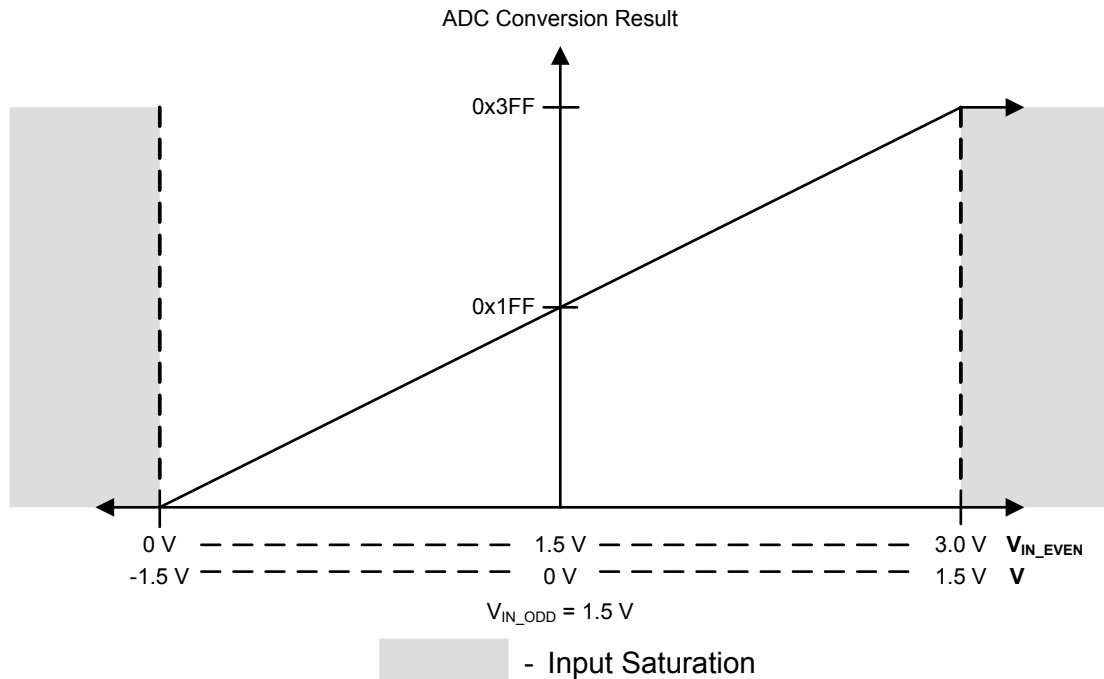
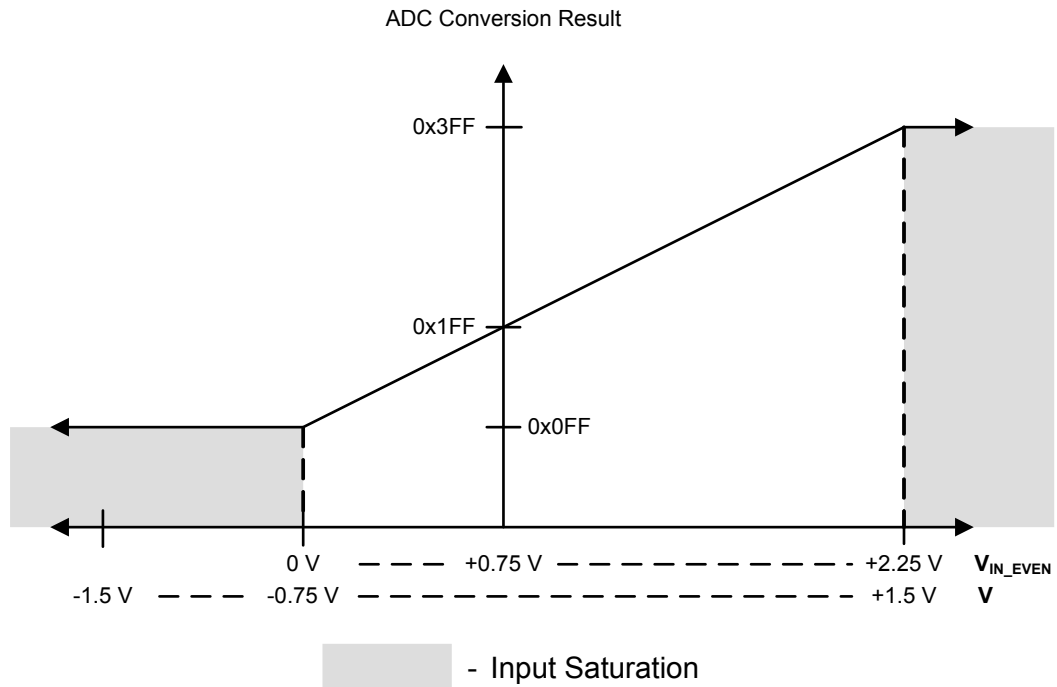
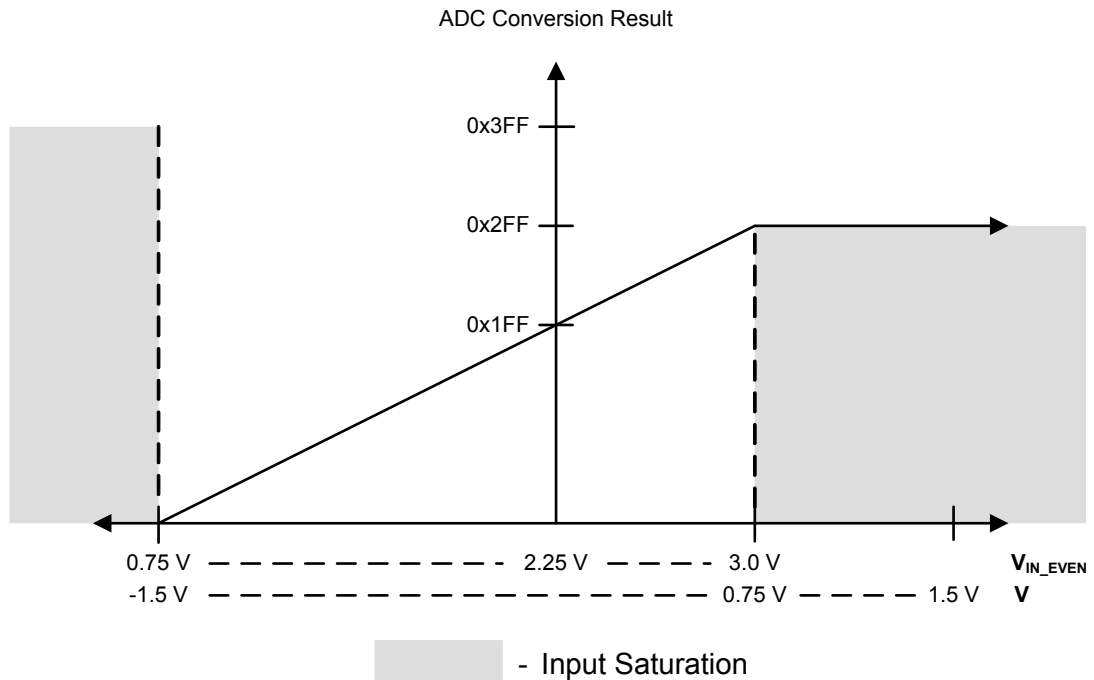




Figure 13-6. Differential Sampling Range,  $V_{IN\_ODD} = 0.75\text{ V}$ Figure 13-7. Differential Sampling Range,  $V_{IN\_ODD} = 2.25\text{ V}$ 

### 13.2.6 Internal Temperature Sensor

The temperature sensor's primary purpose is to notify the system that the internal temperature is too high or low for reliable operation.

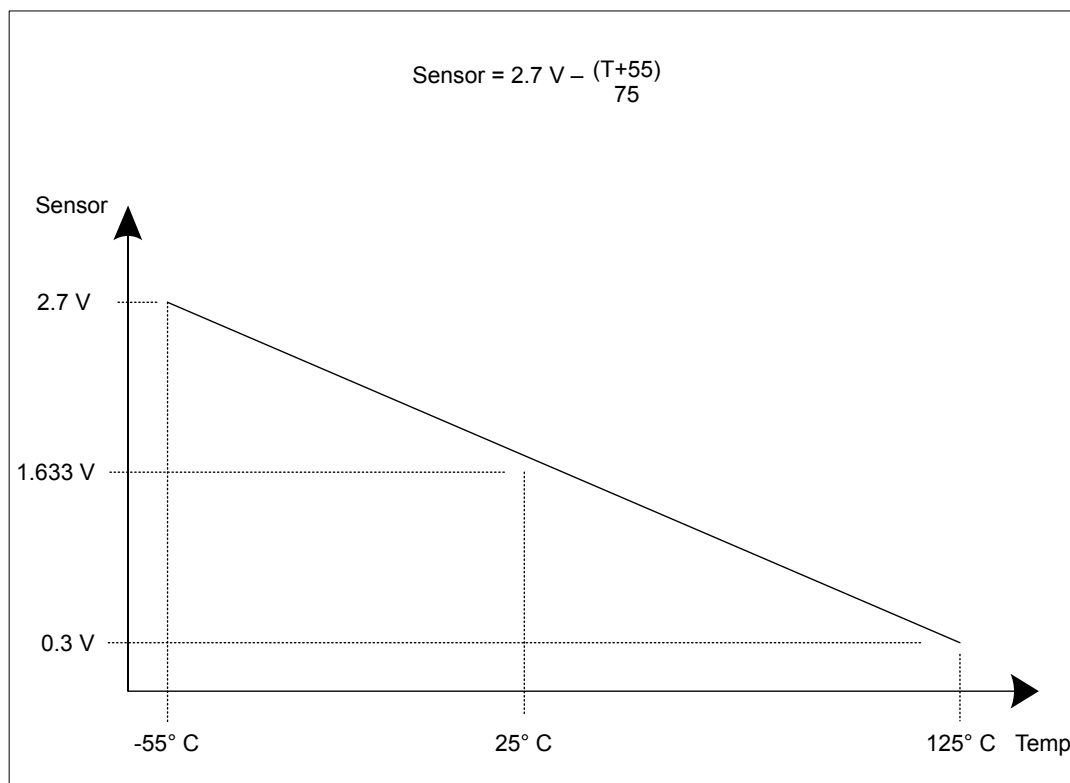
The temperature sensor does not have a separate enable, because it also contains the bandgap reference and must always be enabled. The reference is supplied to other analog modules; not just the ADC.

The internal temperature sensor provides an analog temperature reading as well as a reference voltage. The voltage at the output terminal SENSO is given by the following equation:

$$SENSO = 2.7 - ((T + 55) / 75)$$

This relation is shown in Figure 13-8 on page 454.

**Figure 13-8. Internal Temperature Sensor Characteristic**



### 13.2.7 Digital Comparator Unit

An ADC is commonly used to sample an external signal and to monitor its value to ensure that it remains in a given range. To automate this monitoring procedure and reduce the amount of processor overhead that is required, atwothreefourfivesixseveneight dual-level value digital comparators are provided. Conversions from the ADC that are sent to the digital comparators are compared against the user programmable limits in the **ADC Digital Comparator Range (ADCDCMPn)** registers. If the observed signal moves out of the acceptable range, a processor interrupt can be generated and/or a trigger can be sent to the PWM module. The digital comparators four operational modes (Once, Always, Hysteresis Once, Hysteresis Always) can be applied to three separate regions (low band, mid band, high band) as defined by the user.

### 13.2.7.1 Output Functions

ADC conversions can either be stored in the ADC Sample Sequence FIFOs or compared using the digital comparator resources as defined by the  $S_nDCOP$  bits in the **ADC Sample Sequence n Operation (ADCSSOPn)** register. These selected ADC conversions are used by their respective digital comparator to monitor the external signal. Each comparator has two possible output functions: processor interrupts and triggers.

Each function has its own state machine to track the monitored signal. Even though the interrupt and trigger functions can be enabled individually or both at the same time, the same conversion data is used by each function to determine if the right conditions have been met to assert the associated output.

#### *Interrupts*

The digital comparator interrupt function is enabled by setting the  $CIE$  bit in the **ADC Digital Comparator Control (ADCDCCTLn)** register. This bit enables the interrupt function state machine to start monitoring the incoming ADC conversions. When the appropriate set of conditions is met, and the  $DCONSS_x$  bit is set in the **ADCIM** register, an interrupt is sent to the interrupt controller.

#### *Triggers*

The digital comparator trigger function is enabled by setting the  $CTE$  bit in the **ADCDCCTLn** register. This bit enables the trigger function state machine to start monitoring the incoming ADC conversions. When the appropriate set of conditions is met, the corresponding digital comparator trigger to the PWM module is asserted.

### 13.2.7.2 Operational Modes

Four operational modes are provided to support a broad range of applications and multiple possible signaling requirements: Always, Once, Hysteresis Always, and Hysteresis Once. The operational mode is selected using the  $CIM$  or  $CTM$  field in the **ADCDCCTLn** register.

#### *Always Mode*

In the Always operational mode, the associated interrupt or trigger is asserted whenever the ADC conversion value meets its comparison criteria. The result is a string of assertions on the interrupt or trigger while the conversions are within the appropriate range.

#### *Once Mode*

In the Once operational mode, the associated interrupt or trigger is asserted whenever the ADC conversion value meets its comparison criteria, and the previous ADC conversion value did not. The result is a single assertion of the interrupt or trigger when the conversions are within the appropriate range.

#### *Hysteresis-Always Mode*

The Hysteresis-Always operational mode can only be used in conjunction with the low-band or high-band regions because the mid-band region must be crossed and the opposite region entered to clear the hysteresis condition. In the Hysteresis-Always mode, the associated interrupt or trigger is asserted in the following cases: 1) the ADC conversion value meets its comparison criteria or 2) a previous ADC conversion value has met the comparison criteria, and the hysteresis condition has not been cleared by entering the opposite region. The result is a string of assertions on the interrupt or trigger that continue until the opposite region is entered.

**Hysteresis-Once Mode**

The Hysteresis-Once operational mode can only be used in conjunction with the low-band or high-band regions because the mid-band region must be crossed and the opposite region entered to clear the hysteresis condition. In the Hysteresis-Once mode, the associated interrupt or trigger is asserted only when the ADC conversion value meets its comparison criteria, the hysteresis condition is clear, and the previous ADC conversion did not meet the comparison criteria. The result is a single assertion on the interrupt or trigger.

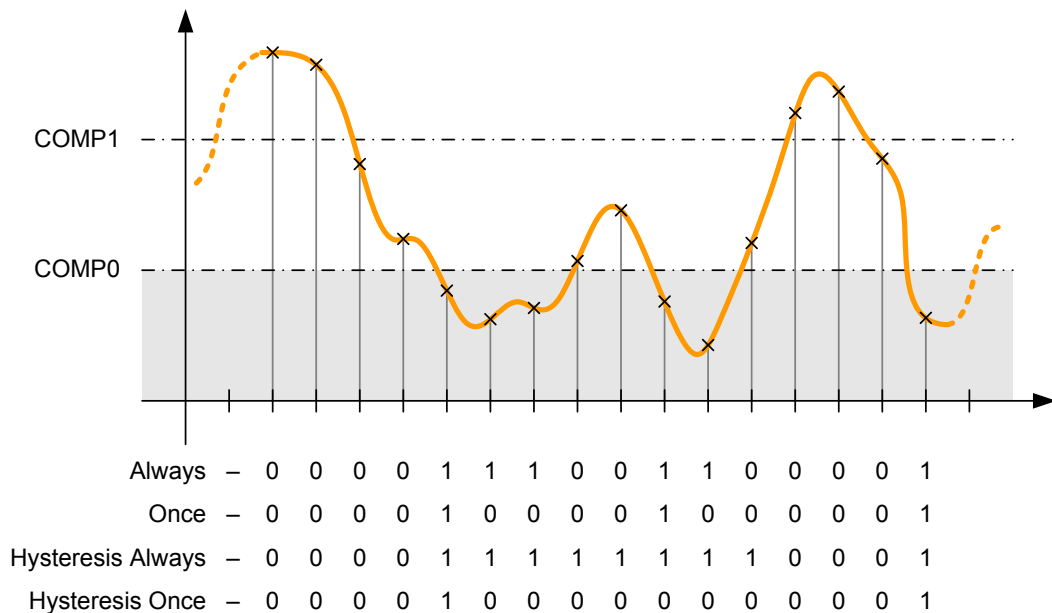
**13.2.7.3 Function Ranges**

The two comparison values, COMP0 and COMP1, in the **ADC Digital Comparator Range (ADCDCMPn)** register effectively break the conversion area into three distinct regions. These regions are referred to as the low-band (less than or equal to COMP0), mid-band (greater than COMP0 but less than or equal to COMP1), and high-band (greater than COMP1) regions. COMP0 and COMP1 may be programmed to the same value, effectively creating two regions, but COMP1 must always be greater than or equal to the value of COMP0. A COMP1 value that is less than COMP0 generates unpredictable results.

**Low-Band Operation**

To operate in the low-band region, either the CIC field or the CTC field in the **ADCDCCTLn** register must be programmed to 0x0. This setting causes interrupts or triggers to be generated in the low-band region as defined by the programmed operational mode. An example of the state of the interrupt/trigger signal in the low-band region for each of the operational modes is shown in Figure 13-9 on page 456. Note that a "0" in a column following the operational mode name (Always, Once, Hysteresis Always, and Hysteresis Once) indicates that the interrupt or trigger signal is de-asserted and a "1" indicates that the signal is asserted.

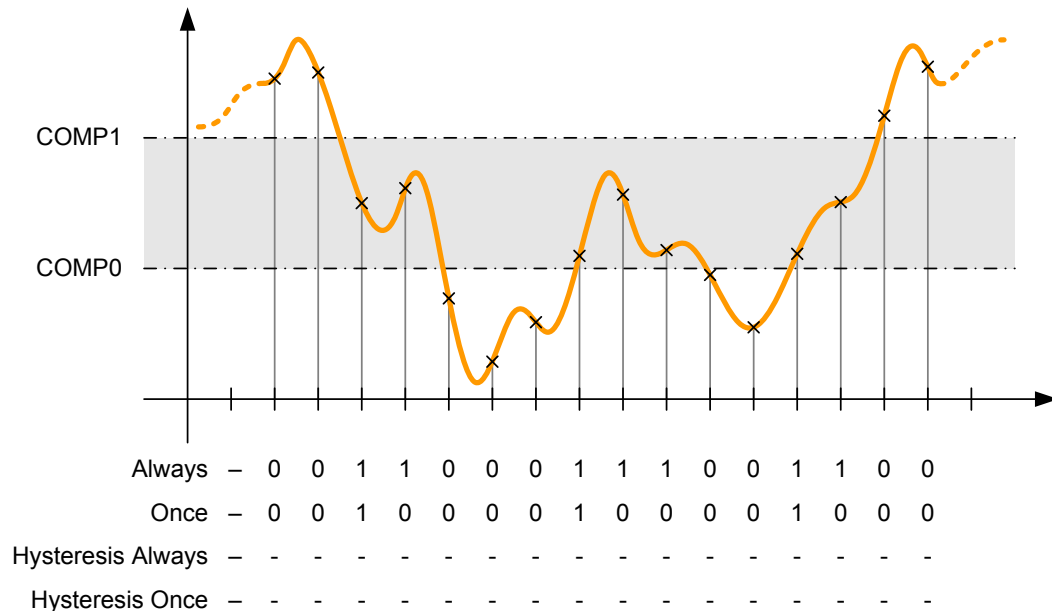
**Figure 13-9. Low-Band Operation (CIC=0x0 and/or CTC=0x0)**



### Mid-Band Operation

To operate in the mid-band region, either the `CIC` field or the `CTC` field in the `ADCDCCTLn` register must be programmed to `0x1`. This setting causes interrupts or triggers to be generated in the mid-band region according to the operation mode. Only the Always and Once operational modes are available in the mid-band region. An example of the state of the interrupt/trigger signal in the mid-band region for each of the allowed operational modes is shown in Figure 13-10 on page 457. Note that a "0" in a column following the operational mode name (Always or Once) indicates that the interrupt or trigger signal is de-asserted and a "1" indicates that the signal is asserted.

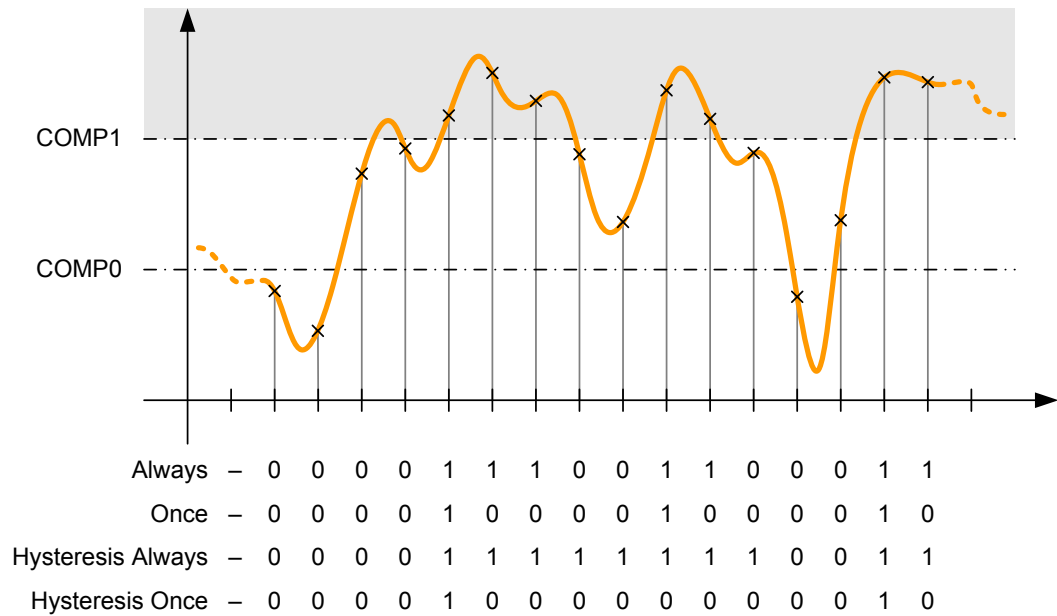
**Figure 13-10. Mid-Band Operation (CIC=0x1 and/or CTC=0x1)**



### High-Band Operation

To operate in the high-band region, either the `CIC` field or the `CTC` field in the `ADCDCCTLn` register must be programmed to `0x3`. This setting causes interrupts or triggers to be generated in the high-band region according to the operation mode. An example of the state of the interrupt/trigger signal in the high-band region for each of the allowed operational modes is shown in Figure 13-11 on page 458. Note that a "0" in a column following the operational mode name (Always, Once, Hysteresis Always, and Hysteresis Once) indicates that the interrupt or trigger signal is de-asserted and a "1" indicates that the signal is asserted.

Figure 13-11. High-Band Operation (CIC=0x3 and/or CTC=0x3)



## 13.3 Initialization and Configuration

In order for the ADC module to be used, the PLL must be enabled and programmed to a supported crystal frequency in the **RCC** register (see page 109). Using unsupported frequencies can cause faulty operation in the ADC module.

### 13.3.1 Module Initialization

Initialization of the ADC module is a simple process with very few steps: enabling the clock to the ADC, disabling the analog isolation circuit associated with all inputs that are to be used, and reconfiguring the sample sequencer priorities (if needed).

The initialization sequence for the ADC is as follows:

1. Enable the ADC clock by writing a value of 0x0001.0000 to the **RCGC0** register (see page 154).
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register (see page 175). To find out which GPIO port to enable, refer to Table 24-5 on page 937.
3. Disable the analog isolation circuit for all ADC input pins that are to be used by writing a 1 to the appropriate bits of the **GPIOAMSEL** register (see page 319) in the associated GPIO block.
4. If required by the application, reconfigure the sample sequencer priorities in the **ADCSSPRI** register. The default configuration has Sample Sequencer 0 with the highest priority and Sample Sequencer 3 as the lowest priority.

### 13.3.2 Sample Sequencer Configuration

Configuration of the sample sequencers is slightly more complex than the module initialization because each sample sequencer is completely programmable.

The configuration for each sample sequencer should be as follows:

1. Ensure that the sample sequencer is disabled by clearing the corresponding `ASENn` bit in the **ADCACTSS** register. Programming of the sample sequencers is allowed without having them enabled. Disabling the sequencer during programming prevents erroneous execution if a trigger event were to occur during the configuration process.
2. Configure the trigger event for the sample sequencer in the **ADCEMUX** register.
3. For each sample in the sample sequence, configure the corresponding input source in the **ADCSSMUX<sub>n</sub>** register.
4. For each sample in the sample sequence, configure the sample control bits in the corresponding nibble in the **ADCSSCTL<sub>n</sub>** 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, set the corresponding `MASK` bit in the **ADCIM** register.
6. Enable the sample sequencer logic by setting the corresponding `ASENn` bit in the **ADCACTSS** register.

## 13.4 Register Map

Table 13-3 on page 459 lists the ADC registers. The offset listed is a hexadecimal increment to the register's address, relative to that ADC module's base address of:

- ADC0: 0x4003.8000
- ADC1: 0x4003.9000

Note that the ADC module clock must be enabled before the registers can be programmed (see page 154).

**Table 13-3. ADC Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	ADCACTSS	R/W	0x0000.0000	ADC Active Sample Sequencer	462
0x004	ADCRIS	RO	0x0000.0000	ADC Raw Interrupt Status	463
0x008	ADCIM	R/W	0x0000.0000	ADC Interrupt Mask	465
0x00C	ADCISC	R/W1C	0x0000.0000	ADC Interrupt Status and Clear	467
0x010	ADCOSTAT	R/W1C	0x0000.0000	ADC Overflow Status	469
0x014	ADCEMUX	R/W	0x0000.0000	ADC Event Multiplexer Select	470
0x018	ADCUSTAT	R/W1C	0x0000.0000	ADC Underflow Status	474
0x020	ADCSSPRI	R/W	0x0000.3210	ADC Sample Sequencer Priority	475
0x028	ADCPSSI	WO	-	ADC Processor Sample Sequence Initiate	477
0x030	ADCSAC	R/W	0x0000.0000	ADC Sample Averaging Control	479
0x034	ADCDCISC	R/W1C	0x0000.0000	ADC Digital Comparator Interrupt Status and Clear	480
0x038	ADCCTL	R/W	0x0000.0000	ADC Control	482

Offset	Name	Type	Reset	Description	See page
0x040	ADCSSMUX0	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 0	483
0x044	ADCSSCTL0	R/W	0x0000.0000	ADC Sample Sequence Control 0	485
0x048	ADCSSFIFO0	RO	0x0000.0000	ADC Sample Sequence Result FIFO 0	488
0x04C	ADCSSFSTAT0	RO	0x0000.0100	ADC Sample Sequence FIFO 0 Status	489
0x050	ADCSSOP0	R/W	0x0000.0000	ADC Sample Sequence 0 Operation	491
0x054	ADCSSDC0	R/W	0x0000.0000	ADC Sample Sequence 0 Digital Comparator Select	493
0x060	ADCSSMUX1	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 1	495
0x064	ADCSSCTL1	R/W	0x0000.0000	ADC Sample Sequence Control 1	496
0x068	ADCSSFIFO1	RO	0x0000.0000	ADC Sample Sequence Result FIFO 1	488
0x06C	ADCSSFSTAT1	RO	0x0000.0100	ADC Sample Sequence FIFO 1 Status	489
0x070	ADCSSOP1	R/W	0x0000.0000	ADC Sample Sequence 1 Operation	498
0x074	ADCSSDC1	R/W	0x0000.0000	ADC Sample Sequence 1 Digital Comparator Select	499
0x080	ADCSSMUX2	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 2	495
0x084	ADCSSCTL2	R/W	0x0000.0000	ADC Sample Sequence Control 2	496
0x088	ADCSSFIFO2	RO	0x0000.0000	ADC Sample Sequence Result FIFO 2	488
0x08C	ADCSSFSTAT2	RO	0x0000.0100	ADC Sample Sequence FIFO 2 Status	489
0x090	ADCSSOP2	R/W	0x0000.0000	ADC Sample Sequence 2 Operation	498
0x094	ADCSSDC2	R/W	0x0000.0000	ADC Sample Sequence 2 Digital Comparator Select	499
0x0A0	ADCSSMUX3	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 3	501
0x0A4	ADCSSCTL3	R/W	0x0000.0002	ADC Sample Sequence Control 3	502
0x0A8	ADCSSFIFO3	RO	0x0000.0000	ADC Sample Sequence Result FIFO 3	488
0x0AC	ADCSSFSTAT3	RO	0x0000.0100	ADC Sample Sequence FIFO 3 Status	489
0x0B0	ADCSSOP3	R/W	0x0000.0000	ADC Sample Sequence 3 Operation	503
0x0B4	ADCSSDC3	R/W	0x0000.0000	ADC Sample Sequence 3 Digital Comparator Select	504
0xD00	ADCDCRIC	R/W	0x0000.0000	ADC Digital Comparator Reset Initial Conditions	505
0xE00	ADCDCCTL0	R/W	0x0000.0000	ADC Digital Comparator Control 0	509
0xE04	ADCDCCTL1	R/W	0x0000.0000	ADC Digital Comparator Control 1	509
0xE08	ADCDCCTL2	R/W	0x0000.0000	ADC Digital Comparator Control 2	509
0xE0C	ADCDCCTL3	R/W	0x0000.0000	ADC Digital Comparator Control 3	509
0xE10	ADCDCCTL4	R/W	0x0000.0000	ADC Digital Comparator Control 4	509
0xE14	ADCDCCTL5	R/W	0x0000.0000	ADC Digital Comparator Control 5	509
0xE18	ADCDCCTL6	R/W	0x0000.0000	ADC Digital Comparator Control 6	509
0xE1C	ADCDCCTL7	R/W	0x0000.0000	ADC Digital Comparator Control 7	509



Offset	Name	Type	Reset	Description	See page
0xE40	ADCDCOMP0	R/W	0x0000.0000	ADC Digital Comparator Range 0	512
0xE44	ADCDCOMP1	R/W	0x0000.0000	ADC Digital Comparator Range 1	512
0xE48	ADCDCOMP2	R/W	0x0000.0000	ADC Digital Comparator Range 2	512
0xE4C	ADCDCOMP3	R/W	0x0000.0000	ADC Digital Comparator Range 3	512
0xE50	ADCDCOMP4	R/W	0x0000.0000	ADC Digital Comparator Range 4	512
0xE54	ADCDCOMP5	R/W	0x0000.0000	ADC Digital Comparator Range 5	512
0xE58	ADCDCOMP6	R/W	0x0000.0000	ADC Digital Comparator Range 6	512
0xE5C	ADCDCOMP7	R/W	0x0000.0000	ADC Digital Comparator Range 7	512

## 13.5 Register Descriptions

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

### Register 1: ADC Active Sample Sequencer (ADCACTSS), offset 0x000

This register controls the activation of the sample sequencers. Each sample sequencer can be enabled or disabled independently.

#### ADC Active Sample Sequencer (ADCACTSS)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x000  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved												ASEN3	ASEN2	ASEN1	ASEN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	Reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	ASEN3	R/W	0	ADC SS3 Enable  When set, this bit enables Sample Sequencer 3. When clear, this bit disables Sample Sequencer 3.
2	ASEN2	R/W	0	ADC SS2 Enable  When set, this bit enables Sample Sequencer 2. When clear, this bit disables Sample Sequencer 2.
1	ASEN1	R/W	0	ADC SS1 Enable  When set, this bit enables Sample Sequencer 1. When clear, this bit disables Sample Sequencer 1.
0	ASEN0	R/W	0	ADC SS0 Enable  When set, this bit enables Sample Sequencer 0. When clear, this bit disables Sample Sequencer 0.

## 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 sending the interrupts to the interrupt controller.

### ADC Raw Interrupt Status (ADCRIS)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x004  
 Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved															INRDC
Type	RO	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												INR3	INR2	INR1	INR0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:17	Reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	INRDC	RO	0	Digital Comparator Raw Interrupt Status  This bit is set when at least one bit in the <b>ADCDCISC</b> register is set, meaning that a digital comparator interrupt has occurred.  This bit is clear when all bits in the <b>ADCDCISC</b> register are clear.
15:4	Reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	INR3	RO	0	SS3 Raw Interrupt Status  This bit is set when a sample has completed conversion and the respective <b>ADCSSCTL3 IEn</b> bit is set, enabling a raw interrupt.  This bit is cleared by writing a 1 to the <b>IN3</b> bit in the <b>ADCISC</b> register.
2	INR2	RO	0	SS2 Raw Interrupt Status  This bit is set when a sample has completed conversion and the respective <b>ADCSSCTL2 IEn</b> bit is set, enabling a raw interrupt.  This bit is cleared by writing a 1 to the <b>IN2</b> bit in the <b>ADCISC</b> register.
1	INR1	RO	0	SS1 Raw Interrupt Status  This bit is set when a sample has completed conversion and the respective <b>ADCSSCTL1 IEn</b> bit is set, enabling a raw interrupt.  This bit is cleared by writing a 1 to the <b>IN1</b> bit in the <b>ADCISC</b> register.

Bit/Field	Name	Type	Reset	Description
0	INR0	RO	0	<p>SS0 Raw Interrupt Status</p> <p>This bit is set when a sample has completed conversion and the respective <b>ADCSSCTL0</b> <i>IE<sub>n</sub></i> bit is set, enabling a raw interrupt.</p> <p>This bit is cleared by writing a 1 to the <i>IN0</i> bit in the <b>ADCISC</b> register.</p>

### Register 3: ADC Interrupt Mask (ADCIM), offset 0x008

This register controls whether the sample sequencer and digital comparator raw interrupt signals are sent to the interrupt controller. Each raw interrupt signal can be masked independently. Only a single `DCONSSn` bit should be set at any given time. Setting more than one of these bits results in the `INRDC` bit from the `ADCRIS` register being masked, and no interrupt is generated on any of the sample sequencer interrupt lines.

#### ADC Interrupt Mask (ADCIM)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x008  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved												DCONSS3	DCONSS2	DCONSS1	DCONSS0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved												MASK3	MASK2	MASK1	MASK0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:20	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.
19	DCONSS3	R/W	0	Digital Comparator Interrupt on SS3  When this bit is set, the raw interrupt signal from the digital comparators ( <code>INRDC</code> bit in the <code>ADCRIS</code> register) is sent to the interrupt controller on the SS3 interrupt line.  When this bit is clear, the status of the digital comparators does not affect the SS3 interrupt status.
18	DCONSS2	R/W	0	Digital Comparator Interrupt on SS2  When this bit is set, the raw interrupt signal from the digital comparators ( <code>INRDC</code> bit in the <code>ADCRIS</code> register) is sent to the interrupt controller on the SS2 interrupt line.  When this bit is clear, the status of the digital comparators does not affect the SS2 interrupt status.
17	DCONSS1	R/W	0	Digital Comparator Interrupt on SS1  When this bit is set, the raw interrupt signal from the digital comparators ( <code>INRDC</code> bit in the <code>ADCRIS</code> register) is sent to the interrupt controller on the SS1 interrupt line.  When this bit is clear, the status of the digital comparators does not affect the SS1 interrupt status.

Bit/Field	Name	Type	Reset	Description
16	DCONSS0	R/W	0	<p>Digital Comparator Interrupt on SS0</p> <p>When this bit is set, the raw interrupt signal from the digital comparators (<b>INRDC</b> bit in the <b>ADCRIS</b> register) is sent to the interrupt controller on the SS0 interrupt line.</p> <p>When this bit is clear, the status of the digital comparators does not affect the SS0 interrupt status.</p>
15:4	Reserved	RO	0x000	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>
3	MASK3	R/W	0	<p>SS3 Interrupt Mask</p> <p>When this bit is set, the raw interrupt signal from Sample Sequencer 3 (<b>ADCRIS</b> register <b>INR3</b> bit) is sent to the interrupt controller.</p> <p>When this bit is clear, the status of Sample Sequencer 3 does not affect the SS3 interrupt status.</p>
2	MASK2	R/W	0	<p>SS2 Interrupt Mask</p> <p>When this bit is set, the raw interrupt signal from Sample Sequencer 2 (<b>ADCRIS</b> register <b>INR2</b> bit) is sent to the interrupt controller.</p> <p>When this bit is clear, the status of Sample Sequencer 2 does not affect the SS2 interrupt status.</p>
1	MASK1	R/W	0	<p>SS1 Interrupt Mask</p> <p>When this bit is set, the raw interrupt signal from Sample Sequencer 1 (<b>ADCRIS</b> register <b>INR1</b> bit) is sent to the interrupt controller.</p> <p>When this bit is clear, the status of Sample Sequencer 1 does not affect the SS1 interrupt status.</p>
0	MASK0	R/W	0	<p>SS0 Interrupt Mask</p> <p>When this bit is set, the raw interrupt signal from Sample Sequencer 0 (<b>ADCRIS</b> register <b>INR0</b> bit) is sent to the interrupt controller.</p> <p>When this bit is clear, the status of Sample Sequencer 0 does not affect the SS0 interrupt status.</p>

## Register 4: ADC Interrupt Status and Clear (ADCISC), offset 0x00C

This register provides the mechanism for clearing sample sequencer interrupt conditions and shows the status of interrupts generated by the sample sequencers and the digital comparators which have been sent to the interrupt controller. When read, each bit field is the logical AND of the respective **INR** and **MASK** bits. Sample sequencer interrupts are cleared by writing a 1 to the corresponding bit position. Digital comparator interrupts are cleared by writing a 1 to the appropriate bits in the **ADCDCISC** register. If software is polling the **ADCRIS** instead of generating interrupts, the sample sequence **INR<sub>n</sub>** bits are still cleared via the **ADCISC** register, even if the **IN<sub>n</sub>** bit is not set.

### ADC Interrupt Status and Clear (ADCISC)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x00C  
 Type R/W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved												DCINSS3	DCINSS2	DCINSS1	DCINSS0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved												IN3	IN2	IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:20	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.
19	DCINSS3	RO	0	Digital Comparator Interrupt Status on SS3  This bit is set when both the <b>INRDC</b> bit in the <b>ADCRIS</b> register and the <b>DCONSS3</b> bit in the <b>ADCIM</b> register are set, providing a level-base interrupt to the interrupt controller.  This bit is cleared by writing a 1 to the appropriate location in the <b>ADCDCISC</b> register. Clearing the <b>ADCDCISC</b> register clears the <b>INRDC</b> bit.
18	DCINSS2	RO	0	Digital Comparator Interrupt Status on SS2  This bit is set when both the <b>INRDC</b> bit in the <b>ADCRIS</b> register and the <b>DCONSS2</b> bit in the <b>ADCIM</b> register are set, providing a level-base interrupt to the interrupt controller.  This bit is cleared by writing a 1 to the appropriate location in the <b>ADCDCISC</b> register. Clearing the <b>ADCDCISC</b> register clears the <b>INRDC</b> bit.
17	DCINSS1	RO	0	Digital Comparator Interrupt Status on SS1  This bit is set when both the <b>INRDC</b> bit in the <b>ADCRIS</b> register and the <b>DCONSS1</b> bit in the <b>ADCIM</b> register are set, providing a level-base interrupt to the interrupt controller.  This bit is cleared by writing a 1 to the appropriate location in the <b>ADCDCISC</b> register. Clearing the <b>ADCDCISC</b> register clears the <b>INRDC</b> bit.

Bit/Field	Name	Type	Reset	Description
16	DCINSS0	RO	0	<p>Digital Comparator Interrupt Status on SS0</p> <p>This bit is set when both the <code>INRDC</code> bit in the <b>ADCRIS</b> register and the <code>DCONSS0</code> bit in the <b>ADCIM</b> register are set, providing a level-base interrupt to the interrupt controller.</p> <p>This bit is cleared by writing a 1 to the appropriate location in the <b>ADDCISC</b> register. Clearing the <b>ADDCISC</b> register clears the <code>INRDC</code> bit.</p>
15:4	Reserved	RO	0x000	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>
3	IN3	R/W1C	0	<p>SS3 Interrupt Status and Clear</p> <p>This bit is set when both the <code>INR3</code> bit in the <b>ADCRIS</b> register and the <code>MASK3</code> bit in the <b>ADCIM</b> register are set, providing a level-based interrupt to the interrupt controller.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the <code>INR3</code> bit.</p>
2	IN2	R/W1C	0	<p>SS2 Interrupt Status and Clear</p> <p>This bit is set when both the <code>INR2</code> bit in the <b>ADCRIS</b> register and the <code>MASK2</code> bit in the <b>ADCIM</b> register are set, providing a level-based interrupt to the interrupt controller.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the <code>INR2</code> bit.</p>
1	IN1	R/W1C	0	<p>SS1 Interrupt Status and Clear</p> <p>This bit is set when both the <code>INR1</code> bit in the <b>ADCRIS</b> register and the <code>MASK1</code> bit in the <b>ADCIM</b> register are set, providing a level-based interrupt to the interrupt controller.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the <code>INR1</code> bit.</p>
0	IN0	R/W1C	0	<p>SS0 Interrupt Status and Clear</p> <p>This bit is set when both the <code>INR0</code> bit in the <b>ADCRIS</b> register and the <code>MASK0</code> bit in the <b>ADCIM</b> register are set, providing a level-based interrupt to the interrupt controller.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the <code>INR0</code> bit.</p>



## Register 5: ADC Overflow Status (ADCOSTAT), offset 0x010

This register indicates overflow conditions in the sample sequencer FIFOs. Once the overflow condition has been handled by software, the condition can be cleared by writing a 1 to the corresponding bit position.

### ADC Overflow Status (ADCOSTAT)

ADC0 base: 0x4003.8000

ADC1 base: 0x4003.9000

Offset 0x010

Type R/W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	Reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Reserved													OV3	OV2	OV1	OV0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

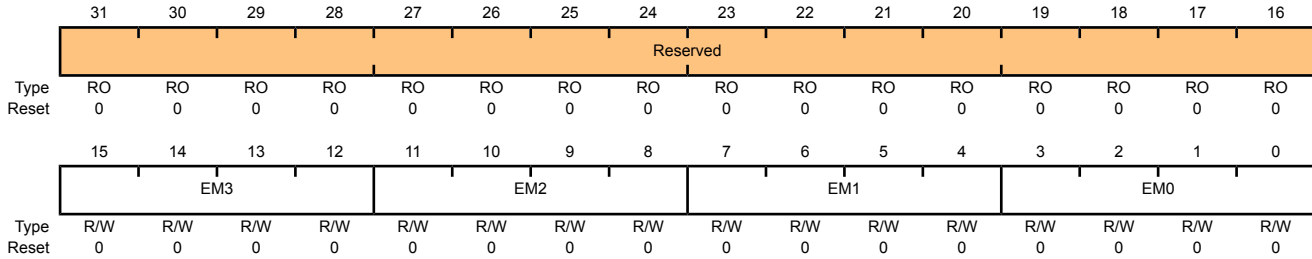
Bit/Field	Name	Type	Reset	Description
31:4	Reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	OV3	R/W1C	0	<p>SS3 FIFO Overflow</p> <p>This bit is set when the FIFO for Sample Sequencer 3 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.</p> <p>This bit is cleared by writing a 1.</p>
2	OV2	R/W1C	0	<p>SS2 FIFO Overflow</p> <p>This bit is set when the FIFO for Sample Sequencer 2 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.</p> <p>This bit is cleared by writing a 1.</p>
1	OV1	R/W1C	0	<p>SS1 FIFO Overflow</p> <p>This bit is set when the FIFO for Sample Sequencer 1 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.</p> <p>This bit is cleared by writing a 1.</p>
0	OV0	R/W1C	0	<p>SS0 FIFO Overflow</p> <p>This bit is set when the FIFO for Sample Sequencer 0 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped.</p> <p>This bit is cleared by writing a 1.</p>

### Register 6: ADC Event Multiplexer Select (ADCEMUX), offset 0x014

The **ADCEMUX** selects the event (trigger) that initiates sampling for each sample sequencer. Each sample sequencer can be configured with a unique trigger source.

#### ADC Event Multiplexer Select (ADCEMUX)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x014  
 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.
15:12	EM3	R/W	0x0	SS3 Trigger Select

This field selects the trigger source for Sample Sequencer 3.  
 The valid configurations for this field are:

Value	Event
0x0	Processor (default)
0x1	Analog Comparator 0
0x2	Analog Comparator 1
0x3	Analog Comparator 2
0x4	External (GPIO PB4)
<b>Note:</b> PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input.	
0x5	Timer
In addition, the trigger must be enabled with the TnOTE bit in the GPTMCTL register (see page 399).	
0x6	PWM0
0x7	PWM1
0x8	PWM2
0x9	PWM3
0xA-0xE	Reserved
0xF	Always (continuously sample)

Bit/Field	Name	Type	Reset	Description																														
11:8	EM2	R/W	0x0	<p>SS2 Trigger Select</p> <p>This field selects the trigger source for Sample Sequencer 2.</p> <p>The valid configurations for this field are:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Event</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Processor (default)</td> </tr> <tr> <td>0x1</td> <td>Analog Comparator 0</td> </tr> <tr> <td>0x2</td> <td>Analog Comparator 1</td> </tr> <tr> <td>0x3</td> <td>Analog Comparator 2</td> </tr> <tr> <td>0x4</td> <td>External (GPIO PB4)</td> </tr> <tr> <td></td> <td><b>Note:</b> PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input.</td> </tr> <tr> <td>0x5</td> <td>Timer</td> </tr> <tr> <td></td> <td>In addition, the trigger must be enabled with the TnOTE bit in the <b>GPTMCTL</b> register (see page 399).</td> </tr> <tr> <td>0x6</td> <td>PWM0</td> </tr> <tr> <td>0x7</td> <td>PWM1</td> </tr> <tr> <td>0x8</td> <td>PWM2</td> </tr> <tr> <td>0x9</td> <td>PWM3</td> </tr> <tr> <td>0xA-0xE</td> <td>Reserved</td> </tr> <tr> <td>0xF</td> <td>Always (continuously sample)</td> </tr> </tbody> </table>	Value	Event	0x0	Processor (default)	0x1	Analog Comparator 0	0x2	Analog Comparator 1	0x3	Analog Comparator 2	0x4	External (GPIO PB4)		<b>Note:</b> PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input.	0x5	Timer		In addition, the trigger must be enabled with the TnOTE bit in the <b>GPTMCTL</b> register (see page 399).	0x6	PWM0	0x7	PWM1	0x8	PWM2	0x9	PWM3	0xA-0xE	Reserved	0xF	Always (continuously sample)
Value	Event																																	
0x0	Processor (default)																																	
0x1	Analog Comparator 0																																	
0x2	Analog Comparator 1																																	
0x3	Analog Comparator 2																																	
0x4	External (GPIO PB4)																																	
	<b>Note:</b> PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input.																																	
0x5	Timer																																	
	In addition, the trigger must be enabled with the TnOTE bit in the <b>GPTMCTL</b> register (see page 399).																																	
0x6	PWM0																																	
0x7	PWM1																																	
0x8	PWM2																																	
0x9	PWM3																																	
0xA-0xE	Reserved																																	
0xF	Always (continuously sample)																																	

Bit/Field	Name	Type	Reset	Description																														
7:4	EM1	R/W	0x0	<p>SS1 Trigger Select</p> <p>This field selects the trigger source for Sample Sequencer 1.</p> <p>The valid configurations for this field are:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Event</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Processor (default)</td> </tr> <tr> <td>0x1</td> <td>Analog Comparator 0</td> </tr> <tr> <td>0x2</td> <td>Analog Comparator 1</td> </tr> <tr> <td>0x3</td> <td>Analog Comparator 2</td> </tr> <tr> <td>0x4</td> <td>External (GPIO PB4)</td> </tr> <tr> <td></td> <td><b>Note:</b> PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input.</td> </tr> <tr> <td>0x5</td> <td>Timer</td> </tr> <tr> <td></td> <td>In addition, the trigger must be enabled with the TnOTE bit in the <b>GPTMCTL</b> register (see page 399).</td> </tr> <tr> <td>0x6</td> <td>PWM0</td> </tr> <tr> <td>0x7</td> <td>PWM1</td> </tr> <tr> <td>0x8</td> <td>PWM2</td> </tr> <tr> <td>0x9</td> <td>PWM3</td> </tr> <tr> <td>0xA-0xE</td> <td>Reserved</td> </tr> <tr> <td>0xF</td> <td>Always (continuously sample)</td> </tr> </tbody> </table>	Value	Event	0x0	Processor (default)	0x1	Analog Comparator 0	0x2	Analog Comparator 1	0x3	Analog Comparator 2	0x4	External (GPIO PB4)		<b>Note:</b> PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input.	0x5	Timer		In addition, the trigger must be enabled with the TnOTE bit in the <b>GPTMCTL</b> register (see page 399).	0x6	PWM0	0x7	PWM1	0x8	PWM2	0x9	PWM3	0xA-0xE	Reserved	0xF	Always (continuously sample)
Value	Event																																	
0x0	Processor (default)																																	
0x1	Analog Comparator 0																																	
0x2	Analog Comparator 1																																	
0x3	Analog Comparator 2																																	
0x4	External (GPIO PB4)																																	
	<b>Note:</b> PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input.																																	
0x5	Timer																																	
	In addition, the trigger must be enabled with the TnOTE bit in the <b>GPTMCTL</b> register (see page 399).																																	
0x6	PWM0																																	
0x7	PWM1																																	
0x8	PWM2																																	
0x9	PWM3																																	
0xA-0xE	Reserved																																	
0xF	Always (continuously sample)																																	

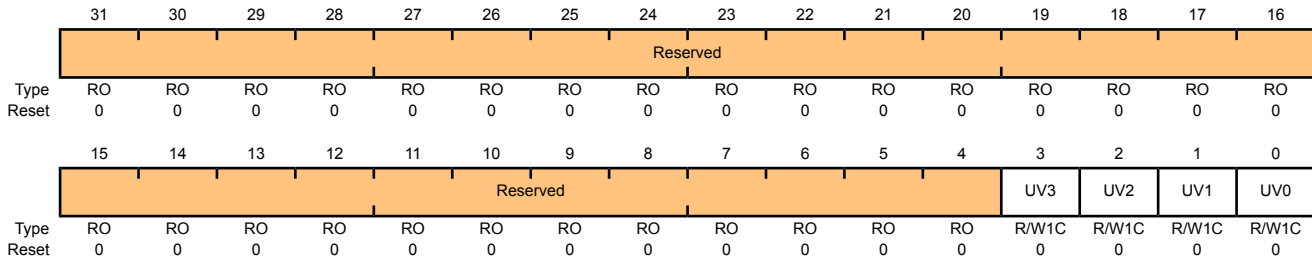
Bit/Field	Name	Type	Reset	Description																														
3:0	EM0	R/W	0x0	<p>SS0 Trigger Select</p> <p>This field selects the trigger source for Sample Sequencer 0.</p> <p>The valid configurations for this field are:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Event</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Processor (default)</td> </tr> <tr> <td>0x1</td> <td>Analog Comparator 0</td> </tr> <tr> <td>0x2</td> <td>Analog Comparator 1</td> </tr> <tr> <td>0x3</td> <td>Analog Comparator 2</td> </tr> <tr> <td>0x4</td> <td>External (GPIO PB4)</td> </tr> <tr> <td></td> <td><b>Note:</b> PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input.</td> </tr> <tr> <td>0x5</td> <td>Timer</td> </tr> <tr> <td></td> <td>In addition, the trigger must be enabled with the TnOTE bit in the <b>GPTMCTL</b> register (see page 399).</td> </tr> <tr> <td>0x6</td> <td>PWM0</td> </tr> <tr> <td>0x7</td> <td>PWM1</td> </tr> <tr> <td>0x8</td> <td>PWM2</td> </tr> <tr> <td>0x9</td> <td>PWM3</td> </tr> <tr> <td>0xA-0xE</td> <td>Reserved</td> </tr> <tr> <td>0xF</td> <td>Always (continuously sample)</td> </tr> </tbody> </table>	Value	Event	0x0	Processor (default)	0x1	Analog Comparator 0	0x2	Analog Comparator 1	0x3	Analog Comparator 2	0x4	External (GPIO PB4)		<b>Note:</b> PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input.	0x5	Timer		In addition, the trigger must be enabled with the TnOTE bit in the <b>GPTMCTL</b> register (see page 399).	0x6	PWM0	0x7	PWM1	0x8	PWM2	0x9	PWM3	0xA-0xE	Reserved	0xF	Always (continuously sample)
Value	Event																																	
0x0	Processor (default)																																	
0x1	Analog Comparator 0																																	
0x2	Analog Comparator 1																																	
0x3	Analog Comparator 2																																	
0x4	External (GPIO PB4)																																	
	<b>Note:</b> PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input.																																	
0x5	Timer																																	
	In addition, the trigger must be enabled with the TnOTE bit in the <b>GPTMCTL</b> register (see page 399).																																	
0x6	PWM0																																	
0x7	PWM1																																	
0x8	PWM2																																	
0x9	PWM3																																	
0xA-0xE	Reserved																																	
0xF	Always (continuously sample)																																	

### Register 7: ADC Underflow Status (ADCUSTAT), offset 0x018

This register indicates underflow conditions in the sample sequencer FIFOs. The corresponding underflow condition is cleared by writing a 1 to the relevant bit position.

#### ADC Underflow Status (ADCUSTAT)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x018  
 Type R/W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	Reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	UV3	R/W1C	0	<p>SS3 FIFO Underflow</p> <p>This bit is set when the FIFO for Sample Sequencer 3 has hit an underflow condition, meaning that the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned.</p> <p>This bit is cleared by writing a 1.</p>
2	UV2	R/W1C	0	<p>SS2 FIFO Underflow</p> <p>This bit is set when the FIFO for Sample Sequencer 2 has hit an underflow condition, meaning that the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned.</p> <p>This bit is cleared by writing a 1.</p>
1	UV1	R/W1C	0	<p>SS1 FIFO Underflow</p> <p>This bit is set when the FIFO for Sample Sequencer 1 has hit an underflow condition, meaning that the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned.</p> <p>This bit is cleared by writing a 1.</p>
0	UV0	R/W1C	0	<p>SS0 FIFO Underflow</p> <p>This bit is set when the FIFO for Sample Sequencer 0 has hit an underflow condition, meaning that the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned.</p> <p>This bit is cleared by writing a 1.</p>

## Register 8: ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020

This register sets the priority for each of the sample sequencers. Out of reset, Sequencer 0 has the highest priority, and Sequencer 3 has the lowest priority. When reconfiguring sequence priorities, each sequence must have a unique priority for the ADC to operate properly.

### ADC Sample Sequencer Priority (ADCSSPRI)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x020  
 Type R/W, reset 0x0000.3210

	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		SS3		Reserved		SS2		Reserved		SS1		Reserved		SS0	
Type	RO	RO	R/W	R/W	RO	RO	R/W	R/W	RO	RO	R/W	R/W	RO	RO	R/W	R/W
Reset	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:14	Reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13:12	SS3	R/W	0x3	SS3 Priority  This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 3. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
11:10	Reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:8	SS2	R/W	0x2	SS2 Priority  This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 2. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
7:6	Reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:4	SS1	R/W	0x1	SS1 Priority  This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 1. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
3:2	Reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
1:0	SS0	R/W	0x0	SS0 Priority  This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 0. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.



## 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.

This register also provides a means to configure and then initiate concurrent sampling on all ADC modules. To do this, the first ADC module should be configured. The **ADCPSSI** register for that module should then be written. The appropriate **SS** bits should be set along with the **SYNCWAIT** bit. Additional ADC modules should then be configured following the same procedure. Once the final ADC module is configured, its **ADCPSSI** register should be written with the appropriate **SS** bits set along with the **GSYNC** bit. All of the ADC modules then begin concurrent sampling according to their configuration.

### ADC Processor Sample Sequence Initiate (ADCPSSI)

ADC0 base: 0x4003.8000

ADC1 base: 0x4003.9000

Offset 0x028

Type WO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	GSYNC	Reserved			SYNCWAIT	Reserved										
Type	R/W	WO	WO	WO	R/W	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	0	-	-	-	0	-	-	-	-	-	-	-	-	-	-	-
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved												SS3	SS2	SS1	SS0
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31	GSYNC	R/W	0	Global Synchronize  When set, this bit initiates sampling in multiple ADC modules at the same time. Any ADC module that has been initialized by setting an <b>SS<sub>n</sub></b> bit and the <b>SYNCWAIT</b> bit starts sampling once this bit is written.  This bit is cleared once sampling has been initiated.
30:28	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.
27	SYNCWAIT	R/W	0	Synchronize Wait  When set, this bit allows the sample sequences to be initiated, but delays sampling until the <b>GSYNC</b> bit is set.  When this bit is clear, sampling begins when a sample sequence has been initiated.
26:4	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.

Bit/Field	Name	Type	Reset	Description
3	SS3	WO	-	<p>SS3 Initiate</p> <p>Setting this bit triggers sampling on Sample Sequencer 3, if the sequencer is enabled in the <b>ADCACTSS</b> register.</p> <p>Only a write by software is valid; a read of this register returns no meaningful data.</p>
2	SS2	WO	-	<p>SS2 Initiate</p> <p>Setting this bit triggers sampling on Sample Sequencer 2, if the sequencer is enabled in the <b>ADCACTSS</b> register.</p> <p>Only a write by software is valid; a read of this register returns no meaningful data.</p>
1	SS1	WO	-	<p>SS1 Initiate</p> <p>Setting this bit triggers sampling on Sample Sequencer 1, if the sequencer is enabled in the <b>ADCACTSS</b> register.</p> <p>Only a write by software is valid; a read of this register returns no meaningful data.</p>
0	SS0	WO	-	<p>SS0 Initiate</p> <p>Setting, this bit triggers sampling on Sample Sequencer 0, if the sequencer is enabled in the <b>ADCACTSS</b> register.</p> <p>Only a write by software is valid; a read of this register returns no meaningful data.</p>

## 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)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x030  
 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													AVG			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	Reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
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 Digital Comparator Interrupt Status and Clear (ADCDCISC), offset 0x034

This register provides status and acknowledgement of digital comparator interrupts. One bit is provided for each comparator.

### ADC Digital Comparator Interrupt Status and Clear (ADCDCISC)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x034  
 Type R/W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								DCINT7	DCINT6	DCINT5	DCINT4	DCINT3	DCINT2	DCINT1	DCINT0
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	Reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	DCINT7	R/W1C	0	Digital Comparator 7 Interrupt Status and Clear This bit is set when Digital Comparator 7 generates an interrupt. This bit is cleared by writing a 1.
6	DCINT6	R/W1C	0	Digital Comparator 6 Interrupt Status and Clear This bit is set when Digital Comparator 6 generates an interrupt. This bit is cleared by writing a 1.
5	DCINT5	R/W1C	0	Digital Comparator 5 Interrupt Status and Clear This bit is set when Digital Comparator 5 generates an interrupt. This bit is cleared by writing a 1.
4	DCINT4	R/W1C	0	Digital Comparator 4 Interrupt Status and Clear This bit is set when Digital Comparator 4 generates an interrupt. This bit is cleared by writing a 1.
3	DCINT3	R/W1C	0	Digital Comparator 3 Interrupt Status and Clear This bit is set when Digital Comparator 3 generates an interrupt. This bit is cleared by writing a 1.
2	DCINT2	R/W1C	0	Digital Comparator 2 Interrupt Status and Clear This bit is set when Digital Comparator 2 generates an interrupt. This bit is cleared by writing a 1.

Bit/Field	Name	Type	Reset	Description
1	DCINT1	R/W1C	0	Digital Comparator 1 Interrupt Status and Clear This bit is set when Digital Comparator 1 generates an interrupt. This bit is cleared by writing a 1.
0	DCINT0	R/W1C	0	Digital Comparator 0 Interrupt Status and Clear This bit is set when Digital Comparator 0 generates an interrupt. This bit is cleared by writing a 1.

## Register 12: ADC Control (ADCCTL), offset 0x038

This register selects the voltage reference.

### ADC Control (ADCCTL)

ADC0 base: 0x4003.8000

ADC1 base: 0x4003.9000

Offset 0x038

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	Reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	VREF	R/W	0	<p>Voltage Reference Select</p> <p>When set, this bit selects the external <code>VREFA</code> input as the voltage reference.</p> <p>When clear, this bit selects the internal reference as the voltage reference.</p>

## Register 13: 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)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x040  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	MUX7				MUX6				MUX5				MUX4			
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
	MUX3				MUX2				MUX1				MUX0			
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
31:28	MUX7	R/W	0x0	8th Sample Input Select  The MUX7 field is used during the eighth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion. The value set here indicates the corresponding pin, for example, a value of 0x1 indicates the input is AIN1.
27:24	MUX6	R/W	0x0	7th Sample Input Select  The MUX6 field is used during the seventh sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
23:20	MUX5	R/W	0x0	6th Sample Input Select  The MUX5 field is used during the sixth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
19:16	MUX4	R/W	0x0	5th Sample Input Select  The MUX4 field is used during the fifth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
15:12	MUX3	R/W	0x0	4th Sample Input Select  The MUX3 field is used during the fourth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
11:8	MUX2	R/W	0x0	3rd Sample Input Select  The MUX2 field is used during the third sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.

Bit/Field	Name	Type	Reset	Description
7:4	MUX1	R/W	0x0	2nd Sample Input Select  The MUX1 field is used during the second sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
3:0	MUX0	R/W	0x0	1st Sample Input Select  The MUX0 field is used during the first sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.



## Register 14: ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044

This register contains the configuration information for each sample for a sequence executed with a sample sequencer. When configuring a sample sequence, the `END` bit must be set for the final sample, whether it be after the first sample, eighth 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)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x044  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TS7	IE7	END7	D7	TS6	IE6	END6	D6	TS5	IE5	END5	D5	TS4	IE4	END4	D4
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	TS7	R/W	0	<p>8th Sample Temp Sensor Select</p> <p>When this bit is set, the temperature sensor is read during the eighth sample of the sample sequence.</p> <p>When this bit is clear, the input pin specified by the <code>ADCSSMUXn</code> register is read during the eighth sample of the sample sequence.</p>
30	IE7	R/W	0	<p>8th Sample Interrupt Enable</p> <p>When this bit is set, the raw interrupt signal (<code>INR0</code> bit) is asserted at the end of the eighth sample's conversion. If the <code>MASK0</code> bit in the <code>ADCIM</code> register is set, the interrupt is promoted to the interrupt controller.</p> <p>When this bit is clear, the raw interrupt is not asserted to the interrupt controller.</p> <p>It is legal to have multiple samples within a sequence generate interrupts.</p>
29	END7	R/W	0	<p>8th Sample is End of Sequence</p> <p>When this bit is set, the eighth sample is the last sample of the sequence. It is possible to end the sequence on any sample position. Samples defined after the sample containing a set <code>ENDn</code> bit are not requested for conversion even though the fields may be non-zero.</p> <p>When this bit is clear, another sample in the sequence is the final sample. Software must set an <code>ENDn</code> bit somewhere within the sequence.</p>
28	D7	R/W	0	<p>8th Sample Diff Input Select</p> <p>When this bit is set, the analog input is differentially sampled. The corresponding <code>ADCSSMUXn</code> nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1".</p> <p>When this bit is clear, the analog inputs are not differentially sampled.</p> <p>Because the temperature sensor does not have a differential option, this bit must not be set when the <code>TS7</code> bit is set.</p>

Bit/Field	Name	Type	Reset	Description
27	TS6	R/W	0	7th Sample Temp Sensor Select Same definition as TS7 but used during the seventh sample.
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.

Bit/Field	Name	Type	Reset	Description
11	TS2	R/W	0	3rd Sample Temp Sensor Select Same definition as TS7 but used during the third sample.
10	IE2	R/W	0	3rd Sample Interrupt Enable Same definition as IE7 but used during the third sample.
9	END2	R/W	0	3rd Sample is End of Sequence Same definition as END7 but used during the third sample.
8	D2	R/W	0	3rd Sample Diff Input Select Same definition as D7 but used during the third sample.
7	TS1	R/W	0	2nd Sample Temp Sensor Select Same definition as TS7 but used during the second sample.
6	IE1	R/W	0	2nd Sample Interrupt Enable Same definition as IE7 but used during the second sample.
5	END1	R/W	0	2nd Sample is End of Sequence Same definition as END7 but used during the second sample.
4	D1	R/W	0	2nd Sample Diff Input Select Same definition as D7 but used during the second sample.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as TS7 but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as IE7 but used during the first sample.
1	END0	R/W	0	1st Sample is End of Sequence Same definition as END7 but used during the first sample.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as D7 but used during the first sample.

**Register 15: ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048**

**Register 16: ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068**

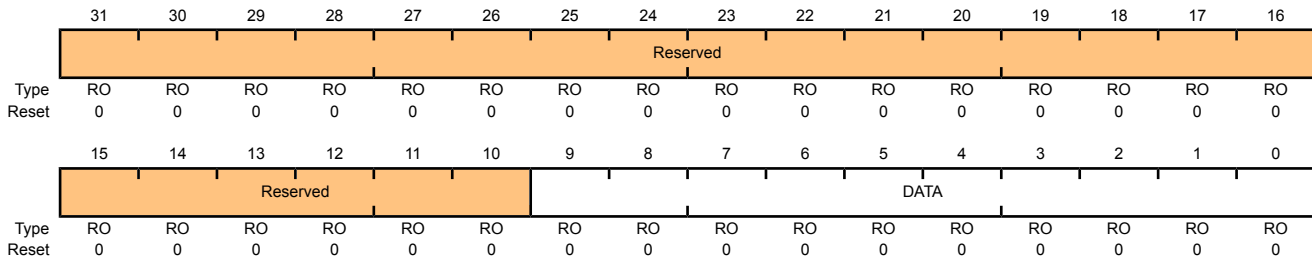
**Register 17: ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088**

**Register 18: 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)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x048  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:10	Reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:0	DATA	RO	0x000	Conversion Result Data

**Register 19: ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C**

**Register 20: ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C**

**Register 21: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C**

**Register 22: 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, which has 8 entries; **ADCSSFSTAT1** on FIFO1, which has 4 entries; **ADCSSFSTAT2** on FIFO2, which has 4 entries; and **ADCSSFSTAT3** on FIFO3 which has a single entry.

#### ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x04C  
 Type RO, reset 0x0000.0100

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

Bit/Field	Name	Type	Reset	Description
31:13	Reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	FULL	RO	0	FIFO Full When this bit is set, the FIFO is currently full. When this bit is clear, the FIFO is not currently full.
11:9	Reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	EMPTY	RO	1	FIFO Empty When this bit is set, the FIFO is currently empty. When this bit is clear, the FIFO is not currently empty.
7:4	HPTR	RO	0x0	FIFO Head Pointer This field contains the current "head" pointer index for the FIFO, that is, the next entry to be written.

Bit/Field	Name	Type	Reset	Description
3:0	TPTR	RO	0x0	FIFO Tail Pointer  This field contains the current "tail" pointer index for the FIFO, that is, the next entry to be read.

## Register 23: ADC Sample Sequence 0 Operation (ADCSSOP0), offset 0x050

This register determines whether the sample from the given conversion on Sample Sequence 0 is saved in the Sample Sequence FIFO0 or sent to the digital comparator unit.

### ADC Sample Sequence 0 Operation (ADCSSOP0)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x050  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved			S7DCOP	Reserved			S6DCOP	Reserved			S5DCOP	Reserved			S4DCOP
Type	RO	RO	RO	R/W	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			S3DCOP	Reserved			S2DCOP	Reserved			S1DCOP	Reserved			S0DCOP
Type	RO	RO	RO	R/W	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

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	S7DCOP	R/W	0	Sample 7 Digital Comparator Operation  When this bit is set, the eighth sample is sent to the digital comparator unit specified by the <i>S7DCSEL</i> bit in the <b>ADCSSDC0</b> register, and the value is not written to the FIFO.  When this bit is clear, the eighth sample is saved in Sample Sequence FIFO0.
27:25	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.
24	S6DCOP	R/W	0	Sample 6 Digital Comparator Operation  Same definition as <i>S7DCOP</i> but used during the seventh sample.
23:21	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.
20	S5DCOP	R/W	0	Sample 5 Digital Comparator Operation  Same definition as <i>S7DCOP</i> but used during the sixth sample.
19:17	Reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	S4DCOP	R/W	0	Sample 4 Digital Comparator Operation  Same definition as <i>S7DCOP</i> but used during the fifth sample.
15:13	Reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
12	S3DCOP	R/W	0	Sample 3 Digital Comparator Operation Same definition as <i>S7DCOP</i> but used during the fourth sample.
11:9	Reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	S2DCOP	R/W	0	Sample 2 Digital Comparator Operation Same definition as <i>S7DCOP</i> but used during the third sample.
7:5	Reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	S1DCOP	R/W	0	Sample 1 Digital Comparator Operation Same definition as <i>S7DCOP</i> but used during the second sample.
3:1	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	S0DCOP	R/W	0	Sample 0 Digital Comparator Operation Same definition as <i>S7DCOP</i> but used during the first sample.



## Register 24: ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0), offset 0x054

This register determines which digital comparator receives the sample from the given conversion on Sample Sequence 0, if the corresponding  $S_nDCOP$  bit in the **ADCSSOP0** register is set.

### ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x054  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	S7DCSEL				S6DCSEL				S5DCSEL				S4DCSEL			
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
	S3DCSEL				S2DCSEL				S1DCSEL				S0DCSEL			
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																		
31:28	S7DCSEL	R/W	0x0	<p>Sample 7 Digital Comparator Select</p> <p>When the <math>S7DCOP</math> bit in the <b>ADCSSOP0</b> register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the eighth sample from Sample Sequencer 0.</p> <p><b>Note:</b> Values not listed are reserved.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Digital Comparator Unit 0 (<b>ADCDCOMP0</b> and <b>ADCCCTL0</b>)</td> </tr> <tr> <td>0x1</td> <td>Digital Comparator Unit 1 (<b>ADCDCOMP1</b> and <b>ADCCCTL1</b>)</td> </tr> <tr> <td>0x2</td> <td>Digital Comparator Unit 2 (<b>ADCDCOMP2</b> and <b>ADCCCTL2</b>)</td> </tr> <tr> <td>0x3</td> <td>Digital Comparator Unit 3 (<b>ADCDCOMP3</b> and <b>ADCCCTL3</b>)</td> </tr> <tr> <td>0x4</td> <td>Digital Comparator Unit 4 (<b>ADCDCOMP4</b> and <b>ADCCCTL4</b>)</td> </tr> <tr> <td>0x5</td> <td>Digital Comparator Unit 5 (<b>ADCDCOMP5</b> and <b>ADCCCTL5</b>)</td> </tr> <tr> <td>0x6</td> <td>Digital Comparator Unit 6 (<b>ADCDCOMP6</b> and <b>ADCCCTL6</b>)</td> </tr> <tr> <td>0x7</td> <td>Digital Comparator Unit 7 (<b>ADCDCOMP7</b> and <b>ADCCCTL7</b>)</td> </tr> </tbody> </table>	Value	Description	0x0	Digital Comparator Unit 0 ( <b>ADCDCOMP0</b> and <b>ADCCCTL0</b> )	0x1	Digital Comparator Unit 1 ( <b>ADCDCOMP1</b> and <b>ADCCCTL1</b> )	0x2	Digital Comparator Unit 2 ( <b>ADCDCOMP2</b> and <b>ADCCCTL2</b> )	0x3	Digital Comparator Unit 3 ( <b>ADCDCOMP3</b> and <b>ADCCCTL3</b> )	0x4	Digital Comparator Unit 4 ( <b>ADCDCOMP4</b> and <b>ADCCCTL4</b> )	0x5	Digital Comparator Unit 5 ( <b>ADCDCOMP5</b> and <b>ADCCCTL5</b> )	0x6	Digital Comparator Unit 6 ( <b>ADCDCOMP6</b> and <b>ADCCCTL6</b> )	0x7	Digital Comparator Unit 7 ( <b>ADCDCOMP7</b> and <b>ADCCCTL7</b> )
Value	Description																					
0x0	Digital Comparator Unit 0 ( <b>ADCDCOMP0</b> and <b>ADCCCTL0</b> )																					
0x1	Digital Comparator Unit 1 ( <b>ADCDCOMP1</b> and <b>ADCCCTL1</b> )																					
0x2	Digital Comparator Unit 2 ( <b>ADCDCOMP2</b> and <b>ADCCCTL2</b> )																					
0x3	Digital Comparator Unit 3 ( <b>ADCDCOMP3</b> and <b>ADCCCTL3</b> )																					
0x4	Digital Comparator Unit 4 ( <b>ADCDCOMP4</b> and <b>ADCCCTL4</b> )																					
0x5	Digital Comparator Unit 5 ( <b>ADCDCOMP5</b> and <b>ADCCCTL5</b> )																					
0x6	Digital Comparator Unit 6 ( <b>ADCDCOMP6</b> and <b>ADCCCTL6</b> )																					
0x7	Digital Comparator Unit 7 ( <b>ADCDCOMP7</b> and <b>ADCCCTL7</b> )																					
27:24	S6DCSEL	R/W	0x0	<p>Sample 6 Digital Comparator Select</p> <p>This field has the same encodings as <math>S7DCSEL</math> but is used during the seventh sample.</p>																		
23:20	S5DCSEL	R/W	0x0	<p>Sample 5 Digital Comparator Select</p> <p>This field has the same encodings as <math>S7DCSEL</math> but is used during the sixth sample.</p>																		
19:16	S4DCSEL	R/W	0x0	<p>Sample 4 Digital Comparator Select</p> <p>This field has the same encodings as <math>S7DCSEL</math> but is used during the fifth sample.</p>																		

Bit/Field	Name	Type	Reset	Description
15:12	S3DCSEL	R/W	0x0	Sample 3 Digital Comparator Select This field has the same encodings as S7DCSEL but is used during the fourth sample.
11:8	S2DCSEL	R/W	0x0	Sample 2 Digital Comparator Select This field has the same encodings as S7DCSEL but is used during the third sample.
7:4	S1DCSEL	R/W	0x0	Sample 1 Digital Comparator Select This field has the same encodings as S7DCSEL but is used during the second sample.
3:0	S0DCSEL	R/W	0x0	Sample 0 Digital Comparator Select This field has the same encodings as S7DCSEL but is used during the first sample.

## Register 25: ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060

## Register 26: 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 483 for detailed bit descriptions. The **ADCSSMUX1** register affects Sample Sequencer 1 and the **ADCSSMUX2** register affects Sample Sequencer 2.

### ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x060  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MUX3				MUX2				MUX1				MUX0			
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	Reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:12	MUX3	R/W	0x0	4th Sample Input Select
11:8	MUX2	R/W	0x0	3rd Sample Input Select
7:4	MUX1	R/W	0x0	2nd Sample Input Select
3:0	MUX0	R/W	0x0	1st Sample Input Select

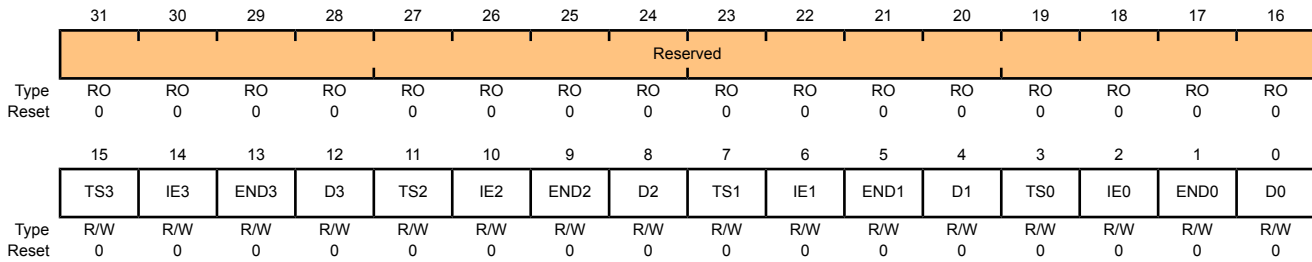
**Register 27: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064**

**Register 28: 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 for the final sample, whether it be after the first sample, fourth sample, or any sample in between. These registers are 16-bits wide and contain information for four possible samples. See the **ADCSSCTL0** register on page 485 for detailed bit descriptions. The **ADCSSCTL1** register configures Sample Sequencer 1 and the **ADCSSCTL2** register configures Sample Sequencer 2.

ADC Sample Sequence Control 1 (ADCSSCTL1)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x064  
 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	TS3	R/W	0	4th Sample Temp Sensor Select Same definition as TS7 but used during the fourth sample.
14	IE3	R/W	0	4th Sample Interrupt Enable Same definition as IE7 but used during the fourth sample.
13	END3	R/W	0	4th Sample is End of Sequence Same definition as END7 but used during the fourth sample.
12	D3	R/W	0	4th Sample Diff Input Select Same definition as D7 but used during the fourth sample.
11	TS2	R/W	0	3rd Sample Temp Sensor Select Same definition as TS7 but used during the third sample.
10	IE2	R/W	0	3rd Sample Interrupt Enable Same definition as IE7 but used during the third sample.
9	END2	R/W	0	3rd Sample is End of Sequence Same definition as END7 but used during the third sample.
8	D2	R/W	0	3rd Sample Diff Input Select Same definition as D7 but used during the third sample.

---

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.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as D7 but used during the first sample.

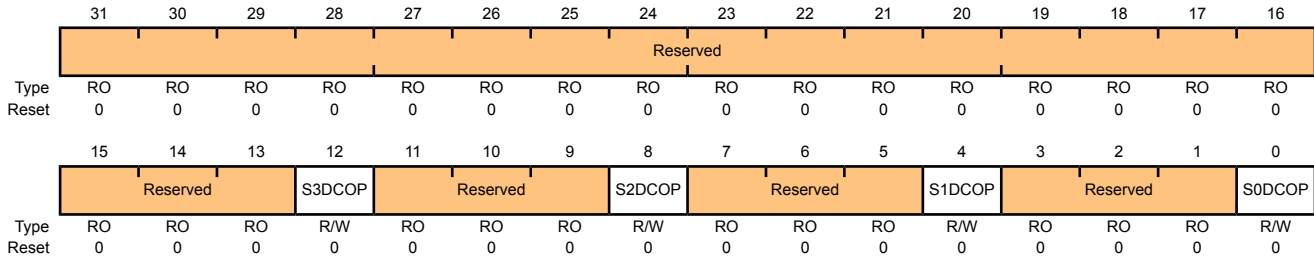
**Register 29: ADC Sample Sequence 1 Operation (ADCSSOP1), offset 0x070**

**Register 30: ADC Sample Sequence 2 Operation (ADCSSOP2), offset 0x090**

This register determines whether the sample from the given conversion on Sample Sequence n is saved in the Sample Sequence n FIFO or sent to the digital comparator unit. The **ADCSSOP1** register controls Sample Sequencer 1 and the **ADCSSOP2** register controls Sample Sequencer 2.

ADC Sample Sequence 1 Operation (ADCSSOP1)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x070  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:13	Reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	S3DCOP	R/W	0	Sample 3 Digital Comparator Operation  When this bit is set, the fourth sample is sent to the digital comparator unit specified by the S3DCSEL bit in the ADCSSDCn register, and the value is not written to the FIFO.  When this bit is clear, the fourth sample is saved in Sample Sequence FIFO.
11:9	Reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	S2DCOP	R/W	0	Sample 2 Digital Comparator Operation  Same definition as S3DCOP but used during the third sample.
7:5	Reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	S1DCOP	R/W	0	Sample 1 Digital Comparator Operation  Same definition as S3DCOP but used during the second sample.
3:1	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	S0DCOP	R/W	0	Sample 0 Digital Comparator Operation  Same definition as S3DCOP but used during the first sample.

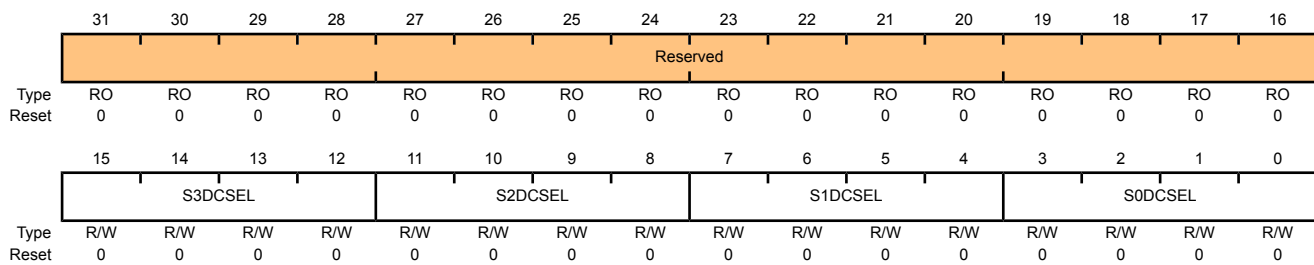
**Register 31: ADC Sample Sequence 1 Digital Comparator Select (ADCSSDC1), offset 0x074**

**Register 32: ADC Sample Sequence 2 Digital Comparator Select (ADCSSDC2), offset 0x094**

These registers determine which digital comparator receives the sample from the given conversion on Sample Sequence n if the corresponding  $S_nDCOP$  bit in the **ADCSSOPn** register is set. The **ADCSSDC1** register controls the selection for Sample Sequencer 1 and the **ADCSSDC2** register controls the selection for Sample Sequencer 2.

ADC Sample Sequence 1 Digital Comparator Select (ADCSSDC1)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x074  
 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.																		
15:12	S3DCSEL	R/W	0x0	<p>Sample 3 Digital Comparator Select</p> <p>When the <math>S3DCOP</math> bit in the <b>ADCSSOPn</b> register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the eighth sample from Sample Sequencer n.</p> <p><b>Note:</b> Values not listed are reserved.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr><td>0x0</td><td>Digital Comparator Unit 0 (<b>ADCDCOMP0</b> and <b>ADCCCTL0</b>)</td></tr> <tr><td>0x1</td><td>Digital Comparator Unit 1 (<b>ADCDCOMP1</b> and <b>ADCCCTL1</b>)</td></tr> <tr><td>0x2</td><td>Digital Comparator Unit 2 (<b>ADCDCOMP2</b> and <b>ADCCCTL2</b>)</td></tr> <tr><td>0x3</td><td>Digital Comparator Unit 3 (<b>ADCDCOMP3</b> and <b>ADCCCTL3</b>)</td></tr> <tr><td>0x4</td><td>Digital Comparator Unit 4 (<b>ADCDCOMP4</b> and <b>ADCCCTL4</b>)</td></tr> <tr><td>0x5</td><td>Digital Comparator Unit 5 (<b>ADCDCOMP5</b> and <b>ADCCCTL5</b>)</td></tr> <tr><td>0x6</td><td>Digital Comparator Unit 6 (<b>ADCDCOMP6</b> and <b>ADCCCTL6</b>)</td></tr> <tr><td>0x7</td><td>Digital Comparator Unit 7 (<b>ADCDCOMP7</b> and <b>ADCCCTL7</b>)</td></tr> </tbody> </table>	Value	Description	0x0	Digital Comparator Unit 0 ( <b>ADCDCOMP0</b> and <b>ADCCCTL0</b> )	0x1	Digital Comparator Unit 1 ( <b>ADCDCOMP1</b> and <b>ADCCCTL1</b> )	0x2	Digital Comparator Unit 2 ( <b>ADCDCOMP2</b> and <b>ADCCCTL2</b> )	0x3	Digital Comparator Unit 3 ( <b>ADCDCOMP3</b> and <b>ADCCCTL3</b> )	0x4	Digital Comparator Unit 4 ( <b>ADCDCOMP4</b> and <b>ADCCCTL4</b> )	0x5	Digital Comparator Unit 5 ( <b>ADCDCOMP5</b> and <b>ADCCCTL5</b> )	0x6	Digital Comparator Unit 6 ( <b>ADCDCOMP6</b> and <b>ADCCCTL6</b> )	0x7	Digital Comparator Unit 7 ( <b>ADCDCOMP7</b> and <b>ADCCCTL7</b> )
Value	Description																					
0x0	Digital Comparator Unit 0 ( <b>ADCDCOMP0</b> and <b>ADCCCTL0</b> )																					
0x1	Digital Comparator Unit 1 ( <b>ADCDCOMP1</b> and <b>ADCCCTL1</b> )																					
0x2	Digital Comparator Unit 2 ( <b>ADCDCOMP2</b> and <b>ADCCCTL2</b> )																					
0x3	Digital Comparator Unit 3 ( <b>ADCDCOMP3</b> and <b>ADCCCTL3</b> )																					
0x4	Digital Comparator Unit 4 ( <b>ADCDCOMP4</b> and <b>ADCCCTL4</b> )																					
0x5	Digital Comparator Unit 5 ( <b>ADCDCOMP5</b> and <b>ADCCCTL5</b> )																					
0x6	Digital Comparator Unit 6 ( <b>ADCDCOMP6</b> and <b>ADCCCTL6</b> )																					
0x7	Digital Comparator Unit 7 ( <b>ADCDCOMP7</b> and <b>ADCCCTL7</b> )																					
11:8	S2DCSEL	R/W	0x0	<p>Sample 2 Digital Comparator Select</p> <p>This field has the same encodings as <b>S3DCSEL</b> but is used during the third sample.</p>																		

Bit/Field	Name	Type	Reset	Description
7:4	S1DCSEL	R/W	0x0	Sample 1 Digital Comparator Select This field has the same encodings as S3DCSEL but is used during the second sample.
3:0	S0DCSEL	R/W	0x0	Sample 0 Digital Comparator Select This field has the same encodings as S3DCSEL but is used during the first sample.



### Register 33: ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0

This register defines the analog input configuration for the sample executed with Sample Sequencer 3. This register is 4 bits wide and contains information for one possible sample. See the **ADCSSMUX0** register on page 483 for detailed bit descriptions.

#### ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x0A0  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved												MUX0			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	Reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	MUX0	R/W	0	1st Sample Input Select

### Register 34: ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4

This register contains the configuration information for a sample executed with Sample Sequencer 3. The `END0` bit is always set as this sequencer can execute only one sample. This register is 4 bits wide and contains information for one possible sample. See the `ADCSSCTL0` register on page 485 for detailed bit descriptions.

#### ADC Sample Sequence Control 3 (ADCSSCTL3)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x0A4  
 Type R/W, reset 0x0000.0002

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	Reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	Reserved												TS0	IE0	END0	D0	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Bit/Field	Name	Type	Reset	Description
31:4	Reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as <code>TS7</code> but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as <code>IE7</code> but used during the first sample.
1	END0	R/W	1	1st Sample is End of Sequence Same definition as <code>END7</code> but used during the first sample. Because 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 35: ADC Sample Sequence 3 Operation (ADCSSOP3), offset 0x0B0

This register determines whether the sample from the given conversion on Sample Sequence 3 is saved in the Sample Sequence 3 FIFO or sent to the digital comparator unit.

### ADC Sample Sequence 3 Operation (ADCSSOP3)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x0B0  
 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															S0DCOP
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

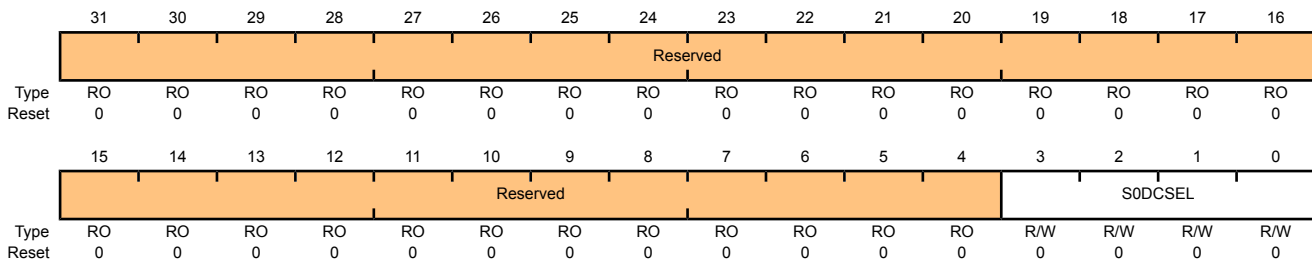
Bit/Field	Name	Type	Reset	Description
31:1	Reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	S0DCOP	R/W	0	<p>Sample 0 Digital Comparator Operation</p> <p>When this bit is set, the sample is sent to the digital comparator unit specified by the S0DCSEL bit in the ADCSSDC3 register, and the value is not written to the FIFO.</p> <p>When this bit is clear, the sample is saved in Sample Sequence FIFO3.</p>

### Register 36: ADC Sample Sequence 3 Digital Comparator Select (ADCSSDC3), offset 0x0B4

This register determines which digital comparator receives the sample from the given conversion on Sample Sequence 3 if the corresponding  $S_{nDCOP}$  bit in the **ADCSSOP3** register is set.

#### ADC Sample Sequence 3 Digital Comparator Select (ADCSSDC3)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0x0B4  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	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	S0DCSEL	R/W	0x0	Sample 0 Digital Comparator Select

When the  $S_{0DCOP}$  bit in the **ADCSSOP3** register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the sample from Sample Sequencer 3.

**Note:** Values not listed are reserved.

Value	Description
0x0	Digital Comparator Unit 0 ( <b>ADCDCOMP0</b> and <b>ADCCCTL0</b> )
0x1	Digital Comparator Unit 1 ( <b>ADCDCOMP1</b> and <b>ADCCCTL1</b> )
0x2	Digital Comparator Unit 2 ( <b>ADCDCOMP2</b> and <b>ADCCCTL2</b> )
0x3	Digital Comparator Unit 3 ( <b>ADCDCOMP3</b> and <b>ADCCCTL3</b> )
0x4	Digital Comparator Unit 4 ( <b>ADCDCOMP4</b> and <b>ADCCCTL4</b> )
0x5	Digital Comparator Unit 5 ( <b>ADCDCOMP5</b> and <b>ADCCCTL5</b> )
0x6	Digital Comparator Unit 6 ( <b>ADCDCOMP6</b> and <b>ADCCCTL6</b> )
0x7	Digital Comparator Unit 7 ( <b>ADCDCOMP7</b> and <b>ADCCCTL7</b> )

## Register 37: ADC Digital Comparator Reset Initial Conditions (ADCDCRIC), offset 0xD00

This register provides the ability to reset any of the digital comparator interrupt or trigger functions back to their initial conditions. Resetting these functions ensures that the data that is being used by the interrupt and trigger functions in the digital comparator unit is not stale.

### ADC Digital Comparator Reset Initial Conditions (ADCDCRIC)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0xD00  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Reserved								DCTRIG7	DCTRIG6	DCTRIG5	DCTRIG4	DCTRIG3	DCTRIG2	DCTRIG1	DCTRIG0
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								DCINT7	DCINT6	DCINT5	DCINT4	DCINT3	DCINT2	DCINT1	DCINT0
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:24	Reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23	DCTRIG7	R/W	0	Digital Comparator Trigger 7  When this bit is set, the Digital Comparator 7 trigger unit is reset to its initial conditions.  Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
22	DCTRIG6	R/W	0	Digital Comparator Trigger 6  When this bit is set, the Digital Comparator 6 trigger unit is reset to its initial conditions.  Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
21	DCTRIG5	R/W	0	Digital Comparator Trigger 5  When this bit is set, the Digital Comparator 5 trigger unit is reset to its initial conditions.  Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.

Bit/Field	Name	Type	Reset	Description
20	DCTRIG4	R/W	0	Digital Comparator Trigger 4  When this bit is set, the Digital Comparator 4 trigger unit is reset to its initial conditions.  Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
19	DCTRIG3	R/W	0	Digital Comparator Trigger 3  When this bit is set, the Digital Comparator 3 trigger unit is reset to its initial conditions.  Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
18	DCTRIG2	R/W	0	Digital Comparator Trigger 2  When this bit is set, the Digital Comparator 2 trigger unit is reset to its initial conditions.  Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
17	DCTRIG1	R/W	0	Digital Comparator Trigger 1  When this bit is set, the Digital Comparator 1 trigger unit is reset to its initial conditions.  Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
16	DCTRIG0	R/W	0	Digital Comparator Trigger 0  When this bit is set, the Digital Comparator 0 trigger unit is reset to its initial conditions.  Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
15:8	Reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	DCINT7	R/W	0	Digital Comparator Trigger 7  When this bit is set, the Digital Comparator 7 interrupt unit is reset to its initial conditions.  Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.

Bit/Field	Name	Type	Reset	Description
6	DCINT6	R/W	0	<p>Digital Comparator Trigger 6</p> <p>When this bit is set, the Digital Comparator 6 interrupt unit is reset to its initial conditions.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>
5	DCINT5	R/W	0	<p>Digital Comparator Trigger 5</p> <p>When this bit is set, the Digital Comparator 5 interrupt unit is reset to its initial conditions.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>
4	DCINT4	R/W	0	<p>Digital Comparator Trigger 4</p> <p>When this bit is set, the Digital Comparator 4 interrupt unit is reset to its initial conditions.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>
3	DCINT3	R/W	0	<p>Digital Comparator Trigger 3</p> <p>When this bit is set, the Digital Comparator 3 interrupt unit is reset to its initial conditions.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>
2	DCINT2	R/W	0	<p>Digital Comparator Trigger 2</p> <p>When this bit is set, the Digital Comparator 2 interrupt unit is reset to its initial conditions.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>
1	DCINT1	R/W	0	<p>Digital Comparator Trigger 1</p> <p>When this bit is set, the Digital Comparator 1 interrupt unit is reset to its initial conditions.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>

Bit/Field	Name	Type	Reset	Description
0	DCINT0	R/W	0	<p>Digital Comparator Trigger 0</p> <p>When this bit is set, the Digital Comparator 0 interrupt unit is reset to its initial conditions.</p> <p>Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.</p>



**Register 38: ADC Digital Comparator Control 0 (ADCDCCTL0), offset 0xE00**

**Register 39: ADC Digital Comparator Control 1 (ADCDCCTL1), offset 0xE04**

**Register 40: ADC Digital Comparator Control 2 (ADCDCCTL2), offset 0xE08**

**Register 41: ADC Digital Comparator Control 3 (ADCDCCTL3), offset 0xE0C**

**Register 42: ADC Digital Comparator Control 4 (ADCDCCTL4), offset 0xE10**

**Register 43: ADC Digital Comparator Control 5 (ADCDCCTL5), offset 0xE14**

**Register 44: ADC Digital Comparator Control 6 (ADCDCCTL6), offset 0xE18**

**Register 45: ADC Digital Comparator Control 7 (ADCDCCTL7), offset 0xE1C**

This register provides the comparison encodings that generate an interrupt or PWM trigger.

#### ADC Digital Comparator Control 0 (ADCDCCTL0)

ADC0 base: 0x4003.8000

ADC1 base: 0x4003.9000

Offset 0xE00

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			CTE	CTC			CTM			Reserved			CIE	CIC		CIM
Type	RO	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	
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	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	CTE	R/W	0	<p>Comparison Trigger Enable</p> <p>When set, this bit enables the trigger function state machine. The ADC conversion data is used to determine if a trigger should be generated according to the programming of the CTC and CTM fields.</p> <p>When clear, this bit disables the trigger function state machine. ADC conversion data is ignored by the trigger function.</p>

Bit/Field	Name	Type	Reset	Description
11:10	CTC	R/W	0x0	<p>Comparison Trigger Condition</p> <p>This field specifies the operational region in which a trigger is generated when the ADC conversion data is compared against the values of <code>COMP0</code> and <code>COMP1</code>. The <code>COMP0</code> and <code>COMP1</code> fields are defined in the <b>ADCDCMPx</b> registers.</p> <p>Value Mode</p> <p>0x0 Low Band  <math>ADC\ Data &lt; COMP0\ and &lt; COMP1</math></p> <p>0x1 Mid Band  <math>COMP0 \leq ADC\ Data &lt; COMP1</math></p> <p>0x2 Reserved</p> <p>0x3 High Band  <math>COMP0 \leq COMP1 \leq ADC\ Data</math></p>
9:8	CTM	R/W	0x0	<p>Comparison Trigger Mode</p> <p>This field specifies the mode by which the trigger comparison is made.</p> <p>Value Mode</p> <p>0x0 Always  This mode generates a trigger every time the ADC conversion data falls within the selected operational region.</p> <p>0x1 Once  This mode generates a trigger the first time that the ADC conversion data enters the selected operational region.</p> <p>0x2 Hysteresis Always  This mode generates a trigger when the ADC conversion data falls within the selected operational region and continues to generate the trigger until the hysteresis condition is cleared by entering the opposite operational region.  Note that the hysteresis modes are only defined for <code>CTC</code> encodings of 0x0 and 0x3.</p> <p>0x3 Hysteresis Once  This mode generates a trigger the first time that the ADC conversion data falls within the selected operational region. No additional triggers are generated until the hysteresis condition is cleared by entering the opposite operational region.  Note that the hysteresis modes are only defined for <code>CTC</code> encodings of 0x0 and 0x3.</p>
7:5	Reserved	RO	0x0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>

Bit/Field	Name	Type	Reset	Description
4	CIE	R/W	0	<p>Comparison Interrupt Enable</p> <p>When set, this bit enables the comparison interrupt. The ADC conversion data is used to determine if an interrupt should be generated according to the programming of the C<sub>IC</sub> and C<sub>IM</sub> fields.</p> <p>When clear, this bit disables the comparison interrupt. ADC conversion data has no effect on interrupt generation.</p>
3:2	CIC	R/W	0x0	<p>Comparison Interrupt Condition</p> <p>This field specifies the operational region in which an interrupt is generated when the ADC conversion data is compared against the values of COMP<sub>0</sub> and COMP<sub>1</sub>. The COMP<sub>0</sub> and COMP<sub>1</sub> fields are defined in the ADC<sub>CCMPx</sub> registers.</p> <p>Value Mode</p> <p>0x0 Low Band ADC Data &lt; COMP<sub>0</sub> and &lt; COMP<sub>1</sub></p> <p>0x1 Mid Band COMP<sub>0</sub> ≤ ADC Data &lt; COMP<sub>1</sub></p> <p>0x2 Reserved</p> <p>0x3 High Band COMP<sub>0</sub> &lt; COMP<sub>1</sub> ≤ ADC Data</p>
1:0	CIM	R/W	0x0	<p>Comparison Interrupt Mode</p> <p>This field specifies the mode by which the interrupt comparison is made.</p> <p>Value Mode</p> <p>0x0 Always This mode generates an interrupt every time the ADC conversion data falls within the selected operational region.</p> <p>0x1 Once This mode generates an interrupt the first time that the ADC conversion data enters the selected operational region.</p> <p>0x2 Hysteresis Always This mode generates an interrupt when the ADC conversion data falls within the selected operational region and continues to generate the interrupt until the hysteresis condition is cleared by entering the opposite operational region. Note that the hysteresis modes are only defined for CTC encodings of 0x0 and 0x3.</p> <p>0x3 Hysteresis Once This mode generates an interrupt the first time that the ADC conversion data falls within the selected operational region. No additional interrupts are generated until the hysteresis condition is cleared by entering the opposite operational region. Note that the hysteresis modes are only defined for CTC encodings of 0x0 and 0x3.</p>

**Register 46: ADC Digital Comparator Range 0 (ADCDCCMP0), offset 0xE40**

**Register 47: ADC Digital Comparator Range 1 (ADCDCCMP1), offset 0xE44**

**Register 48: ADC Digital Comparator Range 2 (ADCDCCMP2), offset 0xE48**

**Register 49: ADC Digital Comparator Range 3 (ADCDCCMP3), offset 0xE4C**

**Register 50: ADC Digital Comparator Range 4 (ADCDCCMP4), offset 0xE50**

**Register 51: ADC Digital Comparator Range 5 (ADCDCCMP5), offset 0xE54**

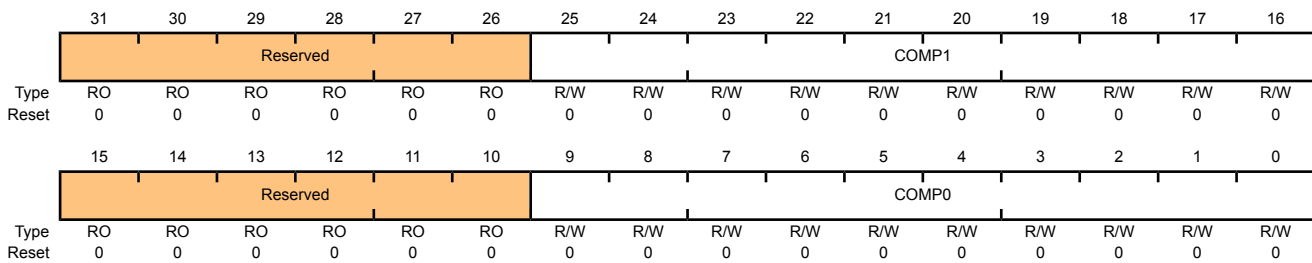
**Register 52: ADC Digital Comparator Range 6 (ADCDCCMP6), offset 0xE58**

**Register 53: ADC Digital Comparator Range 7 (ADCDCCMP7), offset 0xE5C**

This register defines the comparison values that are used to determine if the ADC conversion data falls in the appropriate operating region. Note that the value in the COMP1 field must be greater than or equal to the value in the COMP0 field or unexpected results can occur.

ADC Digital Comparator Range 0 (ADCDCCMP0)

ADC0 base: 0x4003.8000  
 ADC1 base: 0x4003.9000  
 Offset 0xE40  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:26	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.
25:16	COMP1	R/W	0x000	Compare 1  The value in this field is compared against the ADC conversion data. The result of the comparison is used to determine if the data lies within the high-band region.  Note that the value of COMP1 must be greater than or equal to the value of COMP0.
15:10	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.
9:0	COMP0	R/W	0x000	Compare 0  The value in this field is compared against the ADC conversion data. The result of the comparison is used to determine if the data lies within the low-band region.

## 14 Universal Asynchronous Receivers/Transmitters (UARTs)

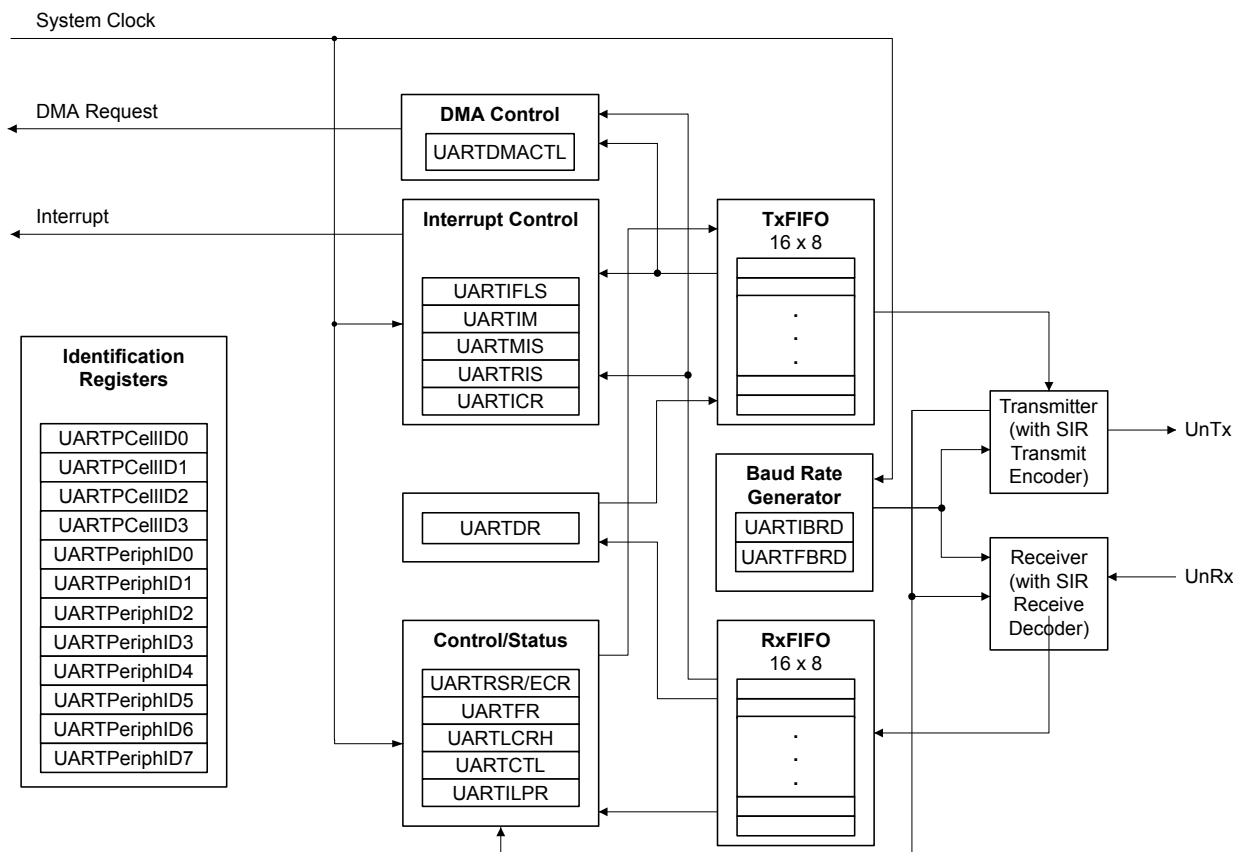
Each Stellaris® Universal Asynchronous Receiver/Transmitter (UART) has the following features:

- Programmable baud-rate generator allowing speeds up to 5 Mbps for regular speed (divide by 16) and 10 Mbps for high speed (divide by 8)
- Separate 16x8 transmit (TX) and receive (RX) FIFOs to reduce CPU interrupt service loading
- LIN protocol support
- 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
- 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
- Support for communication with ISO 7816 smart cards
- Full modem handshake support (on UART1)
- Standard FIFO-level and End-of-Transmission interrupts
- Efficient transfers using Micro Direct Memory Access Controller ( $\mu$ DMA)
  - Separate channels for transmit and receive
  - Receive single request asserted when data is in the FIFO; burst request asserted at programmed FIFO level

- Transmit single request asserted when there is space in the FIFO; burst request asserted at programmed FIFO level

## 14.1 Block Diagram

Figure 14-1. UART Module Block Diagram



## 14.2 Functional Description

Each Stellaris<sup>®</sup> UART performs the functions of parallel-to-serial and serial-to-parallel conversions. It is similar in functionality to a 16C550 UART, but is not register compatible.

The UART is configured for transmit and/or receive via the TXE and RXE bits of the **UART Control (UARTCTL)** register (see page 534). Transmit and receive are both enabled out of reset. Before any control registers are programmed, the UART must be disabled by clearing the UARTEEN bit in **UARTCTL**. If the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

The UART peripheral also includes a serial IR (SIR) encoder/decoder block that can be connected to an infrared transceiver to implement an IrDA SIR physical layer. The SIR function is programmed using the **UARTCTL** register.

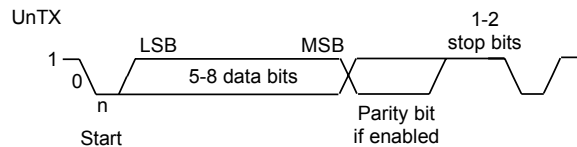
### 14.2.1 Transmit/Receive Logic

The transmit logic performs parallel-to-serial conversion on the data read from the transmit FIFO. The control logic outputs the serial bit stream beginning with a start bit, and followed by the data

bits (LSB first), parity bit, and the stop bits according to the programmed configuration in the control registers. See Figure 14-2 on page 515 for details.

The receive logic performs serial-to-parallel conversion on the received bit stream after a valid start pulse has been detected. Overrun, parity, frame error checking, and line-break detection are also performed, and their status accompanies the data that is written to the receive FIFO.

**Figure 14-2. UART Character Frame**



## 14.2.2 Baud-Rate Generation

The baud-rate divisor is a 22-bit number consisting of a 16-bit integer and a 6-bit fractional part. The number formed by these two values is used by the baud-rate generator to determine the bit period. Having a fractional baud-rate divider allows the UART to generate all the standard baud rates.

The 16-bit integer is loaded through the **UART Integer Baud-Rate Divisor (UARTIBRD)** register (see page 530) and the 6-bit fractional part is loaded with the **UART Fractional Baud-Rate Divisor (UARTFBRD)** register (see page 531). The baud-rate divisor (BRD) has the following relationship to the system clock (where *BRDI* is the integer part of the BRD and *BRDF* is the fractional part, separated by a decimal place.)

$$BRD = BRDI + BRDF = \text{UARTSysClk} / (\text{ClkDiv} * \text{Baud Rate})$$

where *UARTSysClk* is the system clock connected to the UART, and *ClkDiv* is either 16 (if *HSE* in **UARTCTL** is clear) or 8 (if *HSE* is set).

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 8x or 16x the baud-rate (referred to as *Baud8* and *Baud16*, depending upon the setting of the *HSE* bit (bit 5) in **UARTCTL**). This reference clock is divided by 8 or 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 532), the **UARTIBRD** and **UARTFBRD** registers form an internal 30-bit register. This internal register is only updated when a write operation to **UARTLCRH** is performed, so any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register for the changes to take effect.

To update the baud-rate registers, there are four possible sequences:

- **UARTIBRD** write, **UARTFBRD** write, and **UARTLCRH** write
- **UARTFBRD** write, **UARTIBRD** write, and **UARTLCRH** write
- **UARTIBRD** write and **UARTLCRH** write

- **UARTFBRD** write and **UARTLCRH** write

### 14.2.3 Data Transmission

Data received or transmitted is stored in two 16-byte FIFOs, though the receive FIFO has an extra four bits per character for status information. For transmission, data is written into the transmit FIFO. If the UART is enabled, it causes a data frame to start transmitting with the parameters indicated in the **UARTLCRH** register. Data continues to be transmitted until there is no data left in the transmit FIFO. The **BUSY** bit in the **UART Flag (UARTFR)** register (see page 527) 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** or fourth cycle of **Baud8** depending upon the setting of the **HSE** bit (bit 5) in **UARTCTL** (described in “Transmit/Receive Logic” on page 514).

The start bit is valid if **UnRx** is still low on the eighth cycle of **Baud16** (**HSE** clear) or the fourth cycle of **Baud 8** (**HSE** set), otherwise a false start bit is detected and it is ignored. Start bit errors can be viewed in the **UART Receive Status (UARTSR)** register (see page 525). If the start bit was valid, successive data bits are sampled on every 16th cycle of **Baud16** or 8th cycle of **Baud8** (that is, one bit period later) according to the programmed length of the data characters and value of the **HSE** bit in **UARTCTL**. The parity bit is then checked if parity mode was enabled. Data length and parity are defined in the **UARTLCRH** register.

Lastly, a valid stop bit is confirmed if **UnRx** is High, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO, with any error bits associated with that word.

### 14.2.4 Serial IR (SIR)

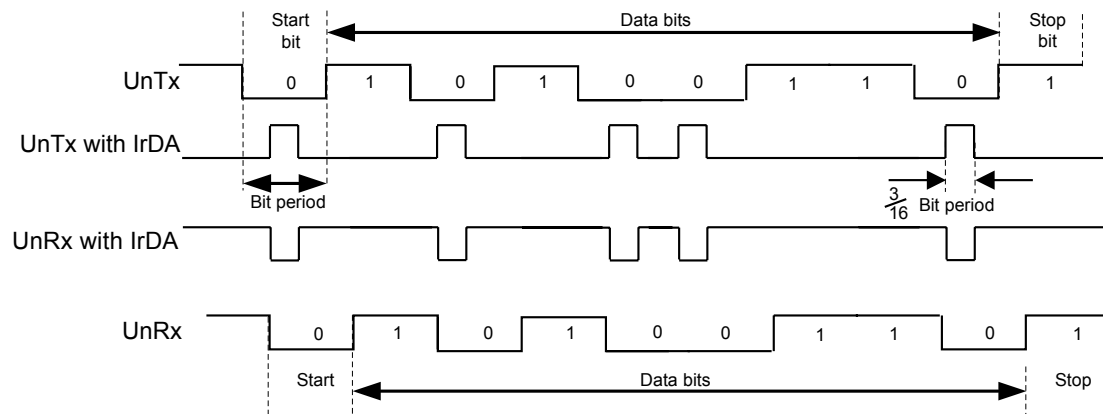
The UART peripheral includes an IrDA serial-IR (SIR) encoder/decoder block. The IrDA SIR block provides functionality that converts between an asynchronous UART data stream, and half-duplex serial SIR interface. No analog processing is performed on-chip. The role of the SIR block is to provide a digital encoded output, and decoded input to the UART. The UART signal pins can be connected to an infrared transceiver to implement an IrDA SIR physical layer link. The SIR block has two modes of operation:

- In normal IrDA mode, a zero logic level is transmitted as high pulse of 3/16th duration of the selected baud rate bit period on the output pin, while logic one levels are transmitted as a static LOW signal. These levels control the driver of an infrared transmitter, sending a pulse of light for each zero. On the reception side, the incoming light pulses energize the photo transistor base of the receiver, pulling its output LOW. This drives the UART input pin LOW.
- In low-power IrDA mode, the width of the transmitted infrared pulse is set to three times the period of the internally generated **IrLPBaud16** signal (1.63  $\mu$ s, assuming a nominal 1.8432 MHz frequency) by changing the appropriate bit in the **UARTCR** register. See page 529 for more information on IrDA low-power pulse-duration configuration.

Figure 14-3 on page 517 shows the UART transmit and receive signals, with and without IrDA modulation.



Figure 14-3. IrDA Data Modulation



In both normal and low-power IrDA modes:

- During transmission, the UART data bit is used as the base for encoding
- During reception, the decoded bits are transferred to the UART receive logic

The IrDA SIR physical layer specifies a half-duplex communication link, with a minimum 10 ms delay between transmission and reception. This delay must be generated by software because it is not automatically supported by the UART. The delay is required because the infrared receiver electronics might become biased, or even saturated from the optical power coupled from the adjacent transmitter LED. This delay is known as latency, or receiver setup time.

### 14.2.5 ISO 7816 Support

The UART offers basic support to allow communication with an ISO 7816 smartcard. When bit 3 (**SMART**) of the **UARTCTL** register is set, the **UnTX** line is used as a bit clock, and **UnRX** is used as the half-duplex communication line connected to the smartcard.

When using ISO 7816 mode, the **UARTLCRH** register must be set to transmit 8-bit words (**WLEN** bits 6:5 set to 0x3) with EVEN parity (**PEN** set to 1 and **EPS** set to 1). In this mode, the UART is automatically set to use 2 stop bits, and the **STP2** bit of the **UARTLCRH** register is ignored.

If a parity error is detected during transmission, the data line (**UARTRXD**) will be pulled Low during the second stop bit. This will trigger the UART to abort the transmission, flushing the transmit FIFO and discarding any data it contains, and raise a parity error interrupt, allowing software to detect the problem and initiate retransmission of the affected data. Note that the UART does not support automatic retransmission in this case.

### 14.2.6 LIN Support

The UART module offers hardware support for the LIN protocol as either a master or a slave. The LIN mode is enabled by setting the **LIN** bit in the **UARTCTL** register. A LIN message is identified by the use of a Sync Break at the beginning of the message. The Sync Break is a transmission of a series of 0s. The Sync Break is followed by the Sync data field (0x55).

The UART should be configured as followed to operate in LIN mode:

1. Configure the UART for 1 start bit, 8 data bits, no parity, and 1 stop bit. Enable the Transmit FIFO.

2. Set the `LIN` bit in the `UARTCTL` register.

When preparing to send a LIN message, the TXFIFO should contain the Sync data (0x55) at FIFO location 0, the Identifier data at location 1, the data to be transmitted, and the checksum in the final FIFO entry.

#### 14.2.6.1 LIN Master

The UART is enabled to be the LIN master by setting the `MASTER` bit in the `UARTLCTL` register. The length of the Sync Break is programmable using the `BLEN` field in the `UARTLCTL` register and can be 13-16 bits (baud clock cycles).

#### 14.2.6.2 LIN Slave

The LIN UART slave is required to adjust its baud rate to that of the LIN master. In slave mode, the LIN UART recognizes the Sync Break, which must be at least 13 bits in duration. A timer is provided to capture timing data on the 1st and 5th falling edges of the Sync field so that the baud rate can be adjusted to match the master.

After detecting a Sync Break, the UART waits for the synchronization field. The first falling edge generates an interrupt using the `LME1RIS` bit in the `UARTRIS` register, and the timer value is captured and stored in the `UARTLSS` register (T1). On the fifth falling edge, a second interrupt is generated using the `LME5RIS` bit in the `UARTRIS` register, and the timer value is captured again (T2). The actual baud rate can be calculated using  $(T2-T1)/8$ , and the local baud rate should be adjusted as needed.

#### 14.2.7 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 523). 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 532).

FIFO status can be monitored via the **UART Flag (UARTFR)** register (see page 527) 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 537). 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.

#### 14.2.8 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, or, if the `EOT` bit in **UARTCTRL** is set, when the last bit of all transmitted data leaves the serializer)
- 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 543).

The interrupt events that can trigger a controller-level interrupt are defined in the **UART Interrupt Mask (UARTIM)** register (see page 539) 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 541).

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 545).

The receive timeout interrupt is asserted when the receive FIFO is not empty, and no further data is received over a 32-bit period. The receive timeout interrupt is cleared either when the FIFO becomes empty through reading all the data (or by reading the holding register), or when a 1 is written to the corresponding bit in the **UARTICR** register.

### 14.2.9 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 534). In loopback mode, data transmitted on UnTx is received on the UnRx input.

### 14.2.10 DMA Operation

The UART provides an interface connected to the  $\mu$ DMA controller. The DMA operation of the UART is enabled through the **UART DMA Control (UARTDMACTL)** register. When DMA operation is enabled, the UART will assert a DMA request on the receive or transmit channel when the associated FIFO can transfer data. For the receive channel, a single transfer request is asserted whenever there is any data in the receive FIFO. A burst transfer request is asserted whenever the amount of data in the receive FIFO is at or above the FIFO trigger level. For the transmit channel, a single transfer request is asserted whenever there is at least one empty location in the transmit FIFO. The burst request is asserted whenever the transmit FIFO contains fewer characters than the FIFO trigger level. The single and burst DMA transfer requests are handled automatically by the  $\mu$ DMA controller depending how the DMA channel is configured.

To enable DMA operation for the receive channel, the `RXDMAE` bit of the **DMA Control (UARTDMACTL)** register should be set. To enable DMA operation for the transmit channel, the `TXDMAE` bit of **UARTDMACTL** should be set. The UART can also be configured to stop using DMA for the receive channel if a receive error occurs. If the `DMAERR` bit of **UARTDMACR** is set, then when a receive error occurs, the DMA receive requests will be automatically disabled. This error condition can be cleared by clearing the UART error interrupt.

If DMA is enabled, then the  $\mu$ DMA controller will trigger an interrupt when a transfer is complete. The interrupt will occur on the UART interrupt vector. Therefore, if interrupts are used for UART

operation and DMA is enabled, the UART interrupt handler must be designed to handle the  $\mu$ DMA completion interrupt.

See “Micro Direct Memory Access ( $\mu$ DMA)” on page 219 for more details about programming the  $\mu$ DMA controller.

#### 14.2.11 IrDA SIR block

The IrDA SIR block contains an IrDA serial IR (SIR) protocol encoder/decoder. When enabled, the SIR block uses the `UnTx` and `UnRx` pins for the SIR protocol, which should be connected to an IR transceiver.

The SIR block can receive and transmit, but it is only half-duplex so it cannot do both at the same time. Transmission must be stopped before data can be received. The IrDA SIR physical layer specifies a minimum 10-ms delay between transmission and reception.

### 14.3 Initialization and Configuration

To use the UARTs, the peripheral clock must be enabled by setting the `UART0`, `UART1`, or `UART2` bits in the **RCGC1** register. See page 163. In addition, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register in the System Control module. See page 175. To find out which GPIO port to enable, refer to Table 24-5 on page 937.

This section discusses the steps that are required to use a UART module. For this example, the UART clock is assumed to be 20 MHz and the desired UART configuration is:

- 115200 baud rate
- Data length of 8 bits
- One stop bit
- No parity
- FIFOs disabled
- No interrupts

The first thing to consider when programming the UART is the baud-rate divisor (BRD), since the **UARTIBRD** and **UARTFBRD** registers must be written before the **UARTLCRH** register. Using the equation described in “Baud-Rate Generation” on page 515, 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 530) should be set to 10. The value to be loaded into the **UARTFBRD** register (see page 531) is calculated by the equation:

$$\text{UARTFBRD}[\text{DIVFRAC}] = \text{integer}(0.8507 * 64 + 0.5) = 54$$

With the BRD values in hand, the UART configuration is written to the module in the following order:

1. Disable the UART by clearing the `UARTEN` bit in the **UARTCTL** register.
2. Write the integer portion of the BRD to the **UARTIBRD** register.
3. Write the fractional portion of the BRD to the **UARTFBRD** register.

4. Write the desired serial parameters to the **UARTLCRH** register (in this case, a value of 0x0000.0060).
5. Optionally, configure the uDMA channel (see “Micro Direct Memory Access (μDMA)” on page 219) and enable the DMA option(s) in the **UARTDMACTL** register.
6. Enable the UART by setting the **UARTEN** bit in the **UARTCTL** register.

## 14.4 Register Map

Table 14-1 on page 521 lists the UART registers. The offset listed is a hexadecimal increment to the register’s address, relative to that UART’s base address:

- UART0: 0x4000.C000
- UART1: 0x4000.D000
- UART2: 0x4000.E000

Note that the UART module clock must be enabled before the registers can be programmed (see page 163).

**Note:** The UART must be disabled (see the **UARTEN** bit in the **UARTCTL** register on page 534) before any of the control registers are reprogrammed. When the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

**Table 14-1. UART Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	UARTDR	R/W	0x0000.0000	UART Data	523
0x004	UARTSR/UARTECR	R/W	0x0000.0000	UART Receive Status/Error Clear	525
0x018	UARTFR	RO	0x0000.0090	UART Flag	527
0x020	UARTILPR	R/W	0x0000.0000	UART IrDA Low-Power Register	529
0x024	UARTIBRD	R/W	0x0000.0000	UART Integer Baud-Rate Divisor	530
0x028	UARTFBRD	R/W	0x0000.0000	UART Fractional Baud-Rate Divisor	531
0x02C	UARTLCRH	R/W	0x0000.0000	UART Line Control	532
0x030	UARTCTL	R/W	0x0000.0300	UART Control	534
0x034	UARTIFLS	R/W	0x0000.0012	UART Interrupt FIFO Level Select	537
0x038	UARTIM	R/W	0x0000.0000	UART Interrupt Mask	539
0x03C	UARTRIS	RO	0x0000.000F	UART Raw Interrupt Status	541
0x040	UARTMIS	RO	0x0000.0000	UART Masked Interrupt Status	543
0x044	UARTICR	W1C	0x0000.0000	UART Interrupt Clear	545
0x048	UARTDMACTL	R/W	0x0000.0000	UART DMA Control	547
0x090	UARTLCTL	R/W	0x0000.0000	UART LIN Control	548
0x094	UARTLSS	RO	0x0000.0000	UART LIN Snap Shot	549
0x098	UARTLTIM	RO	0x0000.0000	UART LIN Timer	550

Offset	Name	Type	Reset	Description	See page
0xFD0	UARTPeriphID4	RO	0x0000.0000	UART Peripheral Identification 4	551
0xFD4	UARTPeriphID5	RO	0x0000.0000	UART Peripheral Identification 5	552
0xFD8	UARTPeriphID6	RO	0x0000.0000	UART Peripheral Identification 6	553
0xFDC	UARTPeriphID7	RO	0x0000.0000	UART Peripheral Identification 7	554
0xFE0	UARTPeriphID0	RO	0x0000.0060	UART Peripheral Identification 0	555
0xFE4	UARTPeriphID1	RO	0x0000.0000	UART Peripheral Identification 1	556
0xFE8	UARTPeriphID2	RO	0x0000.0018	UART Peripheral Identification 2	557
0xFEC	UARTPeriphID3	RO	0x0000.0001	UART Peripheral Identification 3	558
0xFF0	UARTPCellID0	RO	0x0000.000D	UART PrimeCell Identification 0	559
0xFF4	UARTPCellID1	RO	0x0000.00F0	UART PrimeCell Identification 1	560
0xFF8	UARTPCellID2	RO	0x0000.0005	UART PrimeCell Identification 2	561
0xFFC	UARTPCellID3	RO	0x0000.00B1	UART PrimeCell Identification 3	562

## 14.5 Register Descriptions

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

## Register 1: UART Data (UARTDR), offset 0x000

This register is the data register (the interface to the FIFOs).

When FIFOs are enabled, data written to this location is pushed onto the transmit FIFO. If FIFOs are disabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). A write to this register initiates a transmission from the UART.

For received data, if the FIFO is enabled, the data byte and the 4-bit status (break, frame, parity, and overrun) is pushed onto the 12-bit wide receive FIFO. If FIFOs are disabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO). The received data can be retrieved by reading this register.

### UART Data (UARTDR)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x000  
 Type R/W, reset 0x0000.0000

	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				OE	BE	PE	FE	DATA							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:12	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.

### Reads

#### UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x004  
 Type RO, reset 0x0000.0000

	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													OE	BE	PE	FE
Type	RO	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	<p>UART Overrun Error</p> <p>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>.</p> <p>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.</p>
2	BE	RO	0	<p>UART Break Error</p> <p>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).</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. 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.</p>

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>

**Writes**

UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x004  
 Type WO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	WO	0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>
7:0	DATA	WO	0	<p>Error Clear</p> <p>A write to this register of any data clears the framing, parity, break, and overrun flags.</p>

### Register 3: UART Flag (UARTFR), offset 0x018

The **UARTFR** register is the flag register. After reset, the **TXFF**, **RXFF**, and **BUSY** bits are 0, and **TXFE** and **RXFE** bits are 1. The **RI**, **DCD**, **DSR** and **CTS** bits indicate the modem status.

#### UART Flag (UARTFR)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0x018

Type RO, reset 0x0000.0090

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

Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	RI	RO	0	Ring Indicator  This bit is 1 if the <b>U1RI</b> signal is asserted, or 0 otherwise.
7	TXFE	RO	1	UART Transmit FIFO Empty  The meaning of this bit depends on the state of the <b>FEN</b> bit in the <b>UARTLCRH</b> register.  If the FIFO is disabled ( <b>FEN</b> is 0), this bit is set when the transmit holding register is empty.  If the FIFO is enabled ( <b>FEN</b> is 1), this bit is set when the transmit FIFO is empty.
6	RXFF	RO	0	UART Receive FIFO Full  The meaning of this bit depends on the state of the <b>FEN</b> bit in the <b>UARTLCRH</b> register.  If the FIFO is disabled, this bit is set when the receive holding register is full.  If the FIFO is enabled, this bit is set when the receive FIFO is full.
5	TXFF	RO	0	UART Transmit FIFO Full  The meaning of this bit depends on the state of the <b>FEN</b> bit in the <b>UARTLCRH</b> register.  If the FIFO is disabled, this bit is set when the transmit holding register is full.  If the FIFO is enabled, this bit is set when the transmit FIFO is full.

Bit/Field	Name	Type	Reset	Description
4	RXFE	RO	1	<p>UART Receive FIFO Empty</p> <p>The meaning of this bit depends on the state of the <code>FEN</code> 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	DCD	RO	0	<p>Data Carrier Detect</p> <p>This bit is 1 if the <code>U1DCD</code> signal is asserted, or 0 otherwise.</p>
1	DSR	RO	0	<p>Data Set Ready</p> <p>This bit is 1 if the <code>U1DSR</code> signal is asserted, or 0 otherwise.</p>
0	CTS	RO	0	<p>Clear To Send</p> <p>This bit is 1 if the <code>U1CTS</code> signal is asserted, or 0 otherwise.</p>

## Register 4: UART IrDA Low-Power Register (UARTILPR), offset 0x020

The **UARTILPR** register is an 8-bit read/write register that stores the low-power counter divisor value used to derive the low-power SIR pulse width clock by dividing down the system clock (SysClk). All the bits are cleared to 0 when reset.

The internal  $F_{IrLPBaud16}$  clock is generated by dividing down SysClk according to the low-power divisor value written to **UARTILPR**. The duration of SIR pulses generated when low-power mode is enabled is three times the period of the  $F_{IrLPBaud16}$  clock. The low-power divisor value is calculated as follows:

$$ILPDVSR = SysClk / F_{IrLPBaud16}$$

where  $F_{IrLPBaud16}$  is nominally 1.8432 MHz.

You must choose the divisor so that  $1.42 \text{ MHz} < F_{IrLPBaud16} < 2.12 \text{ MHz}$ , which results in a low-power pulse duration of 1.41–2.11  $\mu\text{s}$  (three times the period of  $F_{IrLPBaud16}$ ). The minimum frequency of  $F_{IrLPBaud16}$  ensures that pulses less than one period of  $F_{IrLPBaud16}$  are rejected, but that pulses greater than 1.4  $\mu\text{s}$  are accepted as valid pulses.

**Note:** Zero is an illegal value. Programming a zero value results in no  $F_{IrLPBaud16}$  pulses being generated.

### UART IrDA Low-Power Register (UARTILPR)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x020  
 Type R/W, reset 0x0000.0000

	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								ILPDVSR							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

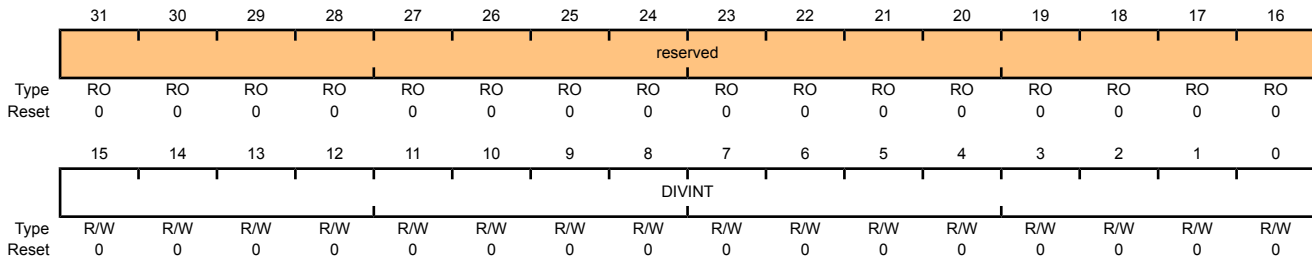
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ILPDVSR	R/W	0x00	IrDA Low-Power Divisor  This is an 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 515 for configuration details.

#### UART Integer Baud-Rate Divisor (UARTIBRD)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x024  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DIVINT	R/W	0x0000	Integer Baud-Rate Divisor

## Register 6: UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028

The **UARTFBRD** register is the fractional part of the baud-rate divisor value. All the bits are cleared on reset. When changing the **UARTFBRD** register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register. See “Baud-Rate Generation” on page 515 for configuration details.

### UART Fractional Baud-Rate Divisor (UARTFBRD)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x028  
 Type R/W, reset 0x0000.0000

	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											DIVFRAC				
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	DIVFRAC	R/W	0x000	Fractional Baud-Rate Divisor

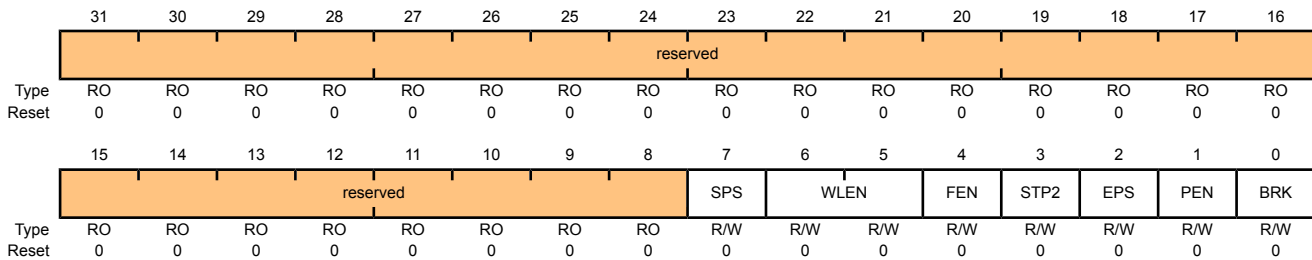
### Register 7: UART Line Control (UARTLCRH), offset 0x02C

The **UARTLCRH** register is the line control register. Serial parameters such as data length, parity, and stop bit selection are implemented in this register.

When updating the baud-rate divisor (**UARTIBRD** and/or **UARTIFRD**), the **UARTLCRH** register must also be written. The write strobe for the baud-rate divisor registers is tied to the **UARTLCRH** register.

#### UART Line Control (UARTLCRH)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x02C  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	SPS	R/W	0	UART Stick Parity Select  When bits 1, 2, and 7 of <b>UARTLCRH</b> are set, the parity bit is transmitted and checked as a 0. When bits 1 and 7 are set and 2 is cleared, the parity bit is transmitted and checked as a 1.  When this bit is cleared, stick parity is disabled.
6:5	WLEN	R/W	0	UART Word Length  The bits indicate the number of data bits transmitted or received in a frame as follows:  Value Description 0x3 8 bits 0x2 7 bits 0x1 6 bits 0x0 5 bits (default)
4	FEN	R/W	0	UART Enable FIFOs  If this bit is set to 1, transmit and receive FIFO buffers are enabled (FIFO mode).  When cleared to 0, FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers.



Bit/Field	Name	Type	Reset	Description
3	STP2	R/W	0	<p>UART Two Stop Bits Select</p> <p>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. The setting of this bit is ignored if the <code>SMART</code> bit is set in the <code>UARTCTL</code> register. When in 7816 smartcard mode, the number of stop bits is forced to 2.</p>
2	EPS	R/W	0	<p>UART Even Parity Select</p> <p>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.</p> <p>When cleared to 0, then odd parity is performed, which checks for an odd number of 1s.</p> <p>This bit has no effect when parity is disabled by the <code>PEN</code> bit.</p>
1	PEN	R/W	0	<p>UART Parity Enable</p> <p>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.</p>
0	BRK	R/W	0	<p>UART Send Break</p> <p>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.</p>

### Register 8: UART Control (UARTCTL), offset 0x030

The **UARTCTL** register is the control register. All the bits are cleared on reset except for the Transmit Enable (TXE) and Receive Enable (RXE) bits, which are set to 1.

To enable the UART module, the **UARTEN** bit must be set to 1. If software requires a configuration change in the module, the **UARTEN** bit must be cleared before the configuration changes are written. If the UART is disabled during a transmit or receive operation, the current transaction is completed prior to the UART stopping.

**Note:** The **UARTCTL** register should not be changed while the UART is enabled or else the results are unpredictable. The following sequence is recommended for making changes to the **UARTCTL** register.

1. Disable the UART.
2. Wait for the end of transmission or reception of the current character.
3. Flush the transmit FIFO by disabling bit 4 (**FEN**) in the line control register (**UARTLCRH**).
4. Reprogram the control register.
5. Enable the UART.

#### UART Control (UARTCTL)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x030  
 Type R/W, reset 0x0000.0300

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CTSEN	RTSEN	reserved	RTS	DTR	RXE	TXE	LBE	reservedLIN	HSE	EOT	SMART	SIRLP	SIREN	UARTEN	
Type	R/W	R/W	RO	RO	R/W	R/W	R/W	R/W	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	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	CTSEN	R/W	0	Enable Clear To Send  If this bit is set to 1, CTS hardware flow control is enabled. Data is only transmitted when the <b>U1CTS</b> signal is asserted.
14	RTSEN	R/W	0	Enable Request to Send  If this bit is set to 1, RTS hardware flow control is enabled. Data is only requested (by asserting <b>U1RTS</b> ) when there is space in the receive FIFO for it to be stored.

Bit/Field	Name	Type	Reset	Description
13:12	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	RTS	R/W	0	Request to Send  This bit sets the state of the U1RTS output. If RTSEN is set to 1, the UART controls the state of U1RTS automatically and this bit is ignored. When flow control is selected, this bit becomes read-only. Read it to determine the state of the RTS handshake that is being controlled by the hardware.
10	DTR	R/W	0	Data Terminal Ready  This bit sets the state of the U1DTR output.
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 UARTEN 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 UARTEN bit must also be set.
7	LBE	R/W	0	UART Loop Back Enable  If this bit is set to 1, the U <sub>n</sub> TX path is fed through the U <sub>n</sub> RX path.
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.
6	LIN	R/W	0	LIN Mode Enable  If set, the UART operates in LIN mode.
5	HSE	R/W	0	High-Speed Enable  If set, the UART is clocked using the system clock divided by 8. If clear, the system clock divided by 16 is used.  <b>Note:</b> System clock used is also dependent on the baud-rate divisor configuration (see page 530) and page 531).
4	EOT	R/W	0	End of Transmission  This bit determines the behavior of the TXRIS bit in the UARTRIS register. If the EOT bit is clear, the TXRIS bit is set when the transmit FIFO condition specified in UARTIFLS is met. If the EOT bit is set, the TXRIS bit is set only once all transmitted data, including stop bits, have cleared the serializer.

Bit/Field	Name	Type	Reset	Description
3	SMART	R/W	0	<p>ISO 7816 Smart Card Support</p> <p>The application must ensure that it sets 8-bit word length (<i>WLEN</i> set to 0x3) and even parity (<i>PEN</i> set to 1, <i>EPS</i> set to 1, <i>SPS</i> set to 0) in <b>UARTLCRH</b> when using ISO 7816 mode.</p> <p>In this mode, the value of the <i>STP2</i> bit in <b>UARTLCRH</b> is ignored and the number of stop bits is forced to 2. Note that the UART does not support automatic retransmission on parity errors. If a parity error is detected on transmission, all further transmit operations are aborted and software must handle retransmission of the affected byte or message.</p>
2	SIRLP	R/W	0	<p>UART SIR Low Power Mode</p> <p>This bit selects the IrDA encoding mode. If this bit is cleared to 0, low-level bits are transmitted as an active High pulse with a width of 3/16th of the bit period. If this bit is set to 1, low-level bits are transmitted with a pulse width which is 3 times the period of the <i>IrLPBaud16</i> input signal, regardless of the selected bit rate. Setting this bit uses less power, but might reduce transmission distances. See page 529 for more information.</p>
1	SIREN	R/W	0	<p>UART SIR Enable</p> <p>If this bit is set to 1, the IrDA SIR block is enabled, and the UART will transmit and receive data using SIR protocol.</p>
0	UARTEN	R/W	0	<p>UART Enable</p> <p>If this bit is set to 1, the UART is enabled. When the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.</p>

### Register 9: UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034

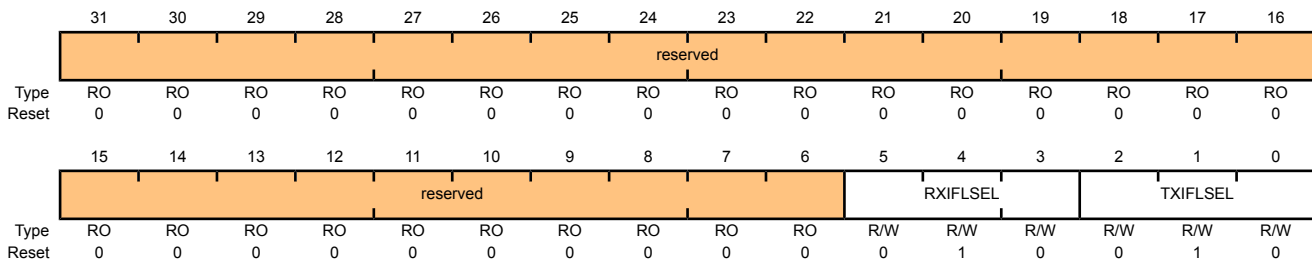
The **UARTIFLS** register is the interrupt FIFO level select register. You can use this register to define the FIFO level at which the **TXRIS** and **RXRIS** bits in the **UARTRIS** register are triggered.

The interrupts are generated based on a transition through a level rather than being based on the level. That is, the interrupts are generated when the fill level progresses through the trigger level. For example, if the receive trigger level is set to the half-way mark, the interrupt is triggered as the module is receiving the 9th character.

Out of reset, the **TXIFLSEL** and **RXIFLSEL** bits are configured so that the FIFOs trigger an interrupt at the half-way mark.

#### UART Interrupt FIFO Level Select (UARTIFLS)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x034  
 Type R/W, reset 0x0000.0012



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:3	RXIFLSEL	R/W	0x2	UART Receive Interrupt FIFO Level Select

The trigger points for the receive interrupt are as follows:

Value	Description
0x0	RX FIFO $\geq$ 1/8 full
0x1	RX FIFO $\geq$ 1/4 full
0x2	RX FIFO $\geq$ 1/2 full (default)
0x3	RX FIFO $\geq$ 3/4 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  <b>Note:</b> If the EOT bit in <b>UARTCTL</b> is set (see page 534), the transmit interrupt is generated once the FIFO is completely empty and all data including stop bits have left the transmit serializer. In this case, the setting of TXIFLSEL is ignored.

## Register 10: UART Interrupt Mask (UARTIM), offset 0x038

The **UARTIM** register is the interrupt mask set/clear register.

On a read, this register gives the current value of the mask on the relevant interrupt. Writing a 1 to a bit allows the corresponding raw interrupt signal to be routed to the interrupt controller. Writing a 0 prevents the raw interrupt signal from being sent to the interrupt controller.

### UART Interrupt Mask (UARTIM)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x038  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LME5IM	LME1IM	LMSBIM	reserved	reserved	OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM	reserved	reserved	reserved	reserved
Type	R/W	R/W	R/W	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	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	LME5IM	R/W	0	<p>LIN Mode Edge 5 Interrupt Mask</p> <p>On a read, the current mask for the <code>LME5IM</code> interrupt is returned.</p> <p>Setting this bit to 1 promotes the <code>LME5IM</code> interrupt to the interrupt controller.</p>
14	LME1IM	R/W	0	<p>LIN Mode Edge 1 Interrupt Mask</p> <p>On a read, the current mask for the <code>LME1IM</code> interrupt is returned.</p> <p>Setting this bit to 1 promotes the <code>LME1IM</code> interrupt to the interrupt controller.</p>
13	LMSBIM	R/W	0	<p>LIN Mode Sync Break Interrupt Mask</p> <p>On a read, the current mask for the <code>LMSBIM</code> interrupt is returned.</p> <p>Setting this bit to 1 promotes the <code>LMSBIM</code> interrupt to the interrupt controller.</p>
12:11	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEIM	R/W	0	<p>UART Overrun Error Interrupt Mask</p> <p>On a read, the current mask for the <code>OEIM</code> interrupt is returned.</p> <p>Setting this bit to 1 promotes the <code>OEIM</code> interrupt to the interrupt controller.</p>

Bit/Field	Name	Type	Reset	Description
9	BEIM	R/W	0	UART Break Error Interrupt Mask On a read, the current mask for the <code>BEIM</code> interrupt is returned. Setting this bit to 1 promotes the <code>BEIM</code> interrupt to the interrupt controller.
8	PEIM	R/W	0	UART Parity Error Interrupt Mask On a read, the current mask for the <code>PEIM</code> interrupt is returned. Setting this bit to 1 promotes the <code>PEIM</code> interrupt to the interrupt controller.
7	FEIM	R/W	0	UART Framing Error Interrupt Mask On a read, the current mask for the <code>FEIM</code> interrupt is returned. Setting this bit to 1 promotes the <code>FEIM</code> interrupt to the interrupt controller.
6	RTIM	R/W	0	UART Receive Time-Out Interrupt Mask On a read, the current mask for the <code>RTIM</code> interrupt is returned. Setting this bit to 1 promotes the <code>RTIM</code> interrupt to the interrupt controller.
5	TXIM	R/W	0	UART Transmit Interrupt Mask On a read, the current mask for the <code>TXIM</code> interrupt is returned. Setting this bit to 1 promotes the <code>TXIM</code> interrupt to the interrupt controller.
4	RXIM	R/W	0	UART Receive Interrupt Mask On a read, the current mask for the <code>RXIM</code> interrupt is returned. Setting this bit to 1 promotes the <code>RXIM</code> interrupt to the interrupt controller.
3:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.



## Register 11: UART Raw Interrupt Status (UARTRIS), offset 0x03C

The **UARTRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt. A write has no effect.

### UART Raw Interrupt Status (UARTRIS)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x03C  
 Type RO, reset 0x0000.000F

	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
	LME5RIS	LME1RIS	LMSBRIS	reserved	OERIS	BERIS	PERIS	FERIS	RTRIS	TXRIS	RXRIS	reserved				
Type	R/W	R/W	R/W	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:16	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	LME5RIS	R/W	0	LIN Mode Edge 5 Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
14	LME1RIS	R/W	0	LIN Mode Edge 1 Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
13	LMSBRIS	R/W	0	LIN Mode Sync Break Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
12:11	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
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.

Bit/Field	Name	Type	Reset	Description
5	TXRIS	RO	0	UART Transmit Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
4	RXRIS	RO	0	UART Receive Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
3:0	reserved	RO	0xF	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 12: UART Masked Interrupt Status (UARTMIS), offset 0x040

The **UARTMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

### UART Masked Interrupt Status (UARTMIS)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x040  
 Type RO, reset 0x0000.0000

	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
	LME5MIS	LME1MIS	LMSBMIS	reserved	OEMIS	BEMIS	PEMIS	FEMIS	RTMIS	TXMIS	RXMIS	reserved				
Type	R/W	R/W	R/W	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
31:16	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	LME5MIS	R/W	0	LIN Mode Edge 5 Masked Interrupt Status Gives the masked interrupt state of this interrupt.
14	LME1MIS	R/W	0	LIN Mode Edge 1 Masked Interrupt Status Gives the masked interrupt state of this interrupt.
13	LMSBMIS	R/W	0	LIN Mode Sync Break Masked Interrupt Status Gives the masked interrupt state of this interrupt.
12:11	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
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.

Bit/Field	Name	Type	Reset	Description
5	TXMIS	RO	0	UART Transmit Masked Interrupt Status Gives the masked interrupt state of this interrupt.
4	RXMIS	RO	0	UART Receive Masked Interrupt Status Gives the masked interrupt state of this interrupt.
3:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 13: UART Interrupt Clear (UARTICR), offset 0x044

The **UARTICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt (both raw interrupt and masked interrupt, if enabled) is cleared. A write of 0 has no effect.

### UART Interrupt Clear (UARTICR)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x044  
 Type W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved					OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC	reserved			
Type	RO	RO	RO	RO	RO	W1C	W1C	W1C	W1C	W1C	W1C	W1C	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEIC	W1C	0	Overrun Error Interrupt Clear The <b>OEIC</b> values are defined as follows:  Value Description 0 No effect on the interrupt. 1 Clears interrupt.
9	BEIC	W1C	0	Break Error Interrupt Clear The <b>BEIC</b> values are defined as follows:  Value Description 0 No effect on the interrupt. 1 Clears interrupt.
8	PEIC	W1C	0	Parity Error Interrupt Clear The <b>PEIC</b> values are defined as follows:  Value Description 0 No effect on the interrupt. 1 Clears interrupt.

Bit/Field	Name	Type	Reset	Description
7	FEIC	W1C	0	Framing Error Interrupt Clear The FEIC values are defined as follows:  Value Description 0 No effect on the interrupt. 1 Clears interrupt.
6	RTIC	W1C	0	Receive Time-Out Interrupt Clear The RTIC values are defined as follows:  Value Description 0 No effect on the interrupt. 1 Clears interrupt.
5	TXIC	W1C	0	Transmit Interrupt Clear The TXIC values are defined as follows:  Value Description 0 No effect on the interrupt. 1 Clears interrupt.
4	RXIC	W1C	0	Receive Interrupt Clear The RXIC values are defined as follows:  Value Description 0 No effect on the interrupt. 1 Clears interrupt.
3:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

## Register 14: UART DMA Control (UARTDMACTL), offset 0x048

The **UARTDMACTL** register is the DMA control register.

### UART DMA Control (UARTDMACTL)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x048  
 Type R/W, reset 0x0000.0000

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

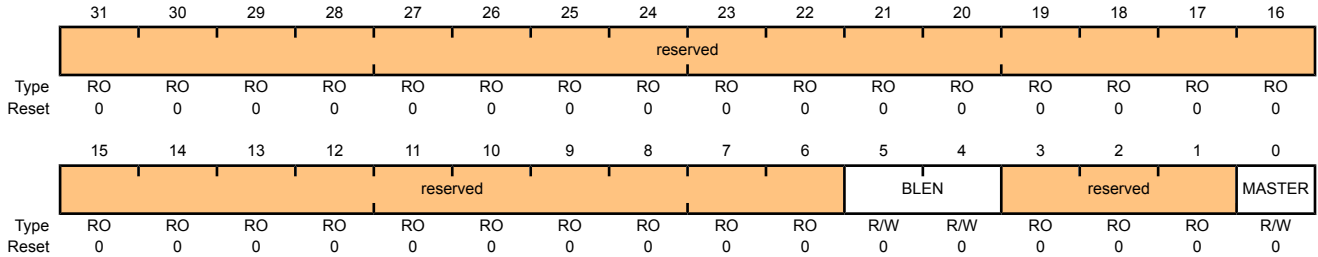
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	DMAERR	R/W	0	DMA on Error  If this bit is set to 1, DMA receive requests are automatically disabled when a receive error occurs.
1	TXDMAE	R/W	0	Transmit DMA Enable  If this bit is set to 1, DMA for the transmit FIFO is enabled.
0	RXDMAE	R/W	0	Receive DMA Enable  If this bit is set to 1, DMA for the receive FIFO is enabled.

### Register 15: UART LIN Control (UARTLCTL), offset 0x090

The **UARTLCTL** register is the configures the operation of the UART when in LIN mode.

#### UART LIN Control (UARTLCTL)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x090  
 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.
5:4	BLEN	R/W	0	Sync Break Length  Value Description 0x3 Sync break length is 16T bits 0x2 Sync break length is 15T bits 0x1 Sync break length is 14T bits 0x0 Sync break length is 13T bits (default)
3:1	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	MASTER	R/W	0	LIN Master Enable  When this bit is set, the UART begins operation as a LIN master.  When this bit is clear, the UART operates as a LIN slave.



## Register 16: UART LIN Snap Shot (UARTLSS), offset 0x094

The **UARTLSS** register captures the free-running timer value when either the Sync Edge 1 or the Sync Edge 5 is detected in LIN mode.

### UART LIN Snap Shot (UARTLSS)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x094  
 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
	TSS															
Type	RO	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

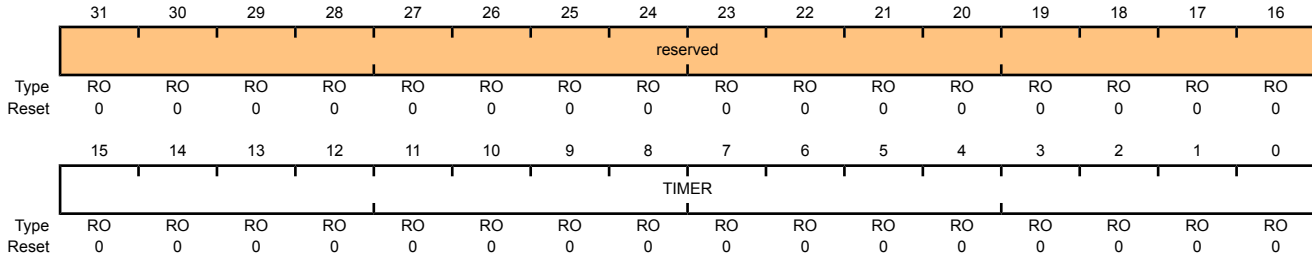
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TSS	RO	0	Timer Snap Shot  This field contains the value of the free-running timer when either the Sync Edge 5 or the Sync Edge 1 was detected.

### Register 17: UART LIN Timer (UARTLTIM), offset 0x098

The **UARTLTIM** register contains the current timer value for the free-running timer that is used to calculate the baud rate when in LIN slave mode. The value in this register is used along with the value in the **UART LIN Snap Shot (UARTLSS)** register to adjust the baud rate to match that of the master.

#### UART LIN Timer (UARTLTIM)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0x098  
 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	TIMER	RO	0	Timer Value This field contains the value of the free-running timer.

**Register 18: UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0**

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

**UART Peripheral Identification 4 (UARTPeriphID4)**

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0xFD0  
 Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

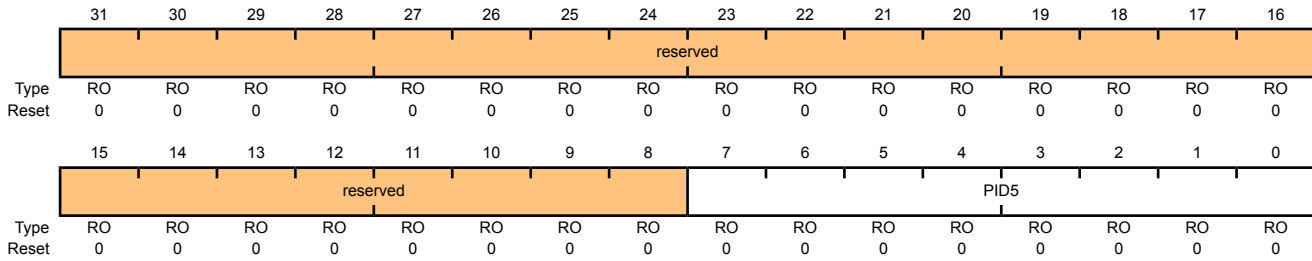
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	UART Peripheral ID Register[7:0]  Can be used by software to identify the presence of this peripheral.

### Register 19: UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

#### UART Peripheral Identification 5 (UARTPeriphID5)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0xFD4  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	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 6 (UARTPeriphID6), offset 0xFD8**

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

## UART Peripheral Identification 6 (UARTPeriphID6)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0xFD8

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID6							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

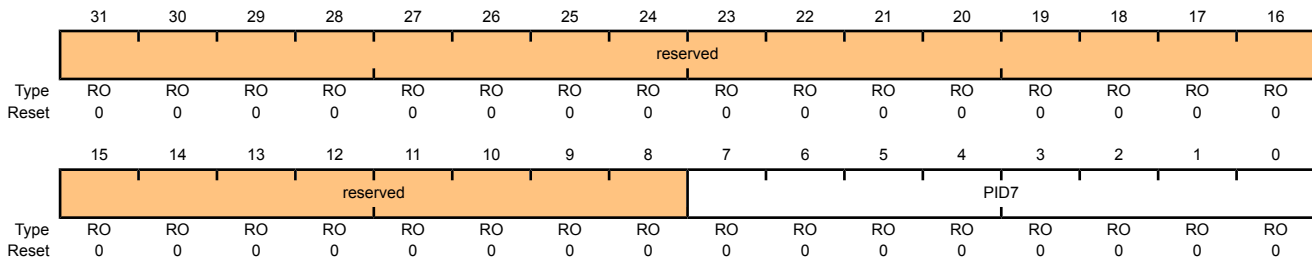
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	UART Peripheral ID Register[23:16] Can be used by software to identify the presence of this peripheral.

## Register 21: UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

### UART Peripheral Identification 7 (UARTPeriphID7)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0xFDC  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	UART Peripheral ID Register[31:24]  Can be used by software to identify the presence of this peripheral.

**Register 22: UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0**

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

## UART Peripheral Identification 0 (UARTPeriphID0)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0xFE0

Type RO, reset 0x0000.0060

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

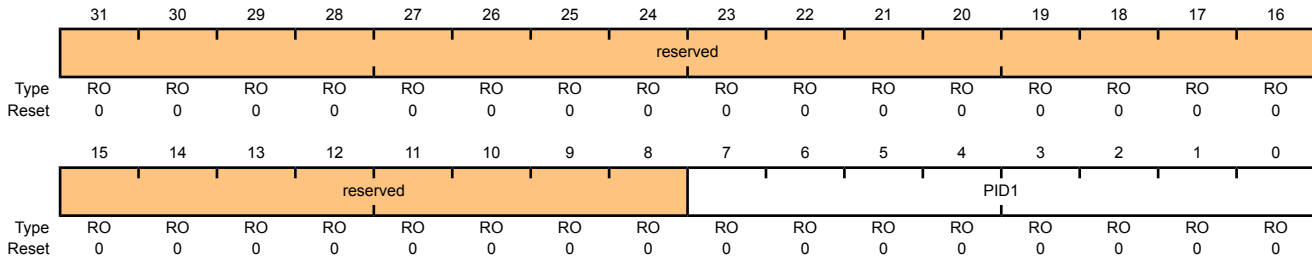
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x60	UART Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

### Register 23: UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

#### UART Peripheral Identification 1 (UARTPeriphID1)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0xFE4  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	UART Peripheral ID Register[15:8]  Can be used by software to identify the presence of this peripheral.



## Register 24: UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

### UART Peripheral Identification 2 (UARTPeriphID2)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0xFE8  
 Type RO, reset 0x0000.0018

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

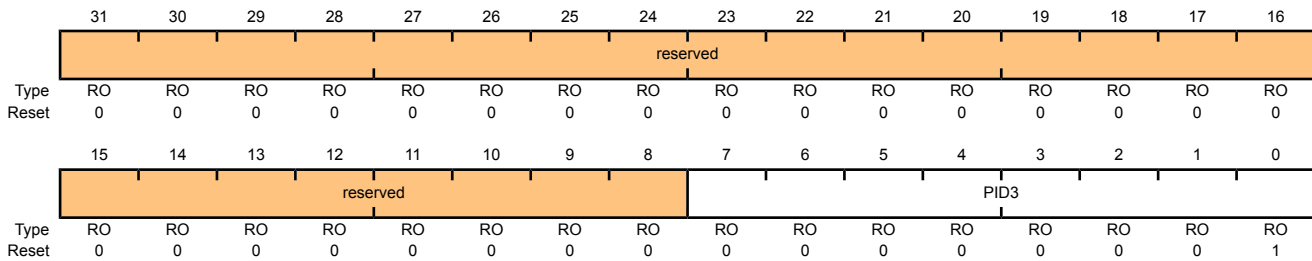
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	UART Peripheral ID Register[23:16]  Can be used by software to identify the presence of this peripheral.

### Register 25: UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

#### UART Peripheral Identification 3 (UARTPeriphID3)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0xFEC  
 Type RO, reset 0x0000.0001



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	UART Peripheral ID Register[31:24]  Can be used by software to identify the presence of this peripheral.

**Register 26: UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0**

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

## UART PrimeCell Identification 0 (UARTPCellID0)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0xFF0  
 Type RO, reset 0x0000.000D

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

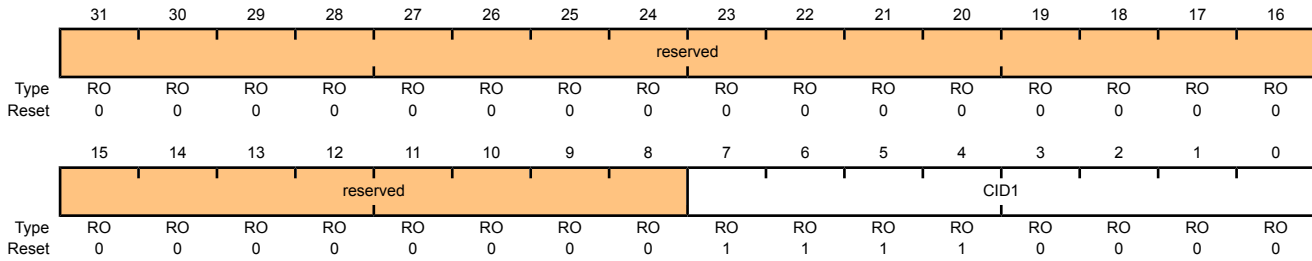
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	UART PrimeCell ID Register[7:0] Provides software a standard cross-peripheral identification system.

### Register 27: UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

#### UART PrimeCell Identification 1 (UARTPCellID1)

UART0 base: 0x4000.C000  
 UART1 base: 0x4000.D000  
 UART2 base: 0x4000.E000  
 Offset 0xFF4  
 Type RO, reset 0x0000.00F0



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	UART PrimeCell ID Register[15:8] Provides software a standard cross-peripheral identification system.

**Register 28: 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

UART2 base: 0x4000.E000

Offset 0xFF8

Type RO, reset 0x0000.0005

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

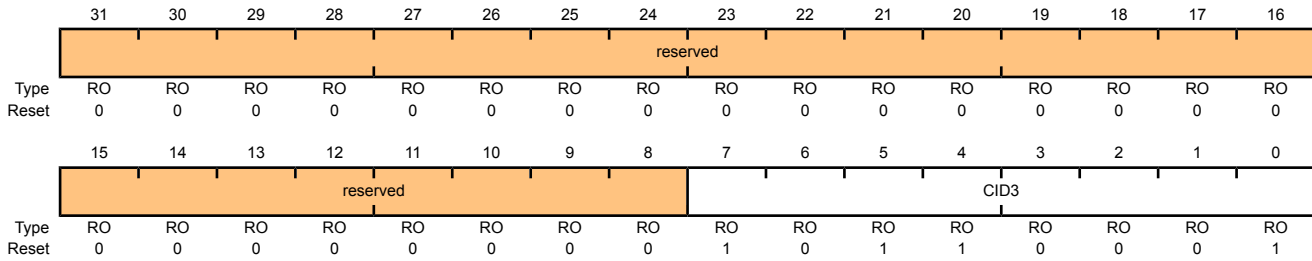
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	UART PrimeCell ID Register[23:16] Provides software a standard cross-peripheral identification system.

### Register 29: 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  
 UART2 base: 0x4000.E000  
 Offset 0xFFC  
 Type RO, reset 0x0000.00B1



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	UART PrimeCell ID Register[31:24] Provides software a standard cross-peripheral identification system.

## 15 Synchronous Serial Interface (SSI)

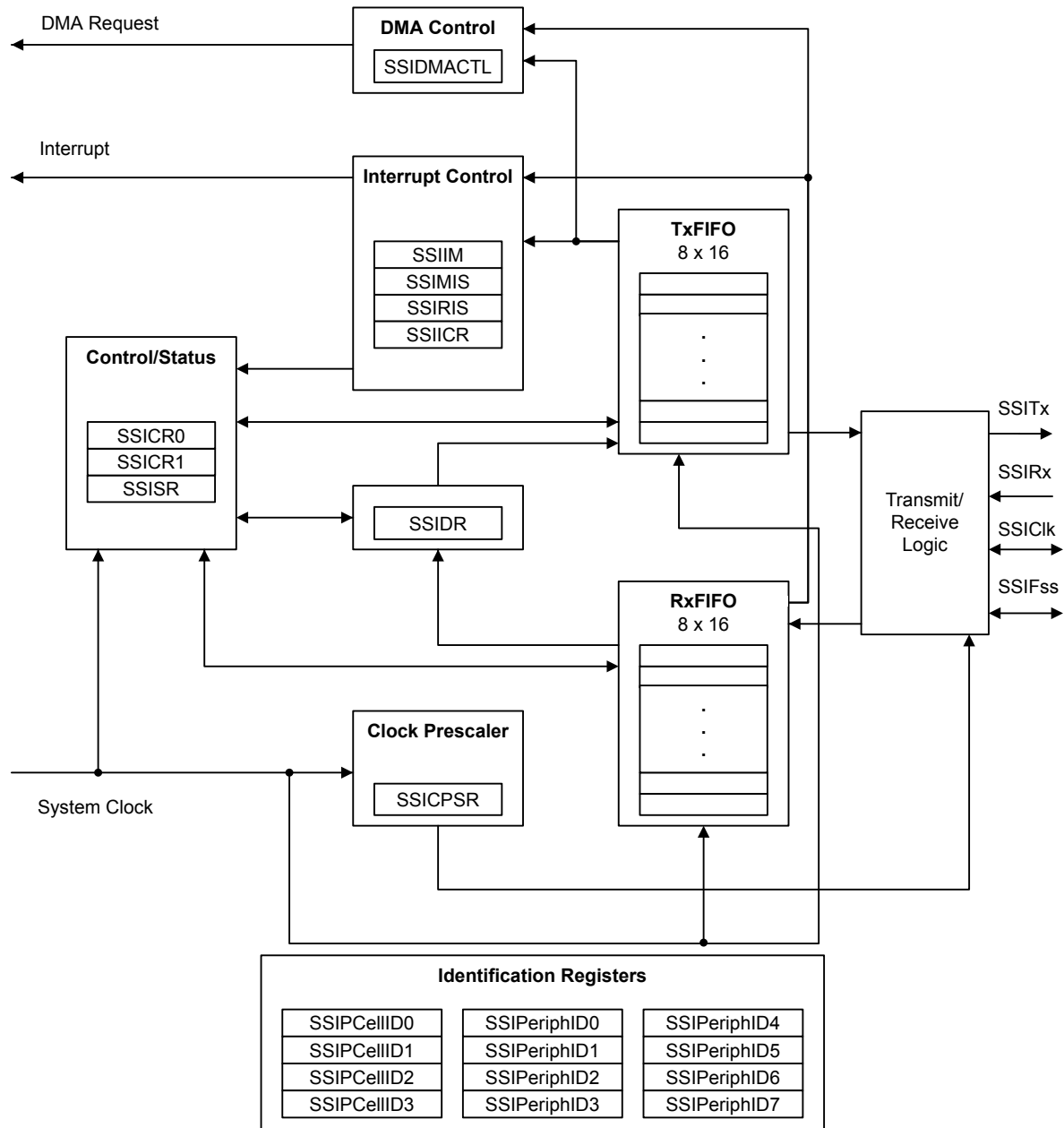
The Stellaris<sup>®</sup> microcontroller includes two Synchronous Serial Interface (SSI) modules. Each SSI is a master or slave interface for synchronous serial communication with peripheral devices that have either Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces.

Each Stellaris<sup>®</sup> SSI module has the following features:

- Programmable interface operation for Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces
- Master or slave operation
- Programmable clock bit rate and prescaler
- Separate transmit and receive FIFOs, 16 bits wide, 8 locations deep
- Programmable data frame size from 4 to 16 bits
- Internal loopback test mode for diagnostic/debug testing
- Standard FIFO-based interrupts and End-of-Transmission interrupt
- Efficient transfers using Micro Direct Memory Access Controller ( $\mu$ DMA)
  - Separate channels for transmit and receive
  - Receive single request asserted when data is in the FIFO; burst request asserted when FIFO contains 4 entries
  - Transmit single request asserted when there is space in the FIFO; burst request asserted when FIFO contains 4 entries

## 15.1 Block Diagram

Figure 15-1. SSI Module Block Diagram



## 15.2 Functional Description

The SSI performs serial-to-parallel conversion on data received from a peripheral device. The CPU accesses data, control, and status information. The transmit and receive paths are buffered with internal FIFO memories allowing up to eight 16-bit values to be stored independently in both transmit and receive modes. The SSI also supports the DMA interface. The transmit and receive FIFOs can be programmed as destination/source addresses in the DMA module. DMA operation is enabled by setting the appropriate bit(s) in the **SSIDMACTL** register (see page 590).



## 15.2.1 Bit Rate Generation

The SSI includes a programmable bit rate clock divider and prescaler to generate the serial output clock. Bit rates are supported to 2 MHz and higher, although maximum bit rate is determined by peripheral devices.

The serial bit rate is derived by dividing down the input clock (FSysClk). The clock is first divided by an even prescale value CPSDVSR from 2 to 254, which is programmed in the **SSI Clock Prescale (SSICPSR)** register (see page 584). 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 577).

The frequency of the output clock SSIClk is defined by:

$$SSIClk = F_{SysClk} / (CPSDVSR * (1 + SCR))$$

**Note:** Although the SSIClk transmit clock can theoretically be 40 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 961 to view SSI timing parameters.

## 15.2.2 FIFO Operation

### 15.2.2.1 Transmit FIFO

The common transmit FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. The CPU writes data to the FIFO by writing the **SSI Data (SSIDR)** register (see page 581), and data is stored in the FIFO until it is read out by the transmission logic.

When configured as a master or a slave, parallel data is written into the transmit FIFO prior to serial conversion and transmission to the attached slave or master, respectively, through the SSITx pin.

### 15.2.2.2 Receive FIFO

The common receive FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. Received data from the serial interface is stored in the buffer until read out by the CPU, which accesses the read FIFO by reading the **SSIDR** register.

When configured as a master or slave, serial data received through the SSIRx pin is registered prior to parallel loading into the attached slave or master receive FIFO, respectively.

## 15.2.3 Interrupts

The SSI can generate interrupts when the following conditions are observed:

- Transmit FIFO service
- Receive FIFO service
- Receive FIFO time-out
- Receive FIFO overrun
- End of transmission

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 585). 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 587 and page 588, respectively).

## 15.2.4 Frame Formats

Each data frame is between 4 and 16 bits long, depending on the size of data programmed, and is transmitted starting with the MSB. There are three basic frame types that can be selected:

- Texas Instruments synchronous serial
- Freescale SPI
- MICROWIRE

For all three formats, the serial clock (*SSIClk*) is held inactive while the SSI is idle, and *SSIClk* transitions at the programmed frequency only during active transmission or reception of data. The idle state of *SSIClk* is utilized to provide a receive timeout indication that occurs when the receive FIFO still contains data after a timeout period.

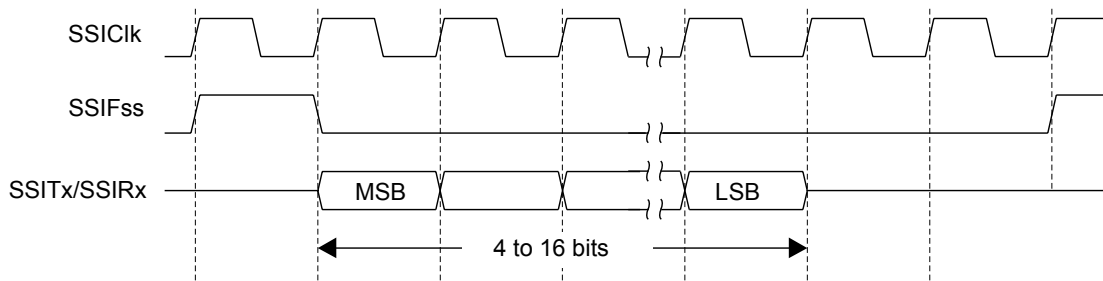
For Freescale SPI and MICROWIRE frame formats, the serial frame (*SSIFSS*) pin is active Low, and is asserted (pulled down) during the entire transmission of the frame.

For Texas Instruments synchronous serial frame format, the *SSIFSS* pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the SSI and the off-chip slave device drive their output data on the rising edge of *SSIClk*, and latch data from the other device on the falling edge.

Unlike the full-duplex transmission of the other two frame formats, the MICROWIRE format uses a special master-slave messaging technique, which operates at half-duplex. In this mode, when a frame begins, an 8-bit control message is transmitted to the off-chip slave. During this transmit, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the requested data. The returned data can be 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

### 15.2.4.1 Texas Instruments Synchronous Serial Frame Format

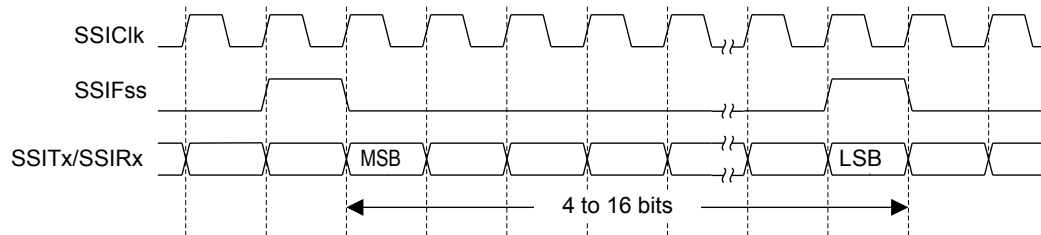
Figure 15-2 on page 567 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

**Figure 15-2. TI Synchronous Serial Frame Format (Single Transfer)**

In this mode, *SSIClk* and *SSIFss* are forced Low, and the transmit data line *SSITx* is tristated whenever the SSI is idle. Once the bottom entry of the transmit FIFO contains data, *SSIFss* is pulsed High for one *SSIClk* period. The value to be transmitted is also transferred from the transmit FIFO to the serial shifter of the transmit logic. On the next rising edge of *SSIClk*, the MSB of the 4 to 16-bit data frame is shifted out on the *SSITx* pin. Likewise, the MSB of the received data is shifted onto the *SSIRx* pin by the off-chip serial slave device.

Both the SSI and the off-chip serial slave device then clock each data bit into their serial shifter on the falling edge of each *SSIClk*. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of *SSIClk* after the LSB has been latched.

Figure 15-3 on page 567 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

**Figure 15-3. TI Synchronous Serial Frame Format (Continuous Transfer)**

#### 15.2.4.2 Freescale SPI Frame Format

The Freescale SPI interface is a four-wire interface where the *SSIFss* signal behaves as a slave select. The main feature of the Freescale SPI format is that the inactive state and phase of the *SSIClk* signal are programmable through the *SPO* and *SPH* bits within the **SSISCR0** control register.

##### **SPO Clock Polarity Bit**

When the *SPO* clock polarity control bit is Low, it produces a steady state Low value on the *SSIClk* pin. If the *SPO* bit is High, a steady state High value is placed on the *SSIClk* pin when data is not being transferred.

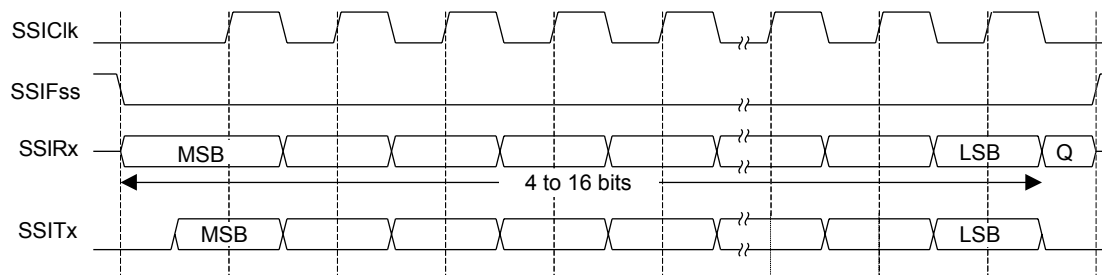
### SPH Phase Control Bit

The *SPH* phase control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the *SPH* phase control bit is Low, data is captured on the first clock edge transition. If the *SPH* bit is High, data is captured on the second clock edge transition.

#### 15.2.4.3 Freescale SPI Frame Format with SPO=0 and SPH=0

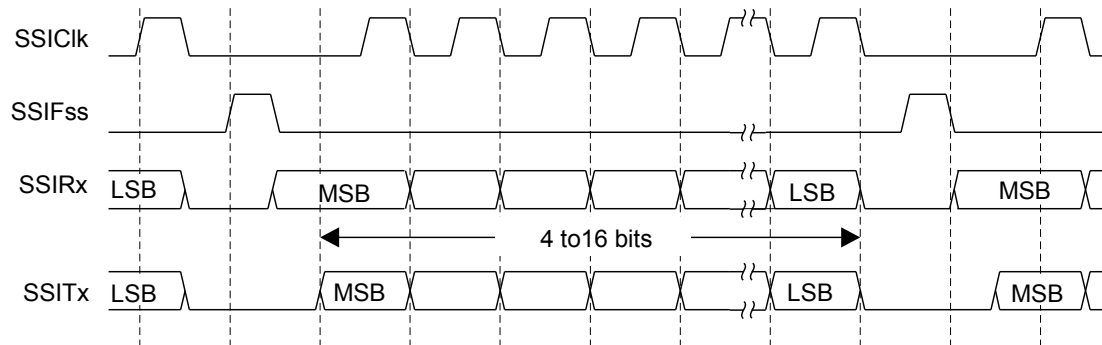
Single and continuous transmission signal sequences for Freescale SPI format with *SPO*=0 and *SPH*=0 are shown in Figure 15-4 on page 568 and Figure 15-5 on page 568.

**Figure 15-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0**



Note: Q is undefined.

**Figure 15-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0**



In this configuration, during idle periods:

- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the *SSIFss* master signal being driven Low. This causes slave data to be enabled onto the *SSIRx* input line of the master. The master *SSITx* output pad is enabled.

One half  $SSIClk$  period later, valid master data is transferred to the  $SSITx$  pin. Now that both the master and slave data have been set, the  $SSIClk$  master clock pin goes High after one further half  $SSIClk$  period.

The data is now captured on the rising and propagated on the falling edges of the  $SSIClk$  signal.

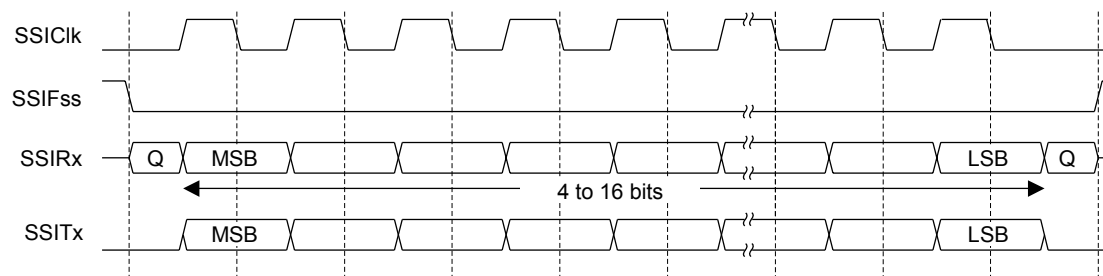
In the case of a single word transmission, after all bits of the data word have been transferred, the  $SSIFss$  line is returned to its idle High state one  $SSIClk$  period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the  $SSIFss$  signal must be pulsed High between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the  $SPH$  bit is logic zero. Therefore, the master device must raise the  $SSIFss$  pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the  $SSIFss$  pin is returned to its idle state one  $SSIClk$  period after the last bit has been captured.

#### 15.2.4.4 Freescale SPI Frame Format with $SPO=0$ and $SPH=1$

The transfer signal sequence for Freescale SPI format with  $SPO=0$  and  $SPH=1$  is shown in Figure 15-6 on page 569, which covers both single and continuous transfers.

**Figure 15-6. Freescale SPI Frame Format with  $SPO=0$  and  $SPH=1$**



**Note:** Q is undefined.

In this configuration, during idle periods:

- $SSIClk$  is forced Low
- $SSIFss$  is forced High
- The transmit data line  $SSITx$  is arbitrarily forced Low
- When the SSI is configured as a master, it enables the  $SSIClk$  pad
- When the SSI is configured as a slave, it disables the  $SSIClk$  pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the  $SSIFss$  master signal being driven Low. The master  $SSITx$  output is enabled. After a further one half  $SSIClk$  period, both master and slave valid data is enabled onto their respective transmission lines. At the same time, the  $SSIClk$  is enabled with a rising edge transition.

Data is then captured on the falling edges and propagated on the rising edges of the  $SSIClk$  signal.

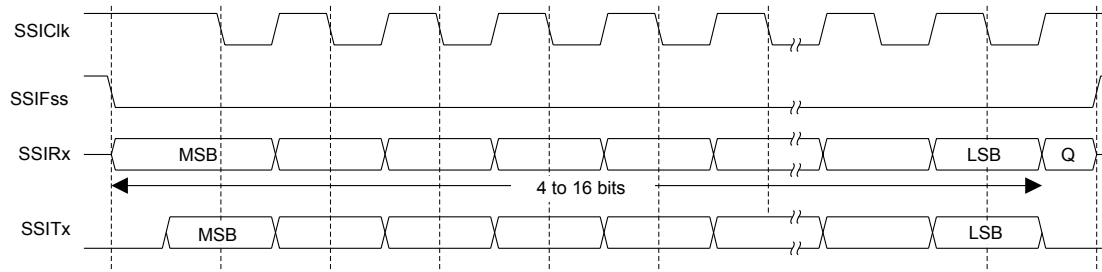
In the case of a single word transfer, after all bits have been transferred, the  $SSIFss$  line is returned to its idle High state one  $SSIClk$  period after the last bit has been captured.

For continuous back-to-back transfers, the  $SSIFSS$  pin is held Low between successive data words and termination is the same as that of the single word transfer.

#### 15.2.4.5 Freescale SPI Frame Format with SPO=1 and SPH=0

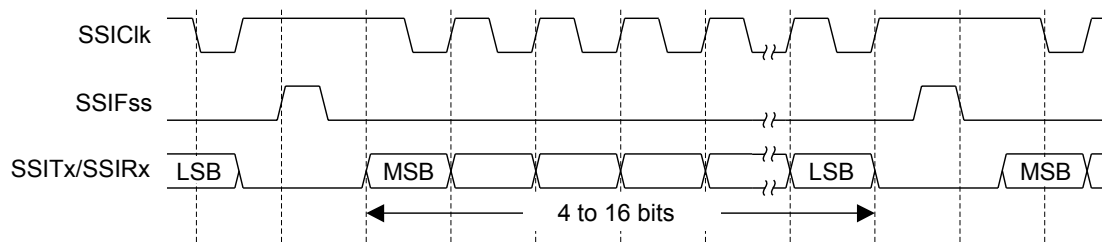
Single and continuous transmission signal sequences for Freescale SPI format with SPO=1 and SPH=0 are shown in Figure 15-7 on page 570 and Figure 15-8 on page 570.

**Figure 15-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0**



**Note:** Q is undefined.

**Figure 15-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0**



In this configuration, during idle periods:

- $SSIClk$  is forced High
- $SSIFSS$  is forced High
- The transmit data line  $SSITx$  is arbitrarily forced Low
- When the SSI is configured as a master, it enables the  $SSIClk$  pad
- When the SSI is configured as a slave, it disables the  $SSIClk$  pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the  $SSIFSS$  master signal being driven Low, which causes slave data to be immediately transferred onto the  $SSIRx$  line of the master. The master  $SSITx$  output pad is enabled.

One half period later, valid master data is transferred to the  $SSITx$  line. Now that both the master and slave data have been set, the  $SSIClk$  master clock pin becomes Low after one further half  $SSIClk$  period. This means that data is captured on the falling edges and propagated on the rising edges of the  $SSIClk$  signal.

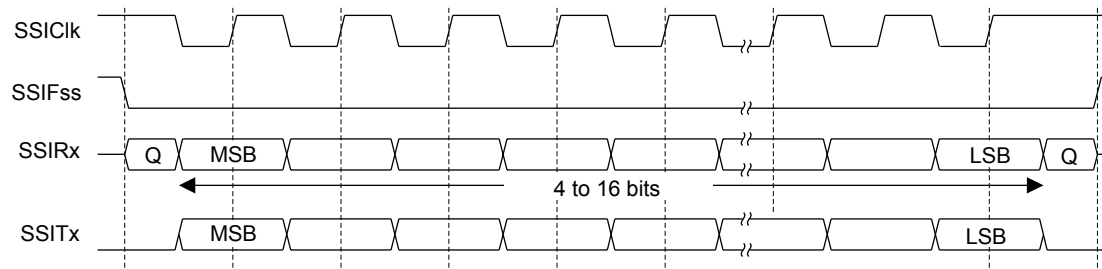
In the case of a single word transmission, after all bits of the data word are transferred, the  $SSIFSS$  line is returned to its idle High state one  $SSIClk$  period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the  $SSIF_{SS}$  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  $SSIF_{SS}$  pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the  $SSIF_{SS}$  pin is returned to its idle state one  $SSIClk$  period after the last bit has been captured.

#### 15.2.4.6 Freescale SPI Frame Format with $SPO=1$ and $SPH=1$

The transfer signal sequence for Freescale SPI format with  $SPO=1$  and  $SPH=1$  is shown in Figure 15-9 on page 571, which covers both single and continuous transfers.

**Figure 15-9. Freescale SPI Frame Format with  $SPO=1$  and  $SPH=1$**



**Note:** Q is undefined.

In this configuration, during idle periods:

- $SSIClk$  is forced High
- $SSIF_{SS}$  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  $SSIF_{SS}$  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.

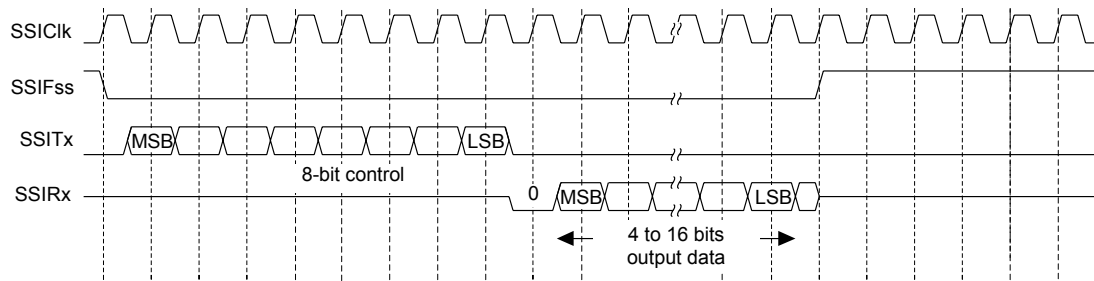
After all bits have been transferred, in the case of a single word transmission, the  $SSIF_{SS}$  line is returned to its idle high state one  $SSIClk$  period after the last bit has been captured.

For continuous back-to-back transmissions, the  $SSIF_{SS}$  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  $SSIF_{SS}$  pin is held Low between successive data words and termination is the same as that of the single word transfer.

#### 15.2.4.7 MICROWIRE Frame Format

Figure 15-10 on page 572 shows the MICROWIRE frame format, again for a single frame. Figure 15-11 on page 573 shows the same format when back-to-back frames are transmitted.

**Figure 15-10. MICROWIRE Frame Format (Single Frame)**

MICROWIRE format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex, using a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SSI to the off-chip slave device. During this transmission, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the required data. The returned data is 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

In this configuration, during idle periods:

- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low

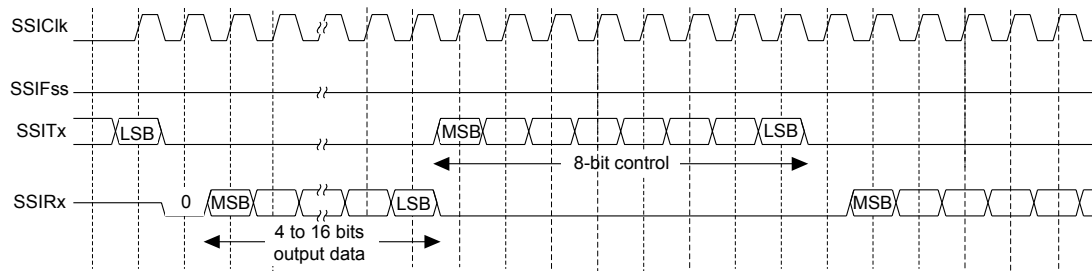
A transmission is triggered by writing a control byte to the transmit FIFO. The falling edge of SSIFss causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic, and the MSB of the 8-bit control frame to be shifted out onto the SSITx pin. SSIFss remains Low for the duration of the frame transmission. The SSIRx pin remains tristated during this transmission.

The off-chip serial slave device latches each control bit into its serial shifter on the rising edge of each SSIClk. After the last bit is latched by the slave device, the control byte is decoded during a one clock wait-state, and the slave responds by transmitting data back to the SSI. Each bit is driven onto the SSIRx line on the falling edge of SSIClk. The SSI in turn latches each bit on the rising edge of SSIClk. At the end of the frame, for single transfers, the SSIFss signal is pulled High one clock period after the last bit has been latched in the receive serial shifter, which causes the data to be transferred to the receive FIFO.

**Note:** The off-chip slave device can tristate the receive line either on the falling edge of SSIClk after the LSB has been latched by the receive shifter, or when the SSIFss pin goes High.

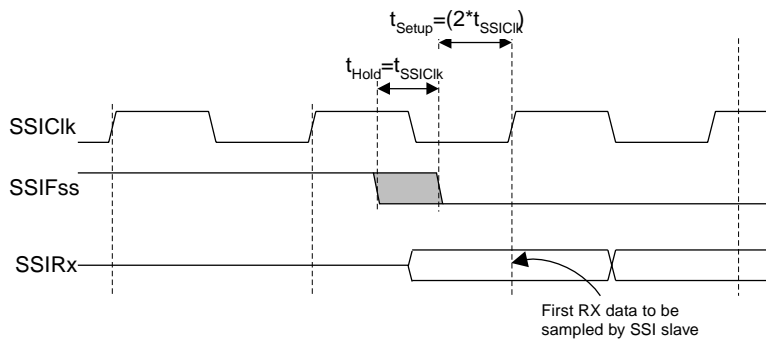
For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the SSIFss line is continuously asserted (held Low) and transmission of data occurs back-to-back. The control byte of the next frame follows directly after the LSB of the received data from the current frame. Each of the received values is transferred from the receive shifter on the falling edge of SSIClk, after the LSB of the frame has been latched into the SSI.



**Figure 15-11. MICROWIRE Frame Format (Continuous Transfer)**

In the MICROWIRE mode, the SSI slave samples the first bit of receive data on the rising edge of `SSIClk` after `SSIFss` has gone Low. Masters that drive a free-running `SSIClk` must ensure that the `SSIFss` signal has sufficient setup and hold margins with respect to the rising edge of `SSIClk`.

Figure 15-12 on page 573 illustrates these setup and hold time requirements. With respect to the `SSIClk` rising edge on which the first bit of receive data is to be sampled by the SSI slave, `SSIFss` must have a setup of at least two times the period of `SSIClk` on which the SSI operates. With respect to the `SSIClk` rising edge previous to this edge, `SSIFss` must have a hold of at least one `SSIClk` period.

**Figure 15-12. MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements**

### 15.2.5 DMA Operation

The SSI peripheral provides an interface connected to the  $\mu$ DMA controller. The DMA operation of the SSI is enabled through the **SSI DMA Control (SSIDMACTL)** register. When DMA operation is enabled, the SSI will assert a DMA request on the receive or transmit channel when the associated FIFO can transfer data. For the receive channel, a single transfer request is asserted whenever there is any data in the receive FIFO. A burst transfer request is asserted whenever the amount of data in the receive FIFO is 4 or more items. For the transmit channel, a single transfer request is asserted whenever there is at least one empty location in the transmit FIFO. The burst request is asserted whenever the transmit FIFO has 4 or more empty slots. The single and burst DMA transfer requests are handled automatically by the  $\mu$ DMA controller depending how the DMA channel is configured. To enable DMA operation for the receive channel, the `RXDMAE` bit of the **DMA Control (SSIDMACTL)** register should be set. To enable DMA operation for the transmit channel, the `TXDMAE` bit of **SSIDMACTL** should be set. If DMA is enabled, then the  $\mu$ DMA controller will trigger an interrupt when a transfer is complete. The interrupt will occur on the SSI interrupt vector. Therefore, if interrupts are used for SSI operation and DMA is enabled, the SSI interrupt handler must be designed to handle the  $\mu$ DMA completion interrupt.

See “Micro Direct Memory Access (μDMA)” on page 219 for more details about programming the μDMA controller.

## 15.3 Initialization and Configuration

To use the SSI, its peripheral clock must be enabled by setting the *SSI* bit in the **RCGC1** register. See page 163. In addition, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register in the System Control module. See page 175. To find out which GPIO port to enable, refer to Table 24-5 on page 937.

For each of the frame formats, the SSI is configured using the following steps:

1. Ensure that the *SSE* bit in the **SSICR1** register is disabled before making any configuration changes.
2. Select whether the SSI is a master or slave:
  - a. For master operations, set the **SSICR1** register to 0x0000.0000.
  - b. For slave mode (output enabled), set the **SSICR1** register to 0x0000.0004.
  - c. For slave mode (output disabled), set the **SSICR1** register to 0x0000.000C.
3. Configure the clock prescale divisor by writing the **SSICPSR** register.
4. Write the **SSICR0** register with the following configuration:
  - Serial clock rate (*SCR*)
  - Desired clock phase/polarity, if using Freescale SPI mode (*SPH* and *SPO*)
  - The protocol mode: Freescale SPI, TI SSF, MICROWIRE (*FRF*)
  - The data size (*DSS*)
5. Optionally, configure the μDMA channel (see “Micro Direct Memory Access (μDMA)” on page 219) and enable the DMA option(s) in the **SSIDMACTL** register.
6. Enable the SSI by setting the *SSE* bit in the **SSICR1** register.

As an example, assume the SSI must be configured to operate with the following parameters:

- Master operation
- Freescale SPI mode (*SPO*=1, *SPH*=1)
- 1 Mbps bit rate
- 8 data bits

Assuming the system clock is 20 MHz, the bit rate calculation would be:

$$F_{SSIClk} = F_{SysClk} / (CPSDVSR * (1 + SCR))$$

$$1 \times 10^6 = 20 \times 10^6 / (CPSDVSR * (1 + SCR))$$

In this case, if *CPSDVSR*=2, *SCR* must be 9.

The configuration sequence would be as follows:

1. Ensure that the `SSE` bit in the **SSICR1** register is disabled.
2. Write the **SSICR1** register with a value of `0x0000.0000`.
3. Write the **SSICPSR** register with a value of `0x0000.0002`.
4. Write the **SSICR0** register with a value of `0x0000.09C7`.
5. The SSI is then enabled by setting the `SSE` bit in the **SSICR1** register to 1.

## 15.4 Register Map

Table 15-1 on page 575 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`
- SSI1: `0x4000.9000`

Note that the SSI module clock must be enabled before the registers can be programmed (see page 163).

**Note:** The SSI must be disabled (see the `SSE` bit in the **SSICR1** register) before any of the control registers are reprogrammed.

**Table 15-1. SSI Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	SSICR0	R/W	0x0000.0000	SSI Control 0	577
0x004	SSICR1	R/W	0x0000.0000	SSI Control 1	579
0x008	SSIDR	R/W	0x0000.0000	SSI Data	581
0x00C	SSISR	RO	0x0000.0003	SSI Status	582
0x010	SSICPSR	R/W	0x0000.0000	SSI Clock Prescale	584
0x014	SSIIM	R/W	0x0000.0000	SSI Interrupt Mask	585
0x018	SSIRIS	RO	0x0000.0008	SSI Raw Interrupt Status	587
0x01C	SSIMIS	RO	0x0000.0000	SSI Masked Interrupt Status	588
0x020	SSIICR	W1C	0x0000.0000	SSI Interrupt Clear	589
0x024	SSIDMACTL	R/W	0x0000.0000	SSI DMA Control	590
0xFD0	SSIPeriphID4	RO	0x0000.0000	SSI Peripheral Identification 4	591
0xFD4	SSIPeriphID5	RO	0x0000.0000	SSI Peripheral Identification 5	592
0xFD8	SSIPeriphID6	RO	0x0000.0000	SSI Peripheral Identification 6	593
0xFDC	SSIPeriphID7	RO	0x0000.0000	SSI Peripheral Identification 7	594
0xFE0	SSIPeriphID0	RO	0x0000.0022	SSI Peripheral Identification 0	595
0xFE4	SSIPeriphID1	RO	0x0000.0000	SSI Peripheral Identification 1	596

Offset	Name	Type	Reset	Description	See page
0xFE8	SSIPeriphID2	RO	0x0000.0018	SSI Peripheral Identification 2	597
0xFEC	SSIPeriphID3	RO	0x0000.0001	SSI Peripheral Identification 3	598
0xFF0	SSIPCellID0	RO	0x0000.000D	SSI PrimeCell Identification 0	599
0xFF4	SSIPCellID1	RO	0x0000.00F0	SSI PrimeCell Identification 1	600
0xFF8	SSIPCellID2	RO	0x0000.0005	SSI PrimeCell Identification 2	601
0xFFC	SSIPCellID3	RO	0x0000.00B1	SSI PrimeCell Identification 3	602

## 15.5 Register Descriptions

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

## Register 1: SSI Control 0 (SSICR0), offset 0x000

**SSICR0** is control register 0 and contains bit fields that control various functions within the SSI module. Functionality such as protocol mode, clock rate, and data size are configured in this register.

### SSI Control 0 (SSICR0)

SSI0 base: 0x4000.8000  
 SSI1 base: 0x4000.9000  
 Offset 0x000  
 Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	SCR	R/W	0x0000	SSI Serial Clock Rate  The value <i>SCR</i> is used to generate the transmit and receive bit rate of the SSI. The bit rate is:  $BR = F_{SSIClk} / (CPSDVSr * (1 + SCR))$  where <i>CPSDVSr</i> is an even value from 2-254 programmed in the <b>SSICPSR</b> register, and <i>SCR</i> 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 <i>SPH</i> 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 <i>SPH</i> bit is 0, data is captured on the first clock edge transition. If <i>SPH</i> 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 <i>SPO</i> bit is 0, it produces a steady state Low value on the <i>SSIClk</i> pin. If <i>SPO</i> is 1, a steady state High value is placed on the <i>SSIClk</i> pin when data is not being transferred.

Bit/Field	Name	Type	Reset	Description
5:4	FRF	R/W	0x0	SSI Frame Format Select The FRF values are defined as follows:  Value Frame Format 0x0 Freescale SPI Frame Format 0x1 Texas Instruments Synchronous Serial Frame Format 0x2 MICROWIRE Frame Format 0x3 Reserved
3:0	DSS	R/W	0x00	SSI Data Size Select The DSS values are defined as follows:  Value Data Size 0x0-0x2 Reserved 0x3 4-bit data 0x4 5-bit data 0x5 6-bit data 0x6 7-bit data 0x7 8-bit data 0x8 9-bit data 0x9 10-bit data 0xA 11-bit data 0xB 12-bit data 0xC 13-bit data 0xD 14-bit data 0xE 15-bit data 0xF 16-bit data

## Register 2: SSI Control 1 (SSICR1), offset 0x004

**SSICR1** is control register 1 and contains bit fields that control various functions within the SSI module. Master and slave mode functionality is controlled by this register.

### SSI Control 1 (SSICR1)

SSI0 base: 0x4000.8000  
 SSI1 base: 0x4000.9000  
 Offset 0x004  
 Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x00	Software should not rely on the value of 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	EOT	R/W	0	End of Transmission  When set to 1, this bit enables the End of Transmit interrupt mode for the TXIM interrupt.
3	SOD	R/W	0	SSI Slave Mode Output Disable  This bit is relevant only in the Slave mode ( $MS=1$ ). In multiple-slave systems, it is possible for the SSI master to broadcast a message to all slaves in the system while ensuring that only one slave drives data onto the serial output line. In such systems, the TXD lines from multiple slaves could be tied together. To operate in such a system, the SOD bit can be configured so that the SSI slave does not drive the SSITx pin.  The SOD values are defined as follows:  Value Description 0 SSI can drive SSITx output in Slave Output mode. 1 SSI must not drive the SSITx output in Slave mode.
2	MS	R/W	0	SSI Master/Slave Select  This bit selects Master or Slave mode and can be modified only when SSI is disabled ( $SSE=0$ ).  The MS values are defined as follows:  Value Description 0 Device configured as a master. 1 Device configured as a slave.

Bit/Field	Name	Type	Reset	Description						
1	SSE	R/W	0	<p>SSI Synchronous Serial Port Enable</p> <p>Setting this bit enables SSI operation.</p> <p>The <code>SSE</code> values are defined as follows:</p> <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0</td><td>SSI operation disabled.</td></tr><tr><td>1</td><td>SSI operation enabled.</td></tr></tbody></table> <p><b>Note:</b> This bit must be set to 0 before any control registers are reprogrammed.</p>	Value	Description	0	SSI operation disabled.	1	SSI operation enabled.
Value	Description									
0	SSI operation disabled.									
1	SSI operation enabled.									
0	LBM	R/W	0	<p>SSI Loopback Mode</p> <p>Setting this bit enables Loopback Test mode.</p> <p>The <code>LBM</code> values are defined as follows:</p> <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0</td><td>Normal serial port operation enabled.</td></tr><tr><td>1</td><td>Output of the transmit serial shift register is connected internally to the input of the receive serial shift register.</td></tr></tbody></table>	Value	Description	0	Normal serial port operation enabled.	1	Output of the transmit serial shift register is connected internally to the input of the receive serial shift register.
Value	Description									
0	Normal serial port operation enabled.									
1	Output of the transmit serial shift register is connected internally to the input of the receive serial shift register.									



### Register 3: SSI Data (SSIDR), offset 0x008

**SSIDR** is the data register and is 16-bits wide. When **SSIDR** is read, the entry in the receive FIFO (pointed to by the current FIFO read pointer) is accessed. As data values are removed by the SSI receive logic from the incoming data frame, they are placed into the entry in the receive FIFO (pointed to by the current FIFO write pointer).

When **SSIDR** is written to, the entry in the transmit FIFO (pointed to by the write pointer) is written to. Data values are removed from the transmit FIFO one value at a time by the transmit logic. It is loaded into the transmit serial shifter, then serially shifted out onto the **SSITx** pin at the programmed bit rate.

When a data size of less than 16 bits is selected, the user must right-justify data written to the transmit FIFO. The transmit logic ignores the unused bits. Received data less than 16 bits is automatically right-justified in the receive buffer.

When the SSI is programmed for MICROWIRE frame format, the default size for transmit data is eight bits (the most significant byte is ignored). The receive data size is controlled by the programmer. The transmit FIFO and the receive FIFO are not cleared even when the **SSE** bit in the **SSICR1** register is set to zero. This allows the software to fill the transmit FIFO before enabling the SSI.

#### SSI Data (SSIDR)

SSI0 base: 0x4000.8000  
 SSI1 base: 0x4000.9000  
 Offset 0x008  
 Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DATA	R/W	0x0000	SSI Receive/Transmit Data  A read operation reads the receive FIFO. A write operation writes the transmit FIFO.  Software must right-justify data when the SSI is programmed for a data size that is less than 16 bits. Unused bits at the top are ignored by the transmit logic. The receive logic automatically right-justifies the data.

### Register 4: SSI Status (SSISR), offset 0x00C

SSISR is a status register that contains bits that indicate the FIFO fill status and the SSI busy status.

#### SSI Status (SSISR)

SSI0 base: 0x4000.8000  
 SSI1 base: 0x4000.9000  
 Offset 0x00C  
 Type RO, reset 0x0000.0003

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

Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	BSY	RO	0	SSI Busy Bit  The 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.

---

Bit/Field	Name	Type	Reset	Description
1	TNF	RO	1	SSI Transmit FIFO Not Full The <code>TNF</code> values are defined as follows:  Value Description 0 Transmit FIFO is full. 1 Transmit FIFO is not full.
0	TFE	RO	1	SSI Transmit FIFO Empty The <code>TFE</code> values are defined as follows:  Value Description 0 Transmit FIFO is not empty. 1 Transmit FIFO is empty.

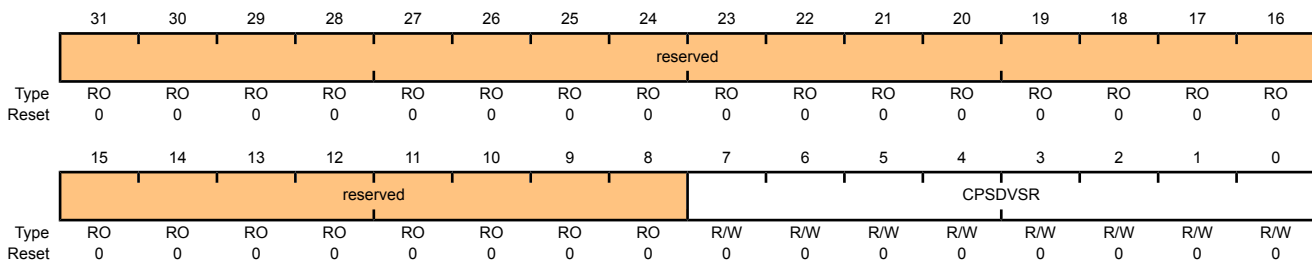
### Register 5: SSI Clock Prescale (SSICPSR), offset 0x010

**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  
 SSI1 base: 0x4000.9000  
 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  
 SSI1 base: 0x4000.9000  
 Offset 0x014  
 Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXIM	R/W	0	SSI Transmit FIFO Interrupt Mask The <b>TXIM</b> values are defined as follows: Value Description 0 TX FIFO half-full or less condition interrupt is masked. 1 TX FIFO half-full or less condition interrupt is not masked.
2	RXIM	R/W	0	SSI Receive FIFO Interrupt Mask The <b>RXIM</b> values are defined as follows: Value Description 0 RX FIFO half-full or more condition interrupt is masked. 1 RX FIFO half-full or more condition interrupt is not masked.
1	RTIM	R/W	0	SSI Receive Time-Out Interrupt Mask The <b>RTIM</b> values are defined as follows: Value Description 0 RX FIFO time-out interrupt is masked. 1 RX FIFO time-out interrupt is not masked.

Bit/Field	Name	Type	Reset	Description
0	RORIM	R/W	0	SSI Receive Overrun Interrupt Mask The RORIM values are defined as follows:  Value Description 0 RX FIFO overrun interrupt is masked. 1 RX FIFO overrun interrupt is not masked.

## Register 7: SSI Raw Interrupt Status (SSIRIS), offset 0x018

The **SSIRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect.

### SSI Raw Interrupt Status (SSIRIS)

SSI0 base: 0x4000.8000  
 SSI1 base: 0x4000.9000  
 Offset 0x018  
 Type RO, reset 0x0000.0008

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXRIS	RO	1	SSI Transmit FIFO Raw Interrupt Status  If the <code>EOT</code> bit in the <b>SSICR1</b> register is set to 0, this bit indicates that the transmit FIFO is half full or less. If the <code>EOT</code> bit is set to 1, this bit indicates that the transmit FIFO is empty, and the last bit has been transmitted out of the serializer.
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  
 SSI1 base: 0x4000.9000  
 Offset 0x01C  
 Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXMIS	RO	0	SSI Transmit FIFO Masked Interrupt Status  If the <code>EOT</code> bit in the <b>SSICR1</b> register is set to 0, this bit indicates that the transmit FIFO is half full or less. If the <code>EOT</code> bit is set to 1, this bit indicates that the transmit FIFO is empty, and the last bit has been transmitted out of the serializer.
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

SSI1 base: 0x4000.9000

Offset 0x020

Type W1C, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	RTIC	W1C	0	SSI Receive Time-Out Interrupt Clear  The <b>RTIC</b> values are defined as follows:  Value Description 0 No effect on interrupt. 1 Clears interrupt.
0	RORIC	W1C	0	SSI Receive Overrun Interrupt Clear  The <b>RORIC</b> values are defined as follows:  Value Description 0 No effect on interrupt. 1 Clears interrupt.

### Register 10: SSI DMA Control (SSIDMACTL), offset 0x024

The **SSIDMACTL** register is the DMA control register.

#### SSI DMA Control (SSIDMACTL)

SSI0 base: 0x4000.8000  
 SSI1 base: 0x4000.9000  
 Offset 0x024  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														TXDMAE	RXDMAE
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
31:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	TXDMAE	R/W	0	Transmit DMA Enable If this bit is set to 1, DMA for the transmit FIFO is enabled.
0	RXDMAE	R/W	0	Receive DMA Enable If this bit is set to 1, DMA for the receive FIFO is enabled.

## Register 11: SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0

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

### SSI Peripheral Identification 4 (SSIPeriphID4)

SSI0 base: 0x4000.8000

SSI1 base: 0x4000.9000

Offset 0xFD0

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

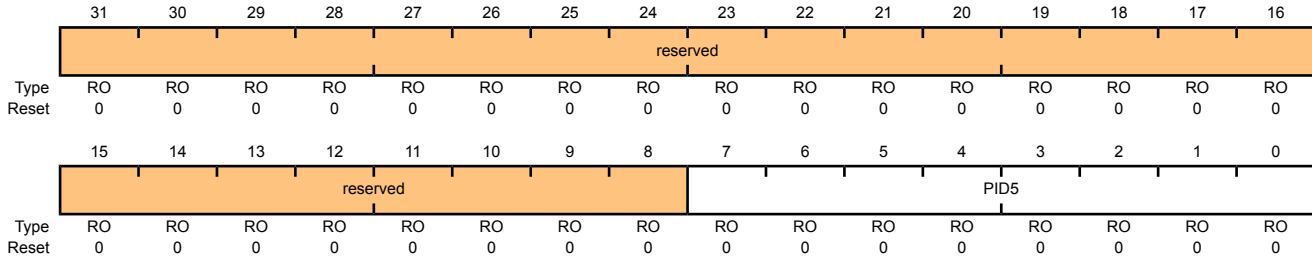
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	SSI Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

### Register 12: SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4

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

#### SSI Peripheral Identification 5 (SSIPeriphID5)

SSI0 base: 0x4000.8000  
 SSI1 base: 0x4000.9000  
 Offset 0xFD4  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	SSI Peripheral ID Register[15:8]  Can be used by software to identify the presence of this peripheral.

### Register 13: SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8

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

#### SSI Peripheral Identification 6 (SSIPeriphID6)

SSI0 base: 0x4000.8000

SSI1 base: 0x4000.9000

Offset 0xFD8

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID6							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

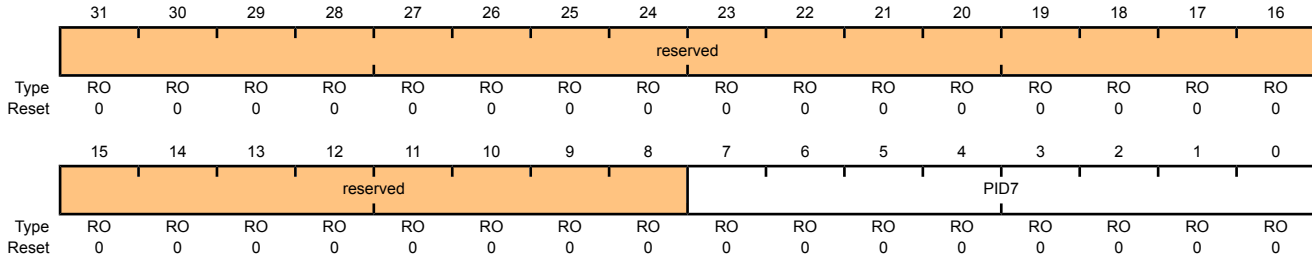
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	SSI Peripheral ID Register[23:16]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

### Register 14: SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC

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

#### SSI Peripheral Identification 7 (SSIPeriphID7)

SSI0 base: 0x4000.8000  
 SSI1 base: 0x4000.9000  
 Offset 0xFDC  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	SSI Peripheral ID Register[31:24]  Can be used by software to identify the presence of this peripheral.

## Register 15: SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0

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

### SSI Peripheral Identification 0 (SSIPeriphID0)

SSI0 base: 0x4000.8000  
 SSI1 base: 0x4000.9000  
 Offset 0xFE0  
 Type RO, reset 0x0000.0022

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0

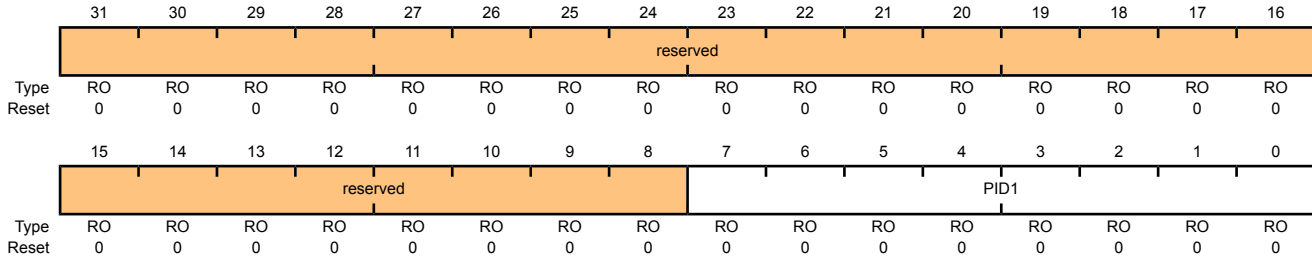
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x22	SSI Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

### Register 16: SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4

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

#### SSI Peripheral Identification 1 (SSIPeriphID1)

SSI0 base: 0x4000.8000  
 SSI1 base: 0x4000.9000  
 Offset 0xFE4  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	SSI Peripheral ID Register [15:8]  Can be used by software to identify the presence of this peripheral.



## Register 17: SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8

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

### SSI Peripheral Identification 2 (SSIPeriphID2)

SSI0 base: 0x4000.8000

SSI1 base: 0x4000.9000

Offset 0xFE8

Type RO, reset 0x0000.0018

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

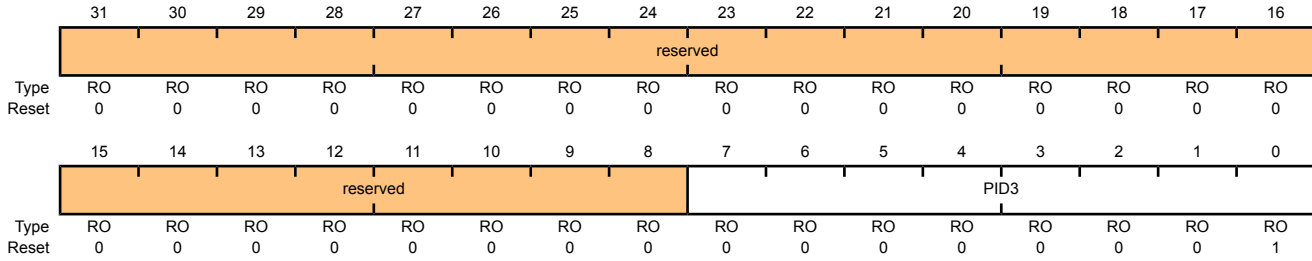
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	SSI Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

### Register 18: SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC

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

#### SSI Peripheral Identification 3 (SSIPeriphID3)

SSI0 base: 0x4000.8000  
 SSI1 base: 0x4000.9000  
 Offset 0xFEC  
 Type RO, reset 0x0000.0001



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

## Register 19: SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0

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

### SSI PrimeCell Identification 0 (SSIPCellID0)

SSI0 base: 0x4000.8000  
 SSI1 base: 0x4000.9000  
 Offset 0xFF0  
 Type RO, reset 0x0000.000D

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved								CID0								
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

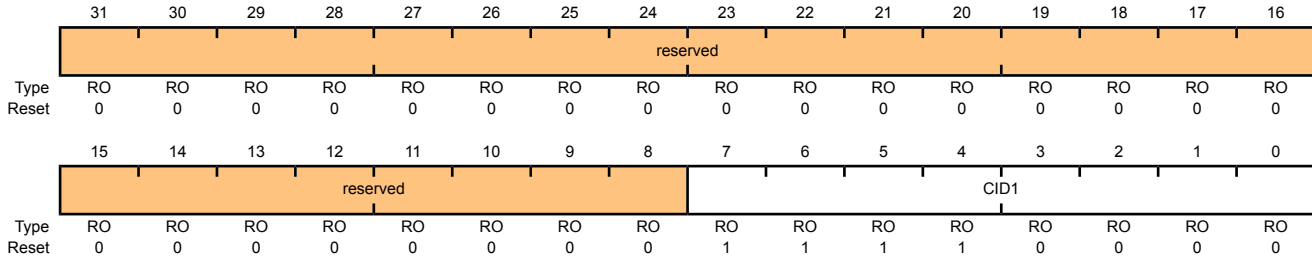
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	SSI PrimeCell ID Register [7:0] Provides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

### Register 20: SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4

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

#### SSI PrimeCell Identification 1 (SSIPCellID1)

SSI0 base: 0x4000.8000  
 SSI1 base: 0x4000.9000  
 Offset 0xFF4  
 Type RO, reset 0x0000.00F0



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

## Register 21: SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8

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

### SSI PrimeCell Identification 2 (SSIPCellID2)

SSI0 base: 0x4000.8000  
 SSI1 base: 0x4000.9000  
 Offset 0xFF8  
 Type RO, reset 0x0000.0005

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

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

### Register 22: SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC

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

#### SSI PrimeCell Identification 3 (SSIPCellID3)

SSI0 base: 0x4000.8000  
 SSI1 base: 0x4000.9000  
 Offset 0xFFC  
 Type RO, reset 0x0000.00B1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID3							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	SSI PrimeCell ID Register [31:24]  Provides software a standard cross-peripheral identification system.

## 16 Inter-Integrated Circuit (I<sup>2</sup>C) Interface

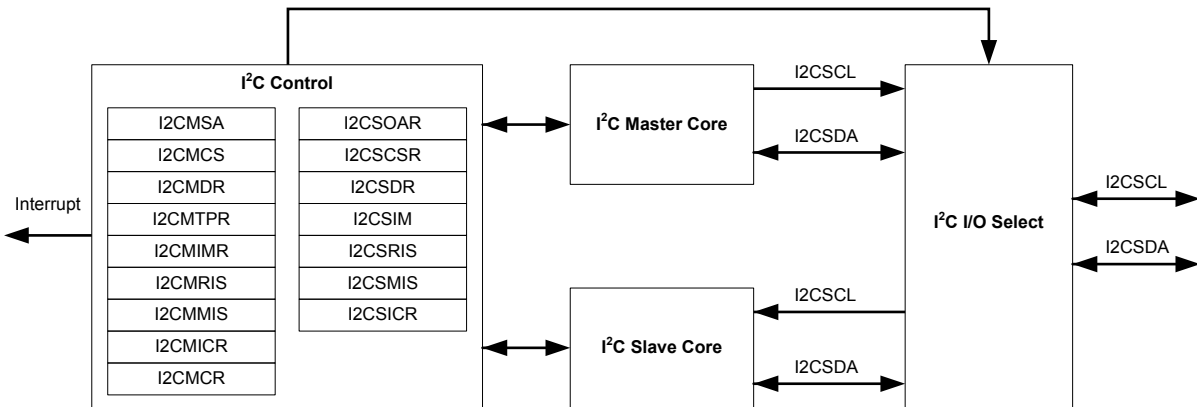
The Inter-Integrated Circuit (I<sup>2</sup>C) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL), and interfaces to external I<sup>2</sup>C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The I<sup>2</sup>C bus may also be used for system testing and diagnostic purposes in product development and manufacture. The LM3S5791 microcontroller includes two I<sup>2</sup>C modules, providing the ability to interact (both send and receive) with other I<sup>2</sup>C devices on the bus.

The Stellaris<sup>®</sup> I<sup>2</sup>C interface has the following features:

- Devices on the I<sup>2</sup>C bus can be designated as either a master or a slave
  - Supports both sending and receiving data as either a master or a slave
  - Supports simultaneous master and slave operation
- Four I<sup>2</sup>C modes
  - Master transmit
  - Master receive
  - Slave transmit
  - Slave receive
- Two transmission speeds: Standard (100 Kbps) and Fast (400 Kbps)
- Master and slave interrupt generation
  - Master generates interrupts when a transmit or receive operation completes (or aborts due to an error)
  - Slave generates interrupts when data has been sent or requested by a master or when a START or STOP condition is detected
- Master with arbitration and clock synchronization, multimaster support, and 7-bit addressing mode

## 16.1 Block Diagram

Figure 16-1. I<sup>2</sup>C Block Diagram

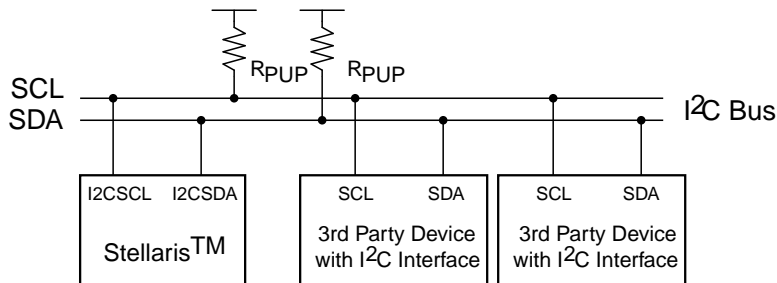


## 16.2 Functional Description

Each I<sup>2</sup>C module is comprised of both master and slave functions which are implemented as separate peripherals. For proper operation, the SDA and SCL pins must be connected to bi-directional open-drain pads. A typical I<sup>2</sup>C bus configuration is shown in Figure 16-2 on page 604.

See “Inter-Integrated Circuit (I<sup>2</sup>C) Interface” on page 962 for I<sup>2</sup>C timing diagrams.

Figure 16-2. I<sup>2</sup>C Bus Configuration



### 16.2.1 I<sup>2</sup>C Bus Functional Overview

The I<sup>2</sup>C bus uses only two signals: SDA and SCL, named I2CSDA and I2CSCL on Stellaris<sup>®</sup> microcontrollers. SDA is the bi-directional serial data line and SCL is the bi-directional serial clock line. The bus is considered idle when both lines are High.

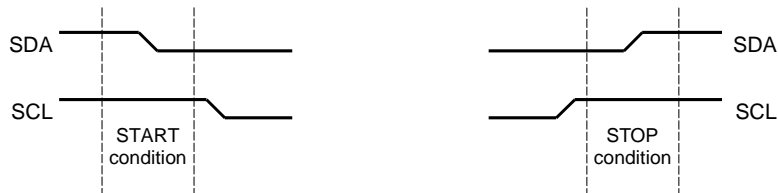
Every transaction on the I<sup>2</sup>C bus is nine bits long, consisting of eight data bits and a single acknowledge bit. The number of bytes per transfer (defined as the time between a valid START and STOP condition, described in “START and STOP Conditions” on page 605) is unrestricted, but each byte has to be followed by an acknowledge bit, and data must be transferred MSB first. When a receiver cannot receive another complete byte, it can hold the clock line SCL Low and force the transmitter into a wait state. The data transfer continues when the receiver releases the clock SCL.



### 16.2.1.1 START and STOP Conditions

The protocol of the I<sup>2</sup>C bus defines two states to begin and end a transaction: START and STOP. A High-to-Low transition on the SDA line while the SCL is High is defined as a START condition, and a Low-to-High transition on the SDA line while SCL is High is defined as a STOP condition. The bus is considered busy after a START condition and free after a STOP condition. See Figure 16-3 on page 605.

**Figure 16-3. START and STOP Conditions**

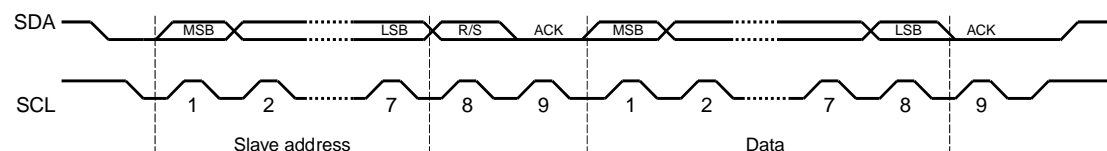


When operating in slave mode, two bits in the **I2CSRIS** register indicate detection of start and stop conditions on the bus; while two bits in the **I2CSMIS** register allow start and stop conditions to be promoted to controller interrupts (when interrupts are enabled).

### 16.2.1.2 Data Format with 7-Bit Address

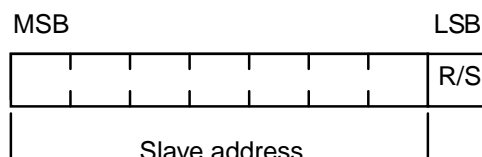
Data transfers follow the format shown in Figure 16-4 on page 605. After the START condition, a slave address is sent. This address is 7-bits long followed by an eighth bit, which is a data direction bit ( $R/S$  bit in the **I2CMSA** register). A zero indicates a transmit operation (send), and a one indicates a request for data (receive). A data transfer is always terminated by a STOP condition generated by the master, however, a master can initiate communications with another device on the bus by generating a repeated START condition and addressing another slave without first generating a STOP condition. Various combinations of receive/send formats are then possible within a single transfer.

**Figure 16-4. Complete Data Transfer with a 7-Bit Address**



The first seven bits of the first byte make up the slave address (see Figure 16-5 on page 605). The eighth bit determines the direction of the message. A zero in the  $R/S$  position of the first byte means that the master will write (send) data to the selected slave, and a one in this position means that the master will receive data from the slave.

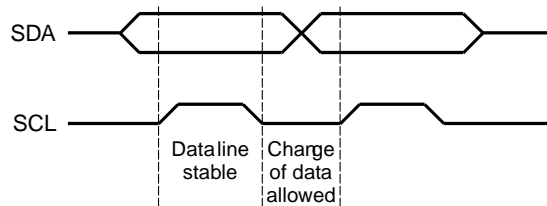
**Figure 16-5. R/S Bit in First Byte**



### 16.2.1.3 Data Validity

The data on the SDA line must be stable during the high period of the clock, and the data line can only change when SCL is Low (see Figure 16-6 on page 606).

**Figure 16-6. Data Validity During Bit Transfer on the I<sup>2</sup>C Bus**



### 16.2.1.4 Acknowledge

All bus transactions have a required acknowledge clock cycle that is generated by the master. During the acknowledge cycle, the transmitter (which can be the master or slave) releases the SDA line. To acknowledge the transaction, the receiver must pull down SDA during the acknowledge clock cycle. The data sent out by the receiver during the acknowledge cycle must comply with the data validity requirements described in “Data Validity” on page 606.

When a slave receiver does not acknowledge the slave address, SDA must be left High by the slave so that the master can generate a STOP condition and abort the current transfer. If the master device is acting as a receiver during a transfer, it is responsible for acknowledging each transfer made by the slave. Since the master controls the number of bytes in the transfer, it signals the end of data to the slave transmitter by not generating an acknowledge on the last data byte. The slave transmitter must then release SDA to allow the master to generate the STOP or a repeated START condition.

### 16.2.1.5 Arbitration

A master may start a transfer only if the bus is idle. It's possible for two or more masters to generate a START condition within minimum hold time of the START condition. In these situations, an arbitration scheme takes place on the SDA line, while SCL is High. During arbitration, the first of the competing master devices to place a '1' (High) on SDA while another master transmits a '0' (Low) will switch off its data output stage and retire until the bus is idle again.

Arbitration can take place over several bits. Its first stage is a comparison of address bits, and if both masters are trying to address the same device, arbitration continues on to the comparison of data bits.

## 16.2.2 Available Speed Modes

The I<sup>2</sup>C clock rate is determined by the parameters: CLK\_PRD, TIMER\_PRD, SCL\_LP, and SCL\_HP. where:

CLK\_PRD is the system clock period

SCL\_LP is the low phase of SCL (fixed at 6)

SCL\_HP is the high phase of SCL (fixed at 4)

TIMER\_PRD is the programmed value in the **I<sup>2</sup>C Master Timer Period (I2CMTPR)** register (see page 624).

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 16-1 on page 607 gives examples of timer period, system clock, and speed mode (Standard or Fast).

**Table 16-1. Examples of I<sup>2</sup>C Master Timer Period versus Speed Mode**

System Clock	Timer Period	Standard Mode	Timer Period	Fast Mode
4 MHz	0x01	100 Kbps	-	-
6 MHz	0x02	100 Kbps	-	-
12.5 MHz	0x06	89 Kbps	0x01	312 Kbps
16.7 MHz	0x08	93 Kbps	0x02	278 Kbps
20 MHz	0x09	100 Kbps	0x02	333 Kbps
25 MHz	0x0C	96.2 Kbps	0x03	312 Kbps
33 MHz	0x10	97.1 Kbps	0x04	330 Kbps
40 MHz	0x13	100 Kbps	0x04	400 Kbps
50 MHz	0x18	100 Kbps	0x06	357 Kbps

### 16.2.3 Interrupts

The I<sup>2</sup>C can generate interrupts when the following conditions are observed:

- Master transaction completed
- Master transaction error
- Slave transaction received
- Slave transaction requested
- Stop condition on bus detected
- Start condition on bus detected

There is a separate interrupt signal for the I<sup>2</sup>C master and I<sup>2</sup>C slave modules. While both modules can generate interrupts for multiple conditions, only a single interrupt signal is sent to the interrupt controller.

#### 16.2.3.1 I<sup>2</sup>C Master Interrupts

The I<sup>2</sup>C master module generates an interrupt when a transaction completes (either transmit or receive), or when an error occurs during a transaction. To enable the I<sup>2</sup>C master interrupt, software must write a '1' to the **I<sup>2</sup>C Master Interrupt Mask (I2CMIMR)** register. When an interrupt condition

is met, software must check the `ERROR` bit in the **I<sup>2</sup>C Master Control/Status (I2CMCS)** register to verify that an error didn't occur during the last transaction. An error condition is asserted if the last transaction wasn't acknowledge by the slave or if the master was forced to give up ownership of the bus due to a lost arbitration round with another master. If an error is not detected, the application can proceed with the transfer. The interrupt is cleared by writing a '1' to the **I<sup>2</sup>C Master Interrupt Clear (I2CMICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I<sup>2</sup>C Master Raw Interrupt Status (I2CMRIS)** register.

### 16.2.3.2 I<sup>2</sup>C Slave Interrupts

The slave module can generate an interrupt when data has been received or requested. This interrupt is enabled by writing a 1 to the `DATAIM` bit in the **I<sup>2</sup>C Slave Interrupt Mask (I2CSIMR)** register. Software determines whether the module should write (transmit) or read (receive) data from the **I<sup>2</sup>C Slave Data (I2CSDR)** register, by checking the `RREQ` and `TREQ` bits of the **I<sup>2</sup>C Slave Control/Status (I2CSCSR)** register. If the slave module is in receive mode and the first byte of a transfer is received, the `FBR` bit is set along with the `RREQ` bit. The interrupt is cleared by writing a 1 to the `DATAIC` bit in the **I<sup>2</sup>C Slave Interrupt Clear (I2CSICR)** register.

In addition, the slave module can generate an interrupt when a start and stop condition is detected. These interrupts are enabled by writing a 1 to the `STARTIM` and `STOPIM` bits of the **I<sup>2</sup>C Slave Interrupt Mask (I2CSIMR)** register and cleared by writing a 1 to the `STOPIC` and `STARTIC` bits of the **I<sup>2</sup>C Slave Interrupt Clear (I2CSICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I<sup>2</sup>C Slave Raw Interrupt Status (I2CSRIS)** register.

### 16.2.4 Loopback Operation

The I<sup>2</sup>C modules can be placed into an internal loopback mode for diagnostic or debug work. This is accomplished by setting the `LPBK` bit in the **I<sup>2</sup>C Master Configuration (I2CMCR)** register. In loopback mode, the SDA and SCL signals from the master and slave modules are tied together.

### 16.2.5 Command Sequence Flow Charts

This section details the steps required to perform the various I<sup>2</sup>C transfer types in both master and slave mode.

#### 16.2.5.1 I<sup>2</sup>C Master Command Sequences

The figures that follow show the command sequences available for the I<sup>2</sup>C master.

Figure 16-7. Master Single SEND

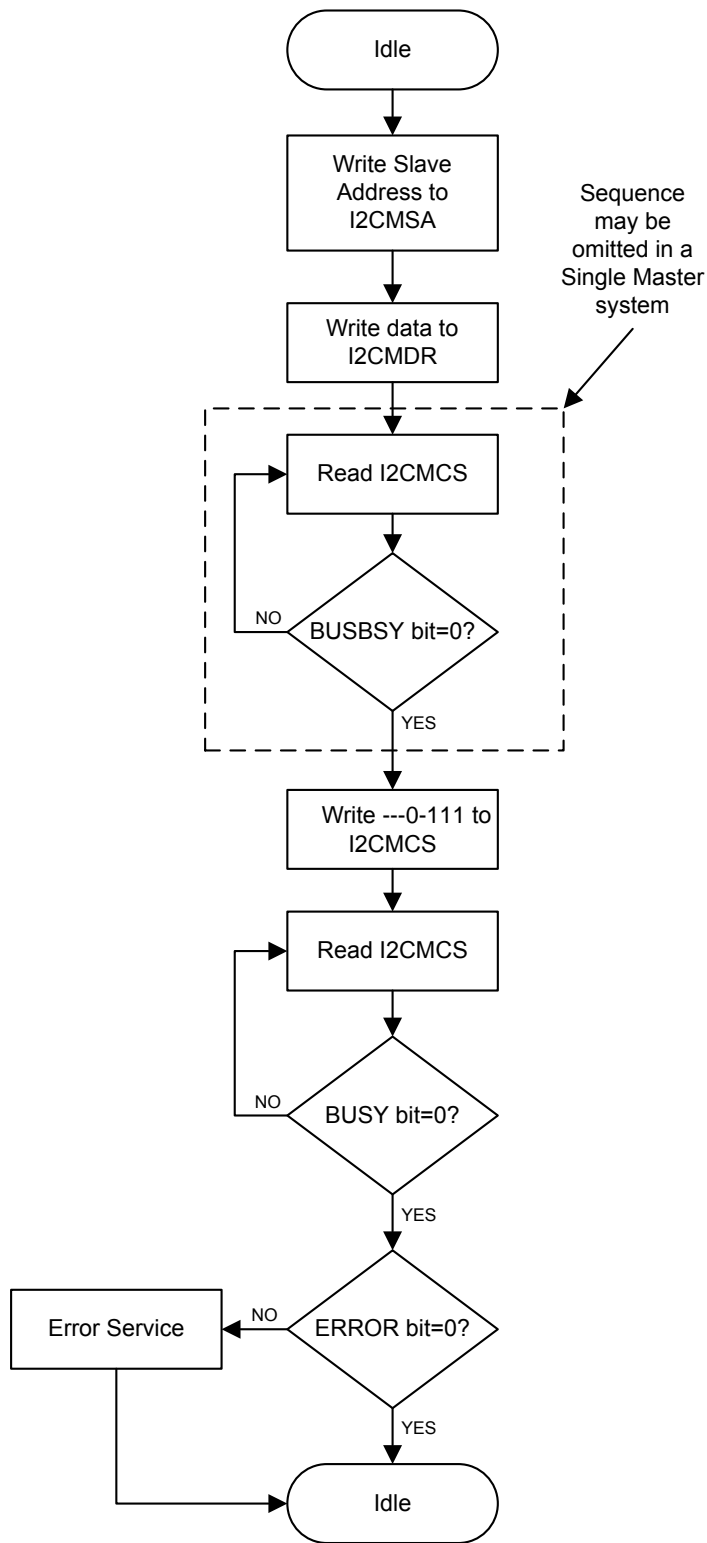


Figure 16-8. Master Single RECEIVE

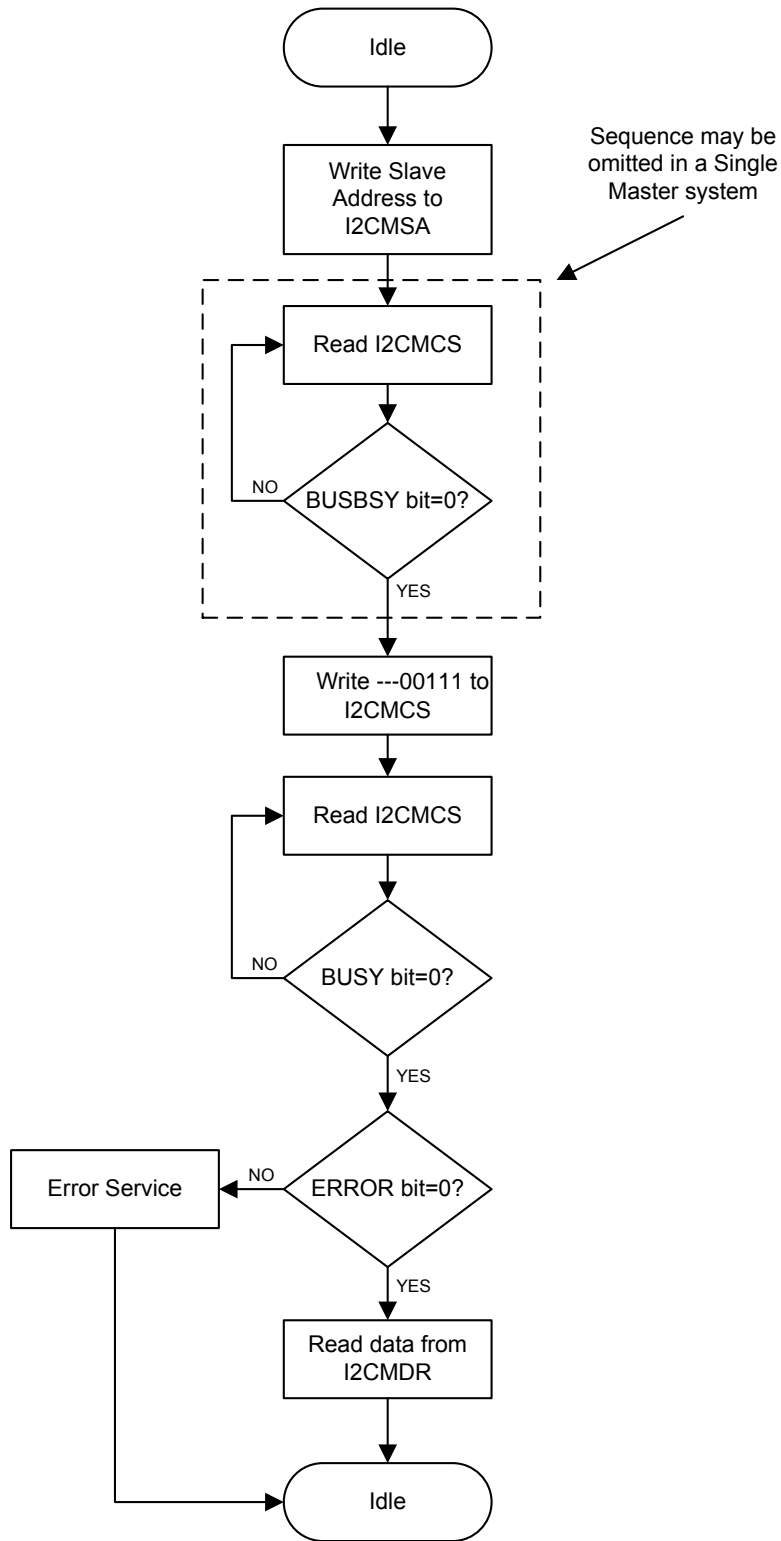


Figure 16-9. Master Burst SEND

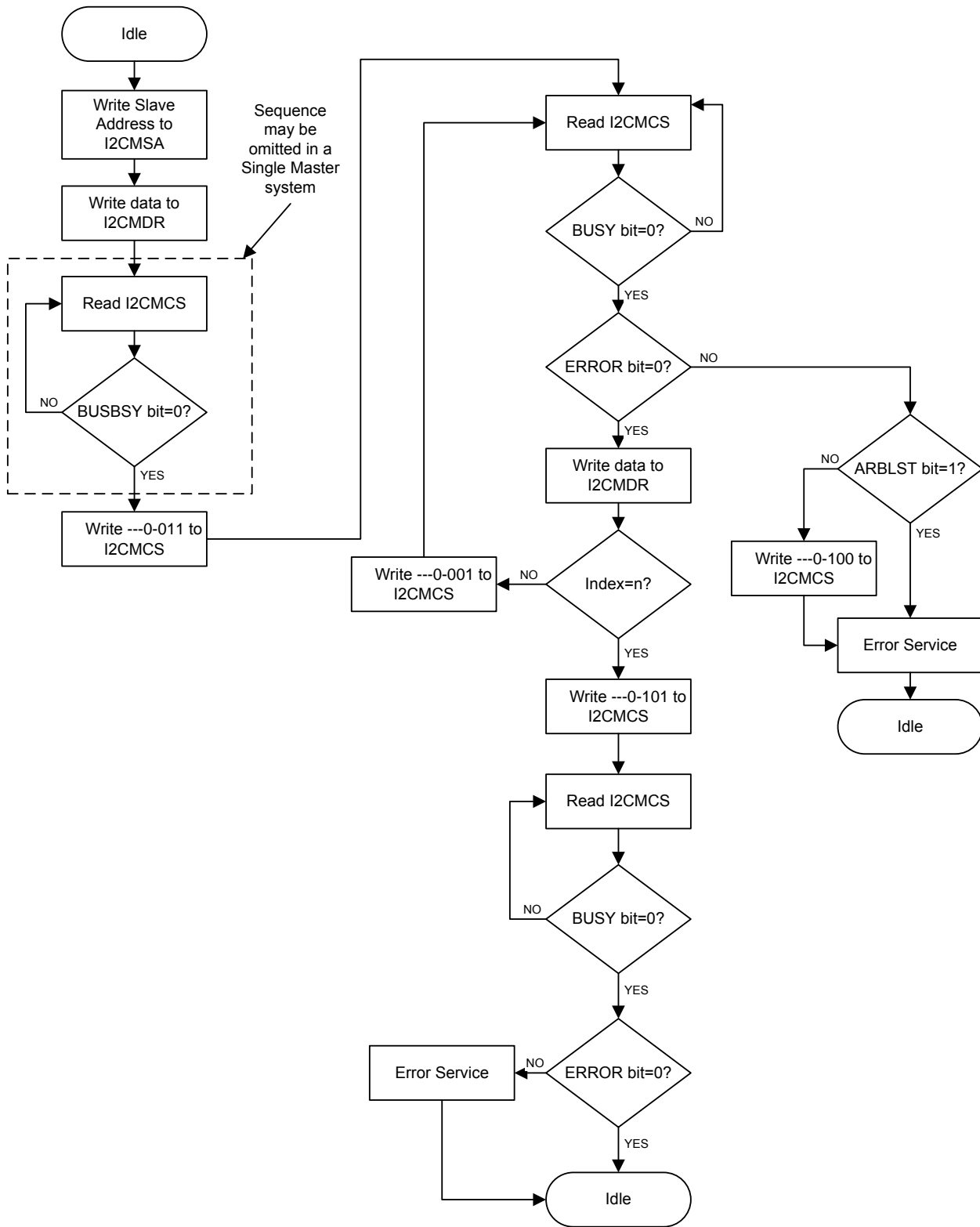


Figure 16-10. Master Burst RECEIVE

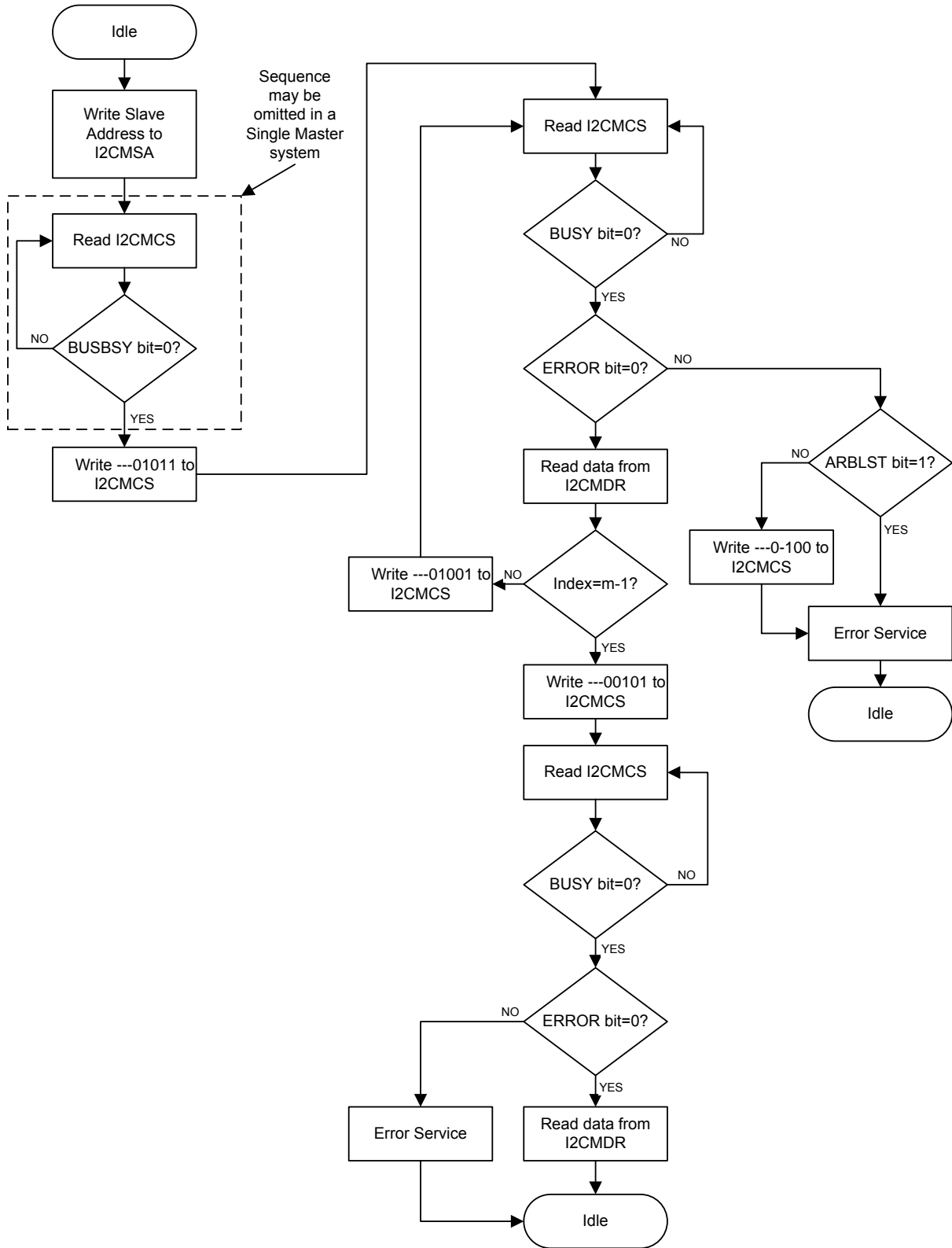
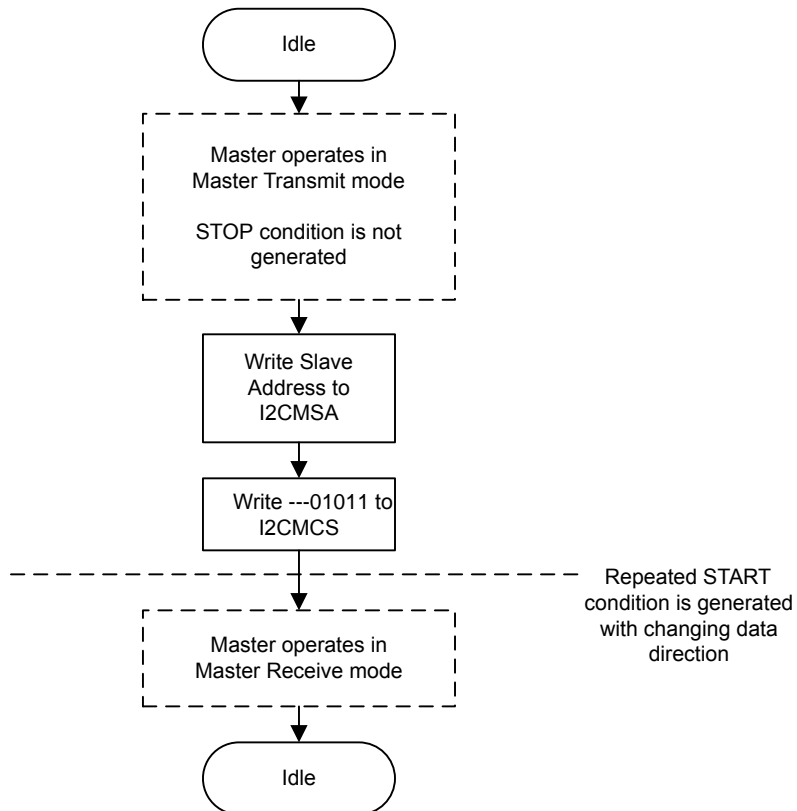
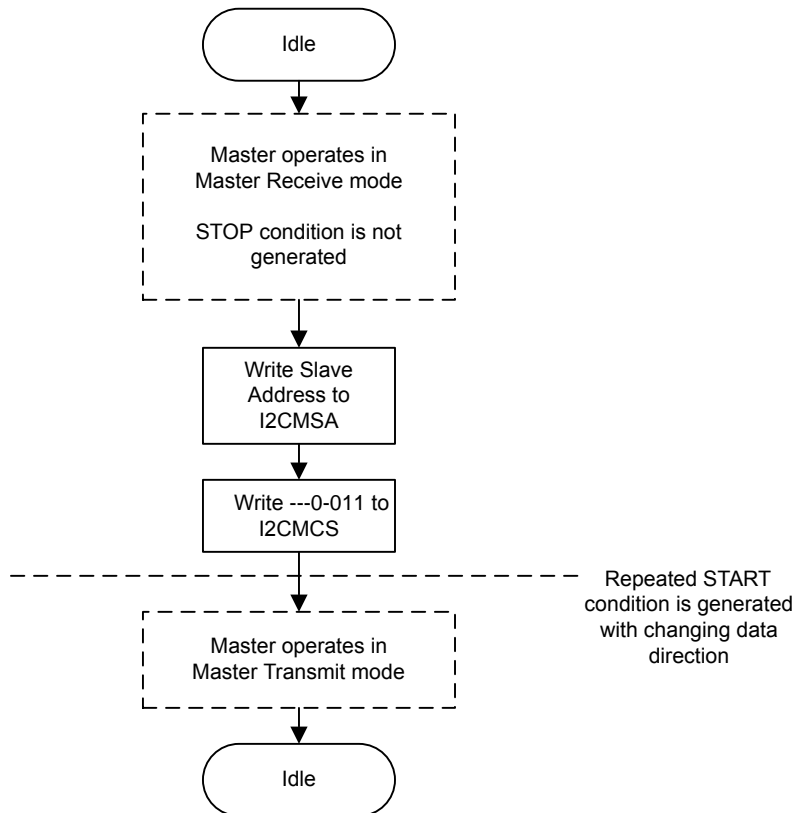




Figure 16-11. Master Burst RECEIVE after Burst SEND



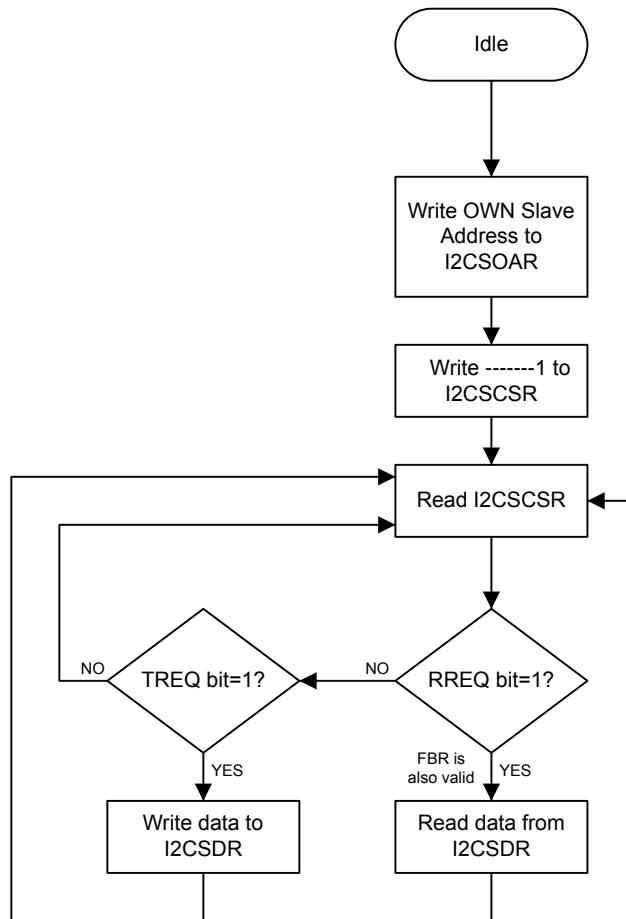
**Figure 16-12. Master Burst SEND after Burst RECEIVE**



### 16.2.5.2 I<sup>2</sup>C Slave Command Sequences

Figure 16-13 on page 615 presents the command sequence available for the I<sup>2</sup>C slave.

Figure 16-13. Slave Command Sequence



## 16.3 Initialization and Configuration

The following example shows how to configure the I<sup>2</sup>C module to send a single byte as a master. This assumes the system clock is 20 MHz.

1. Enable the I<sup>2</sup>C clock by writing a value of 0x0000.1000 to the **RCGC1** register in the System Control module. See page 163.
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module. See page 175. To find out which GPIO port to enable, refer to Table 24-5 on page 937.
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register. Also, be sure to enable the same pins for Open Drain operation.
4. Initialize the I<sup>2</sup>C Master by writing the **I2CMCR** register with a value of 0x0000.0020.
5. Set the desired SCL clock speed of 100 Kbps by writing the **I2CMTPR** register with the correct value. The value written to the **I2CMTPR** register represents the number of system clock periods in one SCL clock period. The TPR value is determined by the following equation:

```
TPR = (System Clock / (2 * (SCL_LP + SCL_HP) * SCL_CLK)) - 1;
TPR = (20MHz / (2 * (6 + 4) * 100000)) - 1;
TPR = 9
```

Write the **I2CMTPR** register with the value of 0x0000.0009.

6. Specify the slave address of the master and that the next operation will be a Send by writing the **I2CMSA** register with a value of 0x0000.0076. This sets the slave address to 0x3B.
7. Place data (byte) to be sent in the data register by writing the **I2CMDR** register with the desired data.
8. Initiate a single byte send of the data from Master to Slave by writing the **I2CMCS** register with a value of 0x0000.0007 (STOP, START, RUN).
9. Wait until the transmission completes by polling the **I2CMCS** register's BUSBSY bit until it has been cleared.

## 16.4 Register Map

Table 16-2 on page 616 lists the I<sup>2</sup>C registers. All addresses given are relative to the I<sup>2</sup>C base addresses for the master and slave:

- I<sup>2</sup>C Master 0: 0x4002.0000
- I<sup>2</sup>C Slave 0: 0x4002.0800
- I<sup>2</sup>C Master 1: 0x4002.1000
- I<sup>2</sup>C Slave 1: 0x4002.1800

Note that the I<sup>2</sup>C module clock must be enabled before the registers can be programmed (see page 163).

**Table 16-2. Inter-Integrated Circuit (I<sup>2</sup>C) Interface Register Map**

Offset	Name	Type	Reset	Description	See page
<b>I<sup>2</sup>C Master</b>					
0x000	I2CMSA	R/W	0x0000.0000	I2C Master Slave Address	618
0x004	I2CMCS	R/W	0x0000.0000	I2C Master Control/Status	619
0x008	I2CMDR	R/W	0x0000.0000	I2C Master Data	623
0x00C	I2CMTPR	R/W	0x0000.0001	I2C Master Timer Period	624
0x010	I2CMIMR	R/W	0x0000.0000	I2C Master Interrupt Mask	625
0x014	I2CMRIS	RO	0x0000.0000	I2C Master Raw Interrupt Status	626
0x018	I2CMMIS	RO	0x0000.0000	I2C Master Masked Interrupt Status	627
0x01C	I2CMICR	WO	0x0000.0000	I2C Master Interrupt Clear	628
0x020	I2CMCR	R/W	0x0000.0000	I2C Master Configuration	629
<b>I<sup>2</sup>C Slave</b>					
0x000	I2CSOAR	R/W	0x0000.0000	I2C Slave Own Address	630

Offset	Name	Type	Reset	Description	See page
0x004	I2CSCSR	RO	0x0000.0000	I2C Slave Control/Status	631
0x008	I2CSDR	R/W	0x0000.0000	I2C Slave Data	633
0x00C	I2CSIMR	R/W	0x0000.0000	I2C Slave Interrupt Mask	634
0x010	I2CSRIS	RO	0x0000.0000	I2C Slave Raw Interrupt Status	635
0x014	I2CSMIS	RO	0x0000.0000	I2C Slave Masked Interrupt Status	636
0x018	I2CSICR	WO	0x0000.0000	I2C Slave Interrupt Clear	637

## 16.5 Register Descriptions (I<sup>2</sup>C Master)

The remainder of this section lists and describes the I<sup>2</sup>C master registers, in numerical order by address offset. See also “Register Descriptions (I<sup>2</sup>C Slave)” on page 629.

### Register 1: I<sup>2</sup>C Master Slave Address (I2CMSA), offset 0x000

This register consists of eight bits: seven address bits (A6-A0), and a Receive/Send bit, which determines if the next operation is a Receive (High), or Send (Low).

#### I2C Master Slave Address (I2CMSA)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x000

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:1	SA	R/W	0	I <sup>2</sup> C Slave Address  This field specifies bits A6 through A0 of the slave address.
0	R/S	R/W	0	Receive/Send  The R/S bit specifies if the next operation is a Receive (High) or Send (Low).  Value Description 0 Send. 1 Receive.

## Register 2: I<sup>2</sup>C Master Control/Status (I2CMCS), offset 0x004

This register accesses four control bits when written, and accesses seven status bits when read.

The status register consists of seven bits, which when read determine the state of the I<sup>2</sup>C bus controller.

The control register consists of four bits: the RUN, START, STOP, and ACK bits. The START bit causes the generation of the START, or REPEATED START condition.

The STOP bit determines if the cycle stops at the end of the data cycle, or continues on to a burst. To generate a single send cycle, the **I<sup>2</sup>C Master Slave Address (I2CMSA)** register is written with the desired address, the R/S bit is set to 0, and the Control register is written with ACK=X (0 or 1), STOP=1, START=1, and RUN=1 to perform the operation and stop. When the operation is completed (or aborted due an error), the interrupt pin becomes active and the data may be read from the **I2CMDR** register. When the I<sup>2</sup>C module operates in Master receiver mode, the ACK bit must be set normally to logic 1. This causes the I<sup>2</sup>C bus controller to send an acknowledge automatically after each byte. This bit must be reset when the I<sup>2</sup>C bus controller requires no further data to be sent from the slave transmitter.

### Reads

#### I2C Master Control/Status (I2CMCS)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x004

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	BUSBSY	RO	0	Bus Busy  This bit specifies the state of the I <sup>2</sup> C bus. If set, the bus is busy; otherwise, the bus is idle. The bit changes based on the START and STOP conditions.
5	IDLE	RO	0	I <sup>2</sup> C Idle  This bit specifies the I <sup>2</sup> C controller state. If set, the controller is idle; otherwise the controller is not idle.
4	ARBLST	RO	0	Arbitration Lost  This bit specifies the result of bus arbitration. If set, the controller lost arbitration; otherwise, the controller won arbitration.

Bit/Field	Name	Type	Reset	Description
3	DATAACK	RO	0	Acknowledge Data  This bit specifies the result of the last data operation. If set, the transmitted data was not acknowledged; otherwise, the data was acknowledged.
2	ADRACK	RO	0	Acknowledge Address  This bit specifies the result of the last address operation. If set, the transmitted address was not acknowledged; otherwise, the address was acknowledged.
1	ERROR	RO	0	Error  This bit specifies the result of the last bus operation. If set, an error occurred on the last operation; otherwise, no error was detected. The error can be from the slave address not being acknowledged, the transmit data not being acknowledged, or because the controller lost arbitration.
0	BUSY	RO	0	I <sup>2</sup> C Busy  This bit specifies the state of the controller. If set, the controller is busy; otherwise, the controller is idle. When the <code>BUSY</code> bit is set, the other status bits are not valid.

**Writes**

**I2C Master Control/Status (I2CMCS)**

I2C Master 0 base: 0x4002.0000  
 I2C Master 1 base: 0x4002.1000  
 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 16-3 on page 621.
2	STOP	WO	0	Generate STOP  When set, causes the generation of the STOP condition. See field decoding in Table 16-3 on page 621.



Bit/Field	Name	Type	Reset	Description
1	START	WO	0	Generate START When set, causes the generation of a START or repeated START condition. See field decoding in Table 16-3 on page 621.
0	RUN	WO	0	I <sup>2</sup> C Master Enable When set, allows the master to send or receive data. See field decoding in Table 16-3 on page 621.

**Table 16-3. Write Field Decoding for I2CMCS[3:0] Field (Sheet 1 of 3)**

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
	R/S	ACK	STOP	START	RUN	
Idle	0	X <sup>a</sup>	0	1	1	START condition followed by SEND (master goes to the Master Transmit state).
	0	X	1	1	1	START condition followed by a SEND and STOP condition (master remains in Idle state).
	1	0	0	1	1	START condition followed by RECEIVE operation with negative ACK (master goes to the Master Receive state).
	1	0	1	1	1	START condition followed by RECEIVE and STOP condition (master remains in Idle state).
	1	1	0	1	1	START condition followed by RECEIVE (master goes to the Master Receive state).
	1	1	1	1	1	Illegal.
	All other combinations not listed are non-operations.					
Master Transmit	X	X	0	0	1	SEND operation (master remains in Master Transmit state).
	X	X	1	0	0	STOP condition (master goes to Idle state).
	X	X	1	0	1	SEND followed by STOP condition (master goes to Idle state).
	0	X	0	1	1	Repeated START condition followed by a SEND (master remains in Master Transmit state).
	0	X	1	1	1	Repeated START condition followed by SEND and STOP condition (master goes to Idle state).
	1	0	0	1	1	Repeated START condition followed by a RECEIVE operation with a negative ACK (master goes to Master Receive state).
	1	0	1	1	1	Repeated START condition followed by a SEND and STOP condition (master goes to Idle state).
	1	1	0	1	1	Repeated START condition followed by RECEIVE (master goes to Master Receive state).
	1	1	1	1	1	Illegal.
	All other combinations not listed are non-operations.					

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
	R/S	ACK	STOP	START	RUN	
Master Receive	X	0	0	0	1	RECEIVE operation with negative ACK (master remains in Master Receive state).
	X	X	1	0	0	STOP condition (master goes to Idle state). <sup>b</sup>
	X	0	1	0	1	RECEIVE followed by STOP condition (master goes to Idle state).
	X	1	0	0	1	RECEIVE operation (master remains in Master Receive state).
	X	1	1	0	1	Illegal.
	1	0	0	1	1	Repeated START condition followed by RECEIVE operation with a negative ACK (master remains in Master Receive state).
	1	0	1	1	1	Repeated START condition followed by RECEIVE and STOP condition (master goes to Idle state).
	1	1	0	1	1	Repeated START condition followed by RECEIVE (master remains in Master Receive state).
	0	X	0	1	1	Repeated START condition followed by SEND (master goes to Master Transmit state).
	0	X	1	1	1	Repeated START condition followed by SEND and STOP condition (master goes to Idle state).
All other combinations not listed are non-operations.						NOP.

a. An X in a table cell indicates the bit can be 0 or 1.

b. In Master Receive mode, a STOP condition should be generated only after a Data Negative Acknowledge executed by the master or an Address Negative Acknowledge executed by the slave.

### Register 3: I<sup>2</sup>C Master Data (I2CMDR), offset 0x008

This register contains the data to be transmitted when in the Master Transmit state, and the data received when in the Master Receive state.

#### I2C Master Data (I2CMDR)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x008

Type R/W, reset 0x0000.0000

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

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

### Register 4: I<sup>2</sup>C Master Timer Period (I2CMTPR), offset 0x00C

This register specifies the period of the SCL clock.

#### I2C Master Timer Period (I2CMTPR)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x00C

Type R/W, reset 0x0000.0001

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TPR	R/W	0x1	SCL Clock Period

This field specifies the period of the SCL clock.

$$SCL\_PRD = 2 * (1 + TPR) * (SCL\_LP + SCL\_HP) * CLK\_PRD$$

where:

SCL\_PRD is the SCL line period (I<sup>2</sup>C clock).

TPR is the Timer Period register value (range of 1 to 255).

SCL\_LP is the SCL Low period (fixed at 6).

SCL\_HP is the SCL High period (fixed at 4).

## Register 5: I<sup>2</sup>C Master Interrupt Mask (I2CMIMR), offset 0x010

This register controls whether a raw interrupt is promoted to a controller interrupt.

### I2C Master Interrupt Mask (I2CMIMR)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x010

Type R/W, reset 0x0000.0000

	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	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of 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	<p>Interrupt Mask</p> <p>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.</p>

## Register 6: I<sup>2</sup>C Master Raw Interrupt Status (I2CMRIS), offset 0x014

This register specifies whether an interrupt is pending.

### I2C Master Raw Interrupt Status (I2CMRIS)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x014

Type RO, reset 0x0000.0000

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

I2C Master 1 base: 0x4002.1000

Offset 0x018

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															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	Masked Interrupt Status  This bit specifies the raw interrupt state (after masking) of the I <sup>2</sup> C master block. If set, an interrupt was signaled; otherwise, an interrupt has not been generated since the bit was last cleared.

## Register 8: I<sup>2</sup>C Master Interrupt Clear (I2CMICR), offset 0x01C

This register clears the raw interrupt.

### I2C Master Interrupt Clear (I2CMICR)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x01C

Type WO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	IC	WO	0	Interrupt Clear  This bit controls the clearing of the raw interrupt. A write of 1 clears the interrupt; otherwise, a write of 0 has no affect on the interrupt state. A read of this register returns no meaningful data.



## Register 9: I<sup>2</sup>C Master Configuration (I2CMCR), offset 0x020

This register configures the mode (Master or Slave) and sets the interface for test mode loopback.

### I2C Master Configuration (I2CMCR)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x020

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SFE	R/W	0	I <sup>2</sup> C Slave Function Enable  This bit specifies whether the interface may operate in Slave mode. If set, Slave mode is enabled; otherwise, Slave mode is disabled.
4	MFE	R/W	0	I <sup>2</sup> C Master Function Enable  This bit specifies whether the interface may operate in Master mode. If set, Master mode is enabled; otherwise, Master mode is disabled and the interface clock is disabled.
3:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	LPBK	R/W	0	I <sup>2</sup> C Loopback  This bit specifies whether the interface is operating normally or in Loopback mode. If set, the device is put in a test mode loopback configuration; otherwise, the device operates normally.

## 16.6 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 617.

### Register 10: I<sup>2</sup>C Slave Own Address (I2CSOAR), offset 0x000

This register consists of seven address bits that identify the Stellaris<sup>®</sup> I<sup>2</sup>C device on the I<sup>2</sup>C bus.

#### I2C Slave Own Address (I2CSOAR)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x000

Type R/W, reset 0x0000.0000

	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									OAR						
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
31:7	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	OAR	R/W	0x00	I <sup>2</sup> C Slave Own Address This field specifies bits A6 through A0 of the slave address.

## Register 11: I<sup>2</sup>C Slave Control/Status (I2CCSR), offset 0x004

This register accesses one control bit when written, and three status bits when read.

The read-only Status register consists of three bits: the `FBR`, `RREQ`, and `TREQ` bits. The `First Byte Received (FBR)` bit is set only after the Stellaris<sup>®</sup> device detects its own slave address and receives the first data byte from the I<sup>2</sup>C master. The `Receive Request (RREQ)` bit indicates that the Stellaris<sup>®</sup> I<sup>2</sup>C device has received a data byte from an I<sup>2</sup>C master. Read one data byte from the **I<sup>2</sup>C Slave Data (I2CSDR)** register to clear the `RREQ` bit. The `Transmit Request (TREQ)` bit indicates that the Stellaris<sup>®</sup> I<sup>2</sup>C device is addressed as a Slave Transmitter. Write one data byte into the **I<sup>2</sup>C Slave Data (I2CSDR)** register to clear the `TREQ` bit.

The write-only Control register consists of one bit: the `DA` bit. The `DA` bit enables and disables the Stellaris<sup>®</sup> I<sup>2</sup>C slave operation.

### Reads

#### I2C Slave Control/Status (I2CCSR)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x004

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	FBR	RO	0	<p>First Byte Received</p> <p>Indicates that the first byte following the slave's own address is received. This bit is only valid when the <code>RREQ</code> bit is set, and is automatically cleared when data has been read from the <b>I2CSDR</b> register.</p> <p><b>Note:</b> This bit is not used for slave transmit operations.</p>
1	TREQ	RO	0	<p>Transmit Request</p> <p>This bit specifies the state of the I<sup>2</sup>C slave with regards to outstanding transmit requests. If set, the I<sup>2</sup>C unit has been addressed as a slave transmitter and uses clock stretching to delay the master until data has been written to the <b>I2CSDR</b> register. Otherwise, there is no outstanding transmit request.</p>

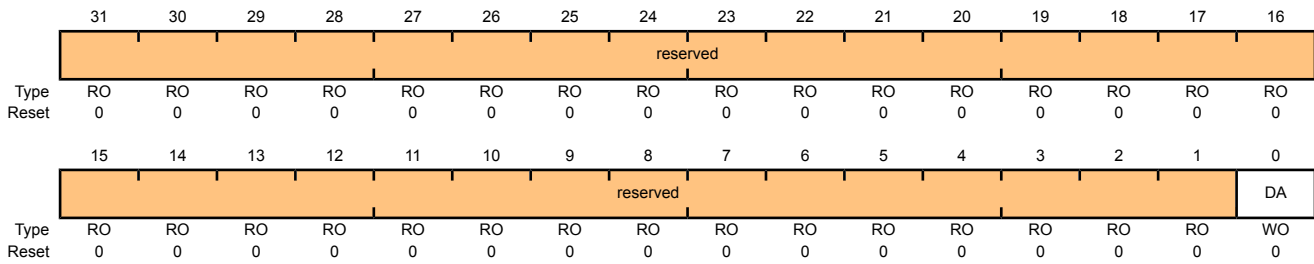
Bit/Field	Name	Type	Reset	Description
0	RREQ	RO	0	Receive Request

This bit specifies the status of the I<sup>2</sup>C slave with regards to outstanding receive requests. If set, the I<sup>2</sup>C unit has outstanding receive data from the I<sup>2</sup>C master and uses clock stretching to delay the master until the data has been read from the **I2CSDR** register. Otherwise, no receive data is outstanding.

**Writes**

**I2C Slave Control/Status (I2CSCSR)**

I2C Slave 0 base: 0x4002.0800  
 I2C Slave 1 base: 0x4002.1800  
 Offset 0x004  
 Type WO, reset 0x0000.0000



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

Value Description  
 0 Disables the I<sup>2</sup>C slave operation.  
 1 Enables the I<sup>2</sup>C slave operation.

## Register 12: I<sup>2</sup>C Slave Data (I2CSDR), offset 0x008

This register contains the data to be transmitted when in the Slave Transmit state, and the data received when in the Slave Receive state.

### I2C Slave Data (I2CSDR)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x008

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x0	Data for Transfer  This field contains the data for transfer during a slave receive or transmit operation.

### Register 13: I<sup>2</sup>C Slave Interrupt Mask (I2CSIMR), offset 0x00C

This register controls whether a raw interrupt is promoted to a controller interrupt.

#### I2C Slave Interrupt Mask (I2CSIMR)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x00C

Type R/W, reset 0x0000.0000

	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													STOPIM	STARTIM	DATAIM
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPIM	RO	0	Stop Condition Interrupt Mask  This bit controls whether the raw interrupt for detection of a stop condition on the I <sup>2</sup> C bus is promoted to a controller interrupt. If set, the interrupt is not masked and the interrupt is promoted; otherwise, the interrupt is masked.
1	STARTIM	RO	0	Start Condition Interrupt Mask  This bit controls whether the raw interrupt for detection of a start condition on the I <sup>2</sup> C bus is promoted to a controller interrupt. If set, the interrupt is not masked and the interrupt is promoted; otherwise, the interrupt is masked.
0	DATAIM	R/W	0	Data Interrupt Mask  This bit controls whether the raw interrupt for data received and data requested is promoted to a controller interrupt. If set, the interrupt is not masked and the interrupt is promoted; otherwise, the interrupt is masked.

## Register 14: I<sup>2</sup>C Slave Raw Interrupt Status (I2CSRIS), offset 0x010

This register specifies whether an interrupt is pending.

### I2C Slave Raw Interrupt Status (I2CSRIS)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x010

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPRIS	RO	0	Stop Condition Raw Interrupt Status  This bit specifies the raw interrupt state for stop condition detect (prior to masking) of the I <sup>2</sup> C slave block. If set, an interrupt is pending; otherwise, an interrupt is not pending.
1	STARTRIS	RO	0	Start Condition Raw Interrupt Status  This bit specifies the raw interrupt state for start condition detect (prior to masking) of the I <sup>2</sup> C slave block. If set, an interrupt is pending; otherwise, an interrupt is not pending.
0	DATARIS	RO	0	Data Raw Interrupt Status  This bit specifies the raw interrupt state for data received and data requested (prior to masking) of the I <sup>2</sup> C slave block. If set, an interrupt is pending; otherwise, an interrupt is not pending.

## Register 15: I<sup>2</sup>C Slave Masked Interrupt Status (I2CSMIS), offset 0x014

This register specifies whether an interrupt was signaled.

### I2C Slave Masked Interrupt Status (I2CSMIS)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x014

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPMIS	R/W	0	Stop Condition Masked Interrupt Status  This bit specifies the interrupt state for stop condition detect (after masking) of the I <sup>2</sup> C slave block. If set, an interrupt was signaled; otherwise, an interrupt has not been generated since the bit was last cleared.
1	STARTMIS	R/W	0	Start Condition Masked Interrupt Status  This bit specifies the interrupt state for start condition detect (after masking) of the I <sup>2</sup> C slave block. If set, an interrupt was signaled; otherwise, an interrupt has not been generated since the bit was last cleared.
0	DATAMIS	RO	0	Data Masked Interrupt Status  This bit specifies the interrupt state for data received and data requested (after masking) of the I <sup>2</sup> C slave block. If set, an interrupt was signaled; otherwise, an interrupt has not been generated since the bit was last cleared.



## Register 16: I<sup>2</sup>C Slave Interrupt Clear (I2CSICR), offset 0x018

This register clears the raw interrupt. A read of this register returns no meaningful data.

### I2C Slave Interrupt Clear (I2CSICR)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4002.1800

Offset 0x018

Type WO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPIC	WO	0	Stop Condition Interrupt Clear  This bit controls the clearing of the raw interrupt for stop condition detect. When set, it clears the STOPRIS interrupt bit; otherwise, it has no effect on the STOPRIS bit value.
1	STARTIC	WO	0	Start Condition Interrupt Clear  This bit controls the clearing of the raw interrupt for start condition detect. When set, it clears the STARTRIS interrupt bit; otherwise, it has no effect on the STARTRIS bit value.
0	DATAIC	WO	0	Data Interrupt Clear  This bit controls the clearing of the raw interrupt for data received and data requested. When set, it clears the DATARIS interrupt bit; otherwise, it has no effect on the DATARIS bit value.

## 17 Inter-Integrated Circuit Sound (I<sup>2</sup>S) Interface

The I<sup>2</sup>S module is a configurable serial audio core that contains a transmit module and a receive module. The module is configurable for the I<sup>2</sup>S as well as Left-Justified and Right-Justified serial audio formats. Data can be in one of four modes: Stereo, Mono, Compact 16-bit Stereo and Compact 8-Bit Stereo.

The transmit and receive modules each have an 8-entry audio-sample FIFO. An audio sample can consist of a Left and Right Stereo sample, a Mono sample, or a Left and Right Compact Stereo sample. In Compact 16-Bit Stereo, each FIFO entry contains both the 16-bit left and 16-bit right samples, allowing efficient data transfers and requiring less memory space. In Compact 8-bit Stereo, each FIFO entry contains an 8-bit left and an 8-bit right sample, reducing memory requirements further.

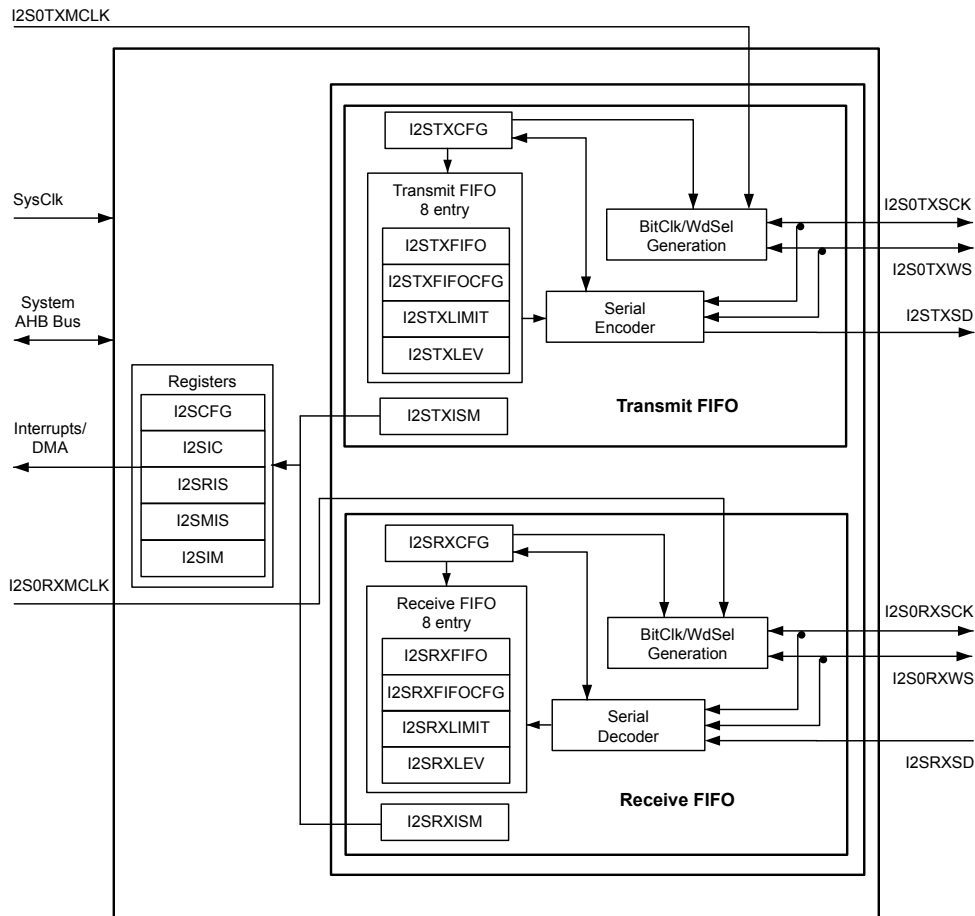
Both the transmitter and receiver are capable of being a master or a slave.

The Stellaris<sup>®</sup> I<sup>2</sup>S module has the following features:

- Configurable audio format supporting I<sup>2</sup>S, Left-justification, and Right-justification
- Configurable sample size from 8 to 32 bits
- Mono and Stereo support
- 8-, 16-, and 32-bit FIFO interface for packing memory
- Independent transmit and receive 8-entry FIFOs
- Configurable FIFO-level interrupt and  $\mu$ DMA requests
- Independent transmit and receive MCLK direction control
- Transmit and receive internal MCLK sources
- Independent transmit and receive control for serial clock and word select
- MCLK and SCLK can be independently set to master or slave
- Configurable transmit zero or last sample when FIFO empty
- Efficient transfers using Micro Direct Memory Access Controller ( $\mu$ DMA)
  - Separate channels for transmit and receive
  - Burst requests
  - Channel requests asserted when FIFO contains required amount of data

## 17.1 Block Diagram

Figure 17-1. I<sup>2</sup>S Block Diagram



## 17.2 Functional Description

The Inter-Integrated Circuit Sound (I<sup>2</sup>S) module contains separate transmit and receive engines. Each engine consists of the following:

- Serial encoder for the transmitter; serial decoder for the receiver
- 8-entry FIFO to store sample data
- Independent configuration of all programmable settings

The basic programming model of the I<sup>2</sup>S block is as follows:

- Configuration
  - Overall I<sup>2</sup>S module configuration in the **I<sup>2</sup>S Module Configuration (I2SCFG)** register. This register is used to select the MCLK source and enable the receiver and transmitter.

- Transmit and receive configuration in the **I<sup>2</sup>S Transmit Module Configuration (I2STXCFG)** and **I<sup>2</sup>S Receive Module Configuration (I2SRXCFG)** registers. These registers set the basic parameters for the receiver and transmitter such as data configuration (justification, delay, read mode, sample size, and system data size); SCLK (polarity and source); and word select polarity.
- Transmit and receive FIFO configuration in the **I<sup>2</sup>S Transmit FIFO Configuration (I2STXFIFOCFG)** and **I<sup>2</sup>S Receive FIFO Configuration (I2SRXFIFOCFG)** registers. These registers select the Compact Stereo mode size (16-bit or 8-bit), provide indication of whether the next sample is Left or Right, and select mono mode for the receiver.
- FIFO
  - Transmit and receive FIFO data in the **I<sup>2</sup>S Transmit FIFO Data (I2STXFIFO)** and **I<sup>2</sup>S Receive FIFO Data (I2SRXFIFO)** registers
  - Information on FIFO data levels in the **I<sup>2</sup>S Transmit FIFO Level (I2STXLEV)** and **I<sup>2</sup>S Receive FIFO Level (I2SRXLEV)** registers
  - Configuration for FIFO service requests based on FIFO levels in the **I<sup>2</sup>S Transmit FIFO Limit (I2STXLIMIT)** and **I<sup>2</sup>S Receive FIFO Limit (I2SRXLIM)** registers
- Interrupt Control
  - Interrupt masking configuration in the **I<sup>2</sup>S Interrupt Mask (I2SIM)** register
  - Raw and masked interrupt status in the **I<sup>2</sup>S Raw Interrupt Status (I2SRIS)** and **I<sup>2</sup>S Masked Interrupt Status (I2SMIS)** registers
  - Interrupt clearing through the **I<sup>2</sup>S Interrupt Clear (I2SIC)** register
  - Configuration for FIFO service requests interrupts and transmit/receive error interrupts in the **I<sup>2</sup>S Transmit Interrupt Status and Mask (I2STXISM)** and **I<sup>2</sup>S Receive Interrupt Status and Mask (I2SRXISM)** registers

Figure 17-2 on page 641 provides an example of an I<sup>2</sup>S data transfer. Figure 17-3 on page 641 provides an example of an Left-Justified data transfer. Figure 17-4 on page 641 provides an example of an Right-Justified data transfer.

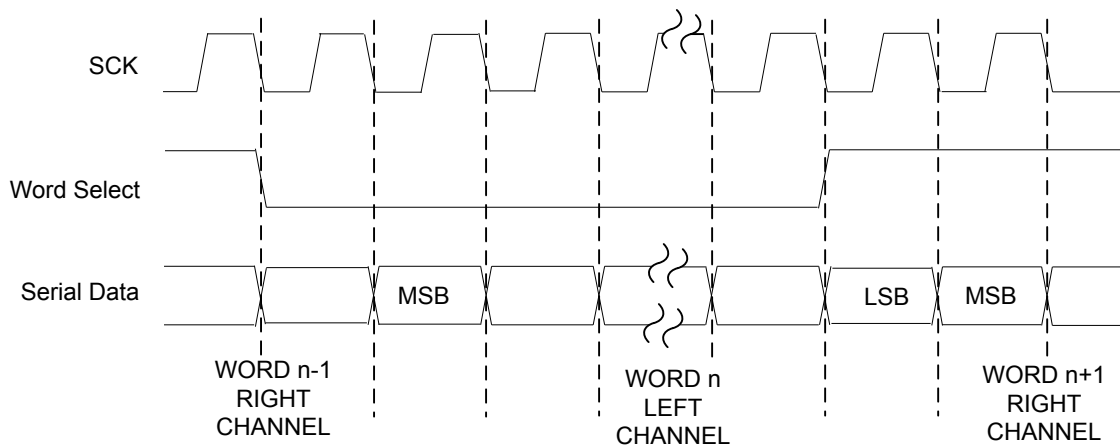
Figure 17-2. I<sup>2</sup>S Data Transfer

Figure 17-3. Left-Justified Data Transfer

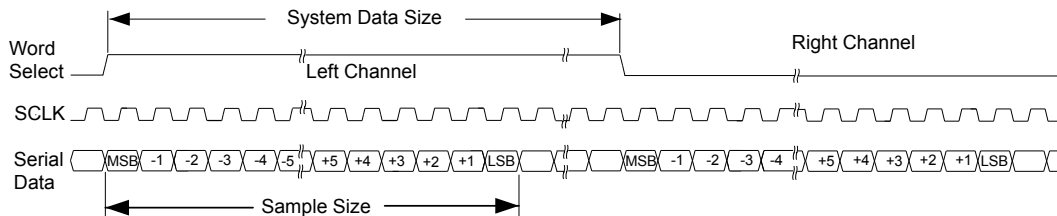
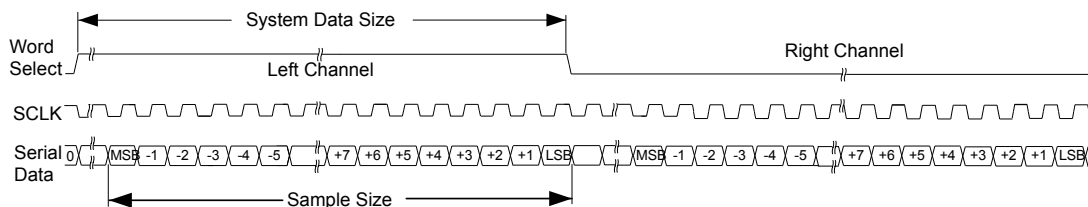


Figure 17-4. Right-Justified Data Transfer



## 17.2.1 Transmit

The transmitter consists of a serial encoder, an 8-entry FIFO, and control logic. The transmitter has independent MCLK (I2S0TXMCLK), SCLK (I2S0TXSCK), and Word-Select (I2S0TXWS) signals.

### 17.2.1.1 Serial Encoder

The serial encoder reads audio samples from the receive FIFO and converts them into an audio stream. By configuring the serial encoder, common audio formats I<sup>2</sup>S, Left-Justified, and Right-Justified are supported. The MSB is transmitted first. The sample size and system data size are configurable with the SSZ and SDSZ bits in the **I<sup>2</sup>S Transmit Module Configuration (I2STXCFG)** register. The sample size is the number of bits of data being transmitted, and the system data size is the number of I2S0TXSCK transitions between the word select transitions. The system data size must be large enough to accommodate the maximum sample size. In Mono mode, the sample data is repeated in both the left and right channels. When the FIFO is empty, the user may select either

transmission of zeros or of the last sample. The serial encoder is enabled using the `TXEN` bit in the **I<sup>2</sup>S Module Configuration (I2SCFG)** register.

### 17.2.1.2 FIFO Operation

The transmit FIFO stores eight Mono samples or eight Stereo sample-pairs of data and is accessed through the **I<sup>2</sup>S Transmit FIFO Data (I2STXFIFO)** register. The FIFO interface for the audio data is different based on the Write mode, defined by the **I<sup>2</sup>S Transmit FIFO Configuration (I2STXFIFOCFG)** Compact Stereo Sample Size bit (`CSS`) and the **I2STXCFG** Write Mode field (`WM`). All data samples are MSB-aligned. Table 17-1 on page 642 defines the interface for each Write mode. Stereo samples are written first left then right. The next sample (right or left) to be written is indicated by the `LRS` bit in the **I2STXFIFOCFG** register.

**Table 17-1. I<sup>2</sup>S Transmit FIFO Interface**

<code>WM</code> field in <b>I2STXCFG</b>	<code>CSS</code> bit in <b>I2STXFIFOCFG</b>	Write Mode	Sample Width	Samples per FIFO Write	Data Alignment
0x0	don't care	Stereo	8-32 bits	1	MSB
0x1	0	Compact Stereo - 16 bit	8-16 bits	2	MSB Right [31:16], Left [15:0]
0x1	1	Compact Stereo - 8 bit	8 bits	2	Right [15:8], Left[7:0]
0x2	don't care	Mono	8-32 bits	1	MSB

The number of samples in the transmit FIFO can be read using the **I<sup>2</sup>S Transmit FIFO Level (I2STXLEV)** register. The value ranges from 0 to 16. Stereo and compact stereo sample pairs are counted as two. The mono samples also increment the count by two, therefore, four mono samples will have a count of eight.

### 17.2.1.3 Clock Control

The transmitter `MCLK` and `SCLK` can be independently programmed to be the master or slave. The transmitter is programmed to be the master or slave of the `SCLK` using the `MSL` bit in the **I2STXCFG** register. When the transmitter is the master, the `I2S0TXSCK` frequency is the specified `I2S0TXMCLK` divided by four. The `I2S0TXSCK` may be inverted using the `SCP` bit in the **I2STXCFG** register.

The transmitter can also be the master or slave of the `MCLK`. When the transmitter is the master, the PLL must be active and a fractional clock divider must be programmed. See page 125 for the setup for the master `I2S0TXMCLK` source. An external transmit `I2S0TXMCLK` is selected using the `TXSLV` bit in the **I2SCFG** register.

The following tables show combinations of the `TXINT` and `TXFRAC` bits in the **I<sup>2</sup>S MCLK Configuration (I2SMCLKCFG)** register that provide `MCLK` frequencies within acceptable error limits. In the table, `Fs` is the sampling frequency in kHz and possible crystal frequencies are shown in MHz across the top row of the table. The words "not supported" in the table mean that it is not possible to obtain the specified sampling frequencies with the specified crystal frequency within the error tolerance of 0.3%. The values in the table are based on the following values:

$$\text{MCLK} = F_s * 256$$

$$\text{VCO} = 400 \text{ MHz}$$

The Integer value is taken from the result of the following calculation:

$$\text{ROUND}(\text{VCO} / \text{MCLK})$$

The remaining fractional component is converted to binary, and the first four bits are the Fractional value.

**Table 17-2. Crystal Frequency (Values from 3.5795 MHz to 5 MHz)**

Fs (kHz)	Crystal Frequency (MHz)											
	3.5795		3.6864		4.0000		4.0960		4.9152		5.0000	
	Integer	Fractional	Integer	Fractional	Integer	Fractional	Integer	Fractional	Integer	Fractional	Integer	Fractional
8	195	11	194	6	195	5	196	0	194	6	195	5
11.025	142	1	141	1	141	11	142	4	141	1	141	11
16	97	13	97	3	97	9	98	0	97	3	97	9
22.05	71	0	70	7	70	13	71	2	70	7	70	13
24	65	4	64	12	65	2	65	5	64	12	65	2
32	48	14	48	9	48	12	49	0	48	9	48	12
44.056	35	8	35	5	35	7	35	8	35	5	35	7
44.1	35	7	35	4	35	7	35	8	35	4	35	7
47.25	33	2	32	14	33	1	33	3	32	14	33	1
48	32	9	32	6	32	8	32	10	32	6	32	8
50	31	5	31	2	31	4	31	6	31	2	31	4
88.2	17	11	17	9	17	10	17	11	17	9	17	10
96	16	5	16	3	16	4	16	5	16	3	16	4
128	12	4	12	2	12	3	12	4	12	2	12	3
176.4	8	13	8	12	8	13	8	13	8	12	8	13
192	Not supported		Not supported		8	2	8	3	Not supported		8	2

**Table 17-3. Crystal Frequency (Values from 5.12 MHz to 8.192 MHz)**

Fs (kHz)	Crystal Frequency (MHz)											
	5.12		6		6.144		7.3728		8		8.192	
	Integer	Fractional	Integer	Fractional	Integer	Fractional	Integer	Fractional	Integer	Fractional	Integer	Fractional
8	195	0	195	5	195	0	194	6	195	5	194	10
11.025	141	8	141	11	141	8	141	1	141	11	141	4
16	97	7	97	9	97	7	97	3	97	9	97	5
22.05	70	11	70	13	70	11	70	7	70	13	70	9
24	65	0	65	2	65	0	64	12	65	2	64	13
32	48	11	48	12	48	11	48	9	48	12	48	10
44.056	35	7	35	7	35	7	35	5	35	7	35	6
44.1	35	6	35	7	35	6	35	4	35	7	35	5
47.25	33	0	33	1	33	0	32	14	33	1	32	14
48	32	7	32	8	32	7	32	6	32	8	32	7
50	31	3	31	4	31	3	31	2	31	4	31	2
88.2	17	10	17	10	17	10	17	9	17	10	17	10
96	16	4	16	4	16	4	16	3	16	4	16	4
128	12	3	12	3	12	3	12	2	12	3	12	3
176.4	Not supported		8	13	Not supported		8	12	8	13	8	12
192	8	2	8	2	8	2	Not supported		8	2	8	2

**Table 17-4. Crystal Frequency (Values from 10 MHz to 14.3181 MHz)**

Fs (kHz)	Crystal Frequency (MHz)									
	10		12		12.288		13.56		14.3181	
	Integer	Fractional	Integer	Fractional	Integer	Fractional	Integer	Fractional	Integer	Fractional
8	195	5	195	5	196	0	194	4	195	11
11.025	141	11	141	11	142	4	140	15	142	1
16	97	9	98	6	98	0	97	2	97	13
22.05	70	13	70	13	71	2	70	7	71	0
24	65	2	65	2	65	5	64	11	65	4
32	48	12	48	12	49	0	48	8	48	14
44.056	35	7	35	7	35	8	35	4	35	8
44.1	35	7	35	7	35	8	35	4	35	7
47.25	33	1	33	1	33	3	32	13	33	2
48	32	8	32	8	32	10	32	6	32	9
50	31	4	31	4	31	6	31	1	31	5
88.2	17	10	17	10	17	11	17	9	17	11
96	16	4	16	4	16	5	16	3	16	5
128	12	3	12	3	12	4	12	2	12	4
176.4	98	13	8	13	8	13	8	12	8	13
192	8	2	8	2	8	2	Not supported		Not supported	

**Table 17-5. Crystal Frequency (Values from 16 MHz to 16.384 MHz)**

Fs (kHz)	Crystal Frequency (MHz)			
	16		16.384	
	Integer	Fractional	Integer	Fractional
8	195	5	192	0
11.025	141	11	139	5
16	97	9	96	0
22.05	70	13	69	10
24	65	2	64	0
32	48	12	48	0
44.056	35	7	34	13
44.1	35	7	35	12
47.25	33	1	32	7
48	32	8	32	0
50	31	4	30	11
88.2	17	10	17	7
96	16	4	16	0
128	12	3	12	0
176.4	8	13	8	10
192	8	2	8	0



### 17.2.1.4 Interrupt Control

There is one I<sup>2</sup>S interrupt to the CPU. The interrupt is asserted to the CPU whenever any of the transmit or receive sources is asserted. The transmit module has two interrupt sources: the FIFO service request and write error. The interrupts may be masked using the TXFSR and TXWE bits in the **I<sup>2</sup>S Interrupt Mask (I2SIM)** register. The status of the interrupt source is indicated by the **I<sup>2</sup>S Raw Interrupt Status (I2SRIS)** register. The status of enabled interrupts is indicated by the **I<sup>2</sup>S Masked Interrupt Status (I2SMIS)** register. The FIFO level interrupt has a second level of masking using the FFM bit in the **I<sup>2</sup>S Transmit Interrupt Status and Mask (I2STXISM)** register.

The FIFO service request interrupt is asserted when the FIFO level (indicated by the LEVEL field in the **I<sup>2</sup>S Transmit FIFO Level (I2STXLEV)** register) is below the FIFO limit (programmed using the **I<sup>2</sup>S Transmit FIFO Limit (I2STXLIMIT)** register) and both the TXFSR and FFM bits are set. If software attempts to write to a full FIFO, a Transmit FIFO Write error occurs (indicated by the TXWE bit in the **I<sup>2</sup>S Raw Interrupt Status (I2SRIS)** register). The TXWE bit in the **I2SRIS/I2SMIS** registers is cleared by setting the TXWE bit in the **I<sup>2</sup>S Interrupt Clear (I2SIC)** register.

### 17.2.1.5 DMA Support

The  $\mu$ DMA can be used to more efficiently stream data to and from the I<sup>2</sup>S bus. The FIFO Interrupt Mask bit (FFM) in the **I2STXISM** register must be set for the request signaling to propagate to the  $\mu$ DMA module. See “Micro Direct Memory Access ( $\mu$ DMA)” on page 219 for channel configuration.

The I<sup>2</sup>S module uses the  $\mu$ DMA burst request signal, not the single request. Thus each time a  $\mu$ DMA request is made, the  $\mu$ DMA controller transfers the number of items specified as the burst size for the  $\mu$ DMA channel. Therefore, the  $\mu$ DMA channel burst size and the I<sup>2</sup>S FIFO service request limit must be set to the same value (using the LIMIT field in the **I2STXLIMIT** register).

## 17.2.2 Receive

### 17.2.2.1 Serial Decoder

The serial decoder accepts incoming audio stream data and places the sample data in the receive FIFO. By configuring the serial decoder, common audio formats I<sup>2</sup>S, Left-Justified, and Right-Justified are supported. The MSB is transmitted first. The sample size and system data size are configurable with the SSZ and SDSZ bits in the **I<sup>2</sup>S Receive Module Configuration (I2SRXCFG)** register. The sample size is the number of bits of data being received, and the system data size is the number of I2S0TXSCK transitions between the word select transitions. The system data size must be large enough to accommodate the maximum sample size. Any bits received after the LSB are 0s. If the FIFO is full, the incoming sample (in Mono) or sample-pairs (Stereo) are dropped until the FIFO has space. The serial decoder is enabled using the RXEN bit in the **I2SCFG** register.

### 17.2.2.2 FIFO Operation

The receive FIFO stores eight Mono samples or eight Stereo sample-pairs of data and is accessed through the **I<sup>2</sup>S Receive FIFO Data (I2SRXFIFO)** register. Table 17-6 on page 646 defines the interface for each Read mode. All data is stored MSB-aligned. The Stereo data is read left sample then right.

In Mono mode, the FIFO interface can be configured to read the right or left channel by setting the FIFO Mono Mode bit (FMM) in the **I<sup>2</sup>S Receive FIFO Configuration (I2SRXFIFOCFG)** register. This enables reads from a single channel, where the channel selected can be either the right or left as determined by the LRS bit in the **I2SRXFIFOCFG** register.

Table 17-6. I<sup>2</sup>S Receive FIFO Interface

RM bit in I2RXCFG	CSS bit in I2SRXFIFOCFG	Read Mode	Sample Width	Samples per FIFO Write	Data Alignment
0	don't care	Stereo	8-32 bits	1	MSB
1	0	Compact Stereo - 16 bit	8-16 bits	2	MSB Right [31:15], Left [15:0]
1	1	Compact Stereo - 8 bit	8 bits	2	Right [15:8] Left[7:0]
0	don't care	Mono (FMM bit in the I2SRXFIFOCFG register must be set.)	8-32 bits	1	MSB

The number of samples in the receive FIFO can be read using the **I<sup>2</sup>S Receive FIFO Level (I2SRXLEV)** register. The value ranges from 0 to 16. Stereo and compact stereo sample pairs are counted as two. The mono samples also increment the count by two, therefore four Mono samples will have a count of eight.

### 17.2.2.3 Clock Control

The receiver MCLK and SCLK can be independently programmed to be the master or slave. The receiver is programmed to be the master or slave of the SCLK using the **MSL** bit in the **I2SRXCFG** register. When the receiver is the master, the **I2S0RXSCK** frequency is the specified **I2S0RXMCLK** divided by four. The **I2S0RXSCK** may be inverted using the **SCP** bit in the **I2SRXCFG** register.

The receiver can also be the master or slave of the MCLK. When the receiver is the master, the PLL must be active and a fractional clock divider must be programmed. See page 125 for the setup for the master **I2S0RXMCLK** source. An external transmit **I2S0RXMCLK** is selected using the **RXSLV** bit in the **I2SCFG** register.

Refer to “Clock Control” on page 642 for combinations of the **RXINT** and **RXFRAC** bits in the **I<sup>2</sup>S MCLK Configuration (I2SMCLKCFG)** register that provide MCLK frequencies within acceptable error limits. In the table,  $F_s$  is the sampling frequency in kHz and possible crystal frequencies are shown in MHz across the top row of the table. The words "not supported" in the table mean that it is not possible to obtain the specified sampling frequencies with the specified crystal frequency within the error tolerance of 0.3%.

### 17.2.2.4 Interrupt Control

There is one I<sup>2</sup>S interrupt to the CPU. The interrupt is asserted to the CPU whenever any of the transmit or receive sources is asserted. The receive module has two interrupt sources: the FIFO service request and read error. The interrupts may be masked using the **RXFMR** and **RXRE** bits in the **I2SIM** register. The status of the interrupt source is indicated by the **I2SRIS** register. The status of enabled interrupts is indicated by the **I2SMIS** register. The FIFO service request interrupt has a second level of masking using the **FMM** bit in the **I<sup>2</sup>S Receive Interrupt Status and Mask (I2SRISM)** register. The sources may be masked using the **I2SIM** register.

The FIFO service request interrupt is asserted when the FIFO level (indicated by the **LEVEL** field in the **I<sup>2</sup>S Receive FIFO Level (I2SRXLEV)** register) is above the FIFO limit (programmed using the **I<sup>2</sup>S Receive FIFO Limit (I2SRXLIMIT)** register) and both the **RXFMR** and **FMM** bits are set. An error occurs when reading an empty FIFO or if a stereo sample pair is not read left then right. To clear an interrupt, write a 1 to the appropriate bit in the **I2SIC** register. If software attempts to read an empty FIFO or if a stereo sample pair is not read left then right, a Receive FIFO Read error occurs (indicated by the **RXRE** bit in the **I2SRIS** register). The **RXRE** bit in the **I2SRIS/I2SMIS** registers is cleared by setting the **RXRE** bit in the **I2SIC** register.

### 17.2.2.5 DMA Support

The  $\mu$ DMA can be used to more efficiently stream data to and from the I<sup>2</sup>S bus. The FIFO Interrupt Mask bit (**FFM**) in the **I2SRXISM** register must be set for the request signaling to propagate to the  $\mu$ DMA module. See “Micro Direct Memory Access ( $\mu$ DMA)” on page 219 for channel configuration.

The I<sup>2</sup>S module uses the  $\mu$ DMA burst request signal, not the single request. Thus each time a  $\mu$ DMA request is made, the  $\mu$ DMA controller transfers the number of items specified as the burst size for the  $\mu$ DMA channel. Therefore, the  $\mu$ DMA channel burst size and the I<sup>2</sup>S FIFO service request limit must be set to the same value (using the **LIMIT** field in the **I2SRXLIMIT** register).

## 17.3 Initialization and Configuration

The default setup for the I<sup>2</sup>S transmit and receive is to be using external MCLK, external SCLK, Stereo, I<sup>2</sup>S audio format, and 32-bit data samples. The following example shows how to configure a system using the internal MCLK, internal SCLK, Compact Stereo, and Left-Justified audio format with 16-bit data samples.

1. Enable the I<sup>2</sup>S peripheral clock by writing a value of 0x1000.0000 to the **RCGC1** register in the System Control module. See page 163.
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module. See page 175. To find out which GPIO port to enable, refer to Table 24-5 on page 937.
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register. See page 304.
4. Set up the MCLK sources for a 48-kHz sample rate. The input crystal is assumed to be 6 MHz for this example (internal source).
  - Enable the PLL by clearing the **PWRDWN** bit in the **RCC** register in the System Control module. See page 109.
  - Set the MCLK dividers and enable them by writing 0x0208.0208 to the **I2SMCLKCFG** register in the System Control module. See page 125.
  - Enable the MCLK internal sources by writing 0x8208.8208 to the **I2SMCLKCFG** register in the System Control module.

To allow an external MCLK to be used, set bits 4 and 5 of the **I2SCFG** register. Starting up the PLL and enabling the MCLK sources is not required.

5. Set up the Serial Bit Clock SCLK source. By default, the SCLK is externally sourced.
  - Receiver: Masters the **I2S0RXSCK** by ORing 0x0040.0000 into the **I2SRXCFCG** register.
  - Transmitter: Masters the **I2S0TXSCK** by ORing 0x0040.0000 into the **I2STXCFCG** register.
6. Configure the Serial Encoder/Decoder (Left-Justified, Compact Stereo, 16-bit samples, 32-bit system data size).
  - Set the audio format using the Justification (**JST**), Data Delay (**DLY**), SCLK polarity (**SCP**), and Left-Right Polarity (**LRP**) bits written to the **I2STXCFCG** and **I2SRXCFCG** registers. The settings are shown in the table below.

Table 17-7. Audio Formats Configuration

Audio Format	I2STXCFG/I2SRXCFG Register Bit			
	JST	DLY	SCP	LRP
I <sup>2</sup> S	0	1	0	1
Left-Justified	0	0	0	0
Right-Justified	1	0	0	0

- Write 0x0140.3DF0 to both the **I2STXCFG** and **I2SRXCFG** registers to program the following configurations:
  - Set the sample size to 16 bits using the **SSZ** field of the **I2STXCFG** and **I2SRXCFG** registers.
  - Set the system data size to 32 bits using the **SDSZ** field of the **I2STXCFG** and **I2SRXCFG** registers.
  - Set the Write and Read modes using the **WM** and **RM** fields in the **I2STXCFG** and **I2SRXCFG** registers, respectively.
- 7. Set up the FIFO limits for triggering interrupts (also used for  $\mu$ DMA)
  - Set up the transmit FIFO to trigger when it has less than four sample pairs by writing a 0x0000.0008 to the **I2STXLIMIT** register.
  - Set up the receive FIFO to trigger when there are more than four sample pairs by writing a 0x0000.00008 to the **I2SRXLIMIT** register.
- 8. Enable interrupts.
  - Enable the transmit FIFO interrupt by setting the **FFM** bit in the **I2STXISM** register (write 0x0000.0001).
  - Set up the receive FIFO interrupts by setting the **FFM** bit in the **I2SRXISM** register (write 0x0000.0001).
  - Enable the TX FIFO service request, the TX Error, the RX FIFO service request, and the RX Error interrupts to be sent to the CPU by writing a 0x0000.0033 to the **I2SSIM** register.
- 9. Enable the Serial Encoder and Serial Decoders by writing a 0x0000.0003 to the **I2SCFG** register.

## 17.4 Register Map

Table 17-8 on page 648 lists the I<sup>2</sup>S registers. The offset listed is a hexadecimal increment to the register's address, relative to the I<sup>2</sup>S interface base address of 0x4005.4000. Note that the I<sup>2</sup>S module clock must be enabled before the registers can be programmed (see page 163).

Table 17-8. Inter-Integrated Circuit Sound (I<sup>2</sup>S) Interface Register Map

Offset	Name	Type	Reset	Description	See page
0x000	I2STXFIFO	WO	0x0000.0000	I2S Transmit FIFO Data	650

Offset	Name	Type	Reset	Description	See page
0x004	I2STXFIFOCFG	R/W	0x0000.0000	I2S Transmit FIFO Configuration	651
0x008	I2STXCFG	R/W	0x1400.7DF0	I2S Transmit Module Configuration	652
0x00C	I2STXLIMIT	R/W	0x0000.0000	I2S Transmit FIFO Limit	654
0x010	I2STXISM	R/W	0x0000.0000	I2S Transmit Interrupt Status and Mask	655
0x018	I2STXLEV	RO	0x0000.0000	I2S Transmit FIFO Level	656
0x800	I2SRXFIFO	RO	0x0000.0000	I2S Receive FIFO Data	657
0x804	I2SRXFIFOCFG	R/W	0x0000.0000	I2S Receive FIFO Configuration	658
0x808	I2SRXCFG	R/W	0x1400.7DF0	I2S Receive Module Configuration	659
0x80C	I2SRXLIMIT	R/W	0x0000.7FFF	I2S Receive FIFO Limit	661
0x810	I2SRXISM	R/W	0x0000.0000	I2S Receive Interrupt Status and Mask	662
0x818	I2SRXLEV	RO	0x0000.0000	I2S Receive FIFO Level	663
0xC00	I2SCFG	R/W	0x0000.0000	I2S Module Configuration	664
0xC10	I2SIM	R/W	0x0000.0000	I2S Interrupt Mask	665
0xC14	I2SRIS	RO	0x0000.0000	I2S Raw Interrupt Status	666
0xC18	I2SMIS	RO	0x0000.0000	I2S Masked Interrupt Status	668
0xC1C	I2SIC	WO	0x0000.0000	I2S Interrupt Clear	669

## 17.5 Register Descriptions

The remainder of this section lists and describes the I<sup>2</sup>S registers, in numerical order by address offset.

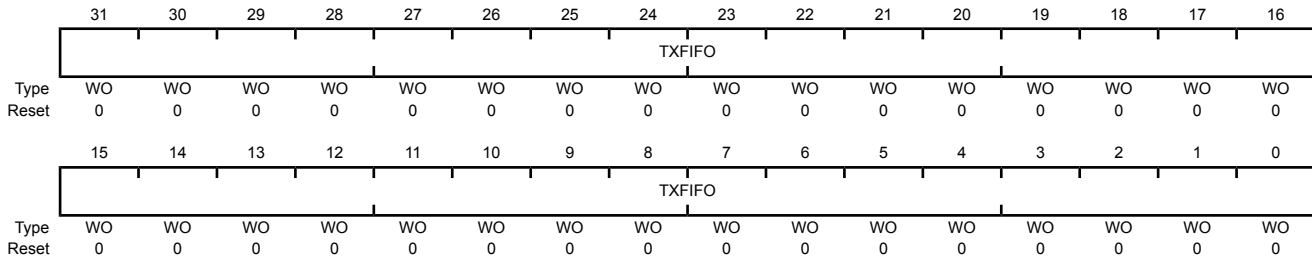
### Register 1: I<sup>2</sup>S Transmit FIFO Data (I2STXFIFO), offset 0x000

This register is the 32-bit serial audio transmit data register. In Stereo mode, the data is written left, right, left, right, and so on. The `LRS` bit in the **I<sup>2</sup>S Transmit FIFO Configuration (I2STXFIFOCFG)** register can be read to verify the next position expected. In Compact 16-bit mode, bits [31:16] contain the right sample, and bits [15:0] contain the left sample. In Compact 8-bit mode, bits [15:8] contain the right sample, and bits [7:0] contain the left sample. In Mono mode, each 32-bit entry is a single sample.

Note that if the FIFO is full and a write is attempted, a transmit FIFO write error is generated.

#### I2S Transmit FIFO Data (I2STXFIFO)

Base 0x4005.4000  
 Offset 0x000  
 Type WO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	TXFIFO	WO	0x0000.0000	TX Data Serial audio sample data to be transmitted.

## Register 2: I<sup>2</sup>S Transmit FIFO Configuration (I2STXFIFOCFG), offset 0x004

This register configures the sample for dual-channel operation. In Stereo mode, the LRS bit toggles between left and right samples as the Transmit FIFO is written. The left sample is written first, followed by the right.

### I<sup>2</sup>S Transmit FIFO Configuration (I2STXFIFOCFG)

Base 0x4005.4000  
Offset 0x004  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														CSS	LRS
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
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	CSS	R/W	0	Compact Stereo Sample Size  When clear, this bit selects Compact 16-bit Stereo Mode, and programs the sample size to 16 bits.  When set, this bit selects Compact 8-bit Stereo Mode, and programs the sample size to 8 bits.
0	LRS	R/W	0	Left-Right Sample Indicator  When clear, this bit indicates that the left sample is the next position.  When set, this bit indicates that the right sample is the next position.  In Mono mode and Compact stereo mode, this bit toggles as if it were in Stereo mode, but it has no meaning and should be ignored.

### Register 3: I<sup>2</sup>S Transmit Module Configuration (I2STXCFG), offset 0x008

This register controls the configuration of the Transmit module.

#### I2S Transmit Module Configuration (I2STXCFG)

Base 0x4005.4000  
 Offset 0x008  
 Type R/W, reset 0x1400.7DF0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved		JST	DLY	SCP	LRP	WM		FMT	MSL	reserved						
Type	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	1	0	1	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	
	SSZ				SDSZ							reserved					
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	RO	RO	RO	RO
Reset	0	1	1	1	1	1	0	1	1	1	1	1	1	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:30	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
29	JST	R/W	0	Justification of Output Data  When clear, this bit configures the data to be Left-Justified.  When set, this bit configures the data to be Right-Justified.
28	DLY	R/W	1	Data Delay  When clear, data is latched on the next latching edge of I2S0TXSCK as defined by the SCP bit. This bit should be clear in Left-Justified or Right-Justified mode.  When set, this bit causes a one-I2S0TXSCK delay from the edge of I2S0TXWS before data is latched. This bit should be set in I <sup>2</sup> S mode.
27	SCP	R/W	0	SCLK Polarity  When clear, this bit causes data to be latched on the falling edge of I2S0TXSCK.  When set, this bit causes data to be latched on the rising edge of I2S0TXSCK.
26	LRP	R/W	1	Left/Right Clock Polarity  When clear, this bit causes I2S0TXWS to be high during the transmission of the left channel data.  When set, this bit causes I2S0TXWS to be high during the transmission of the right channel data.



Bit/Field	Name	Type	Reset	Description												
25:24	WM	R/W	0x0	<p>Write Mode</p> <p>This bit field selects the mode in which the transmit data is stored in the FIFO and transmitted.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Stereo mode</td> </tr> <tr> <td>0x1</td> <td>Compact Stereo mode</td> </tr> <tr> <td></td> <td>Left/Right sample packed. Refer to <b>I2STXFIFOCFG</b> for 8/16-bit sample size selection.</td> </tr> <tr> <td>0x2</td> <td>Mono mode</td> </tr> <tr> <td>0x3</td> <td>reserved</td> </tr> </tbody> </table>	Value	Description	0x0	Stereo mode	0x1	Compact Stereo mode		Left/Right sample packed. Refer to <b>I2STXFIFOCFG</b> for 8/16-bit sample size selection.	0x2	Mono mode	0x3	reserved
Value	Description															
0x0	Stereo mode															
0x1	Compact Stereo mode															
	Left/Right sample packed. Refer to <b>I2STXFIFOCFG</b> for 8/16-bit sample size selection.															
0x2	Mono mode															
0x3	reserved															
23	FMT	R/W	0	<p>FIFO Empty</p> <p>When clear, this bit causes all zeroes to be transmitted if the FIFO is empty.</p> <p>When set, this bit causes the last sample to be transmitted if the FIFO is empty.</p>												
22	MSL	R/W	0	<p>SCLK Master/Slave</p> <p>Source of serial bit clock (<b>I2S0TXSCK</b>) and Word Select (<b>I2S0TXWS</b>).</p> <p>When clear, this bit configures the transmitter as a slave using the externally driven <b>I2S0TXSCK</b> and <b>I2S0TXWS</b> signals.</p> <p>When set, this bit configures the transmitter as a master using the internally generated <b>I2S0TXSCK</b> and <b>I2S0TXWS</b> signals.</p>												
21:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.												
15:10	SSZ	R/W	0x1F	<p>Sample Size</p> <p>This field contains the number of bits minus one in the sample.</p>												
9:4	SDSZ	R/W	0x1F	<p>System Data Size</p> <p>This field contains the number of bits minus one during the high or low phase of the <b>I2S0TXWS</b> signal.</p>												
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 4: I<sup>2</sup>S Transmit FIFO Limit (I2STXLIMIT), offset 0x00C

This register sets the lower FIFO limit at which a FIFO service request is issued.

#### I2S Transmit FIFO Limit (I2STXLIMIT)

Base 0x4005.4000  
 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												LIMIT			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4:0	LIMIT	R/W	0x00	<p>FIFO Limit</p> <p>This field sets the FIFO level at which a FIFO service request is issued, generating an interrupt or a <math>\mu</math>DMA transfer request.</p> <p>The transmit FIFO generates a service request when the number of items in the FIFO is less than the level specified by the <code>LIMIT</code> field. For example, if the <code>LIMIT</code> field is set to 8, then a service request is generated when there are less than 8 samples remaining in the transmit FIFO.</p>

## Register 5: I<sup>2</sup>S Transmit Interrupt Status and Mask (I2STXISM), offset 0x010

This register indicates the transmit interrupt status and interrupt masking control.

### I2S Transmit Interrupt Status and Mask (I2STXISM)

Base 0x4005.4000

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															FFI
Type	RO	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															FFM
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

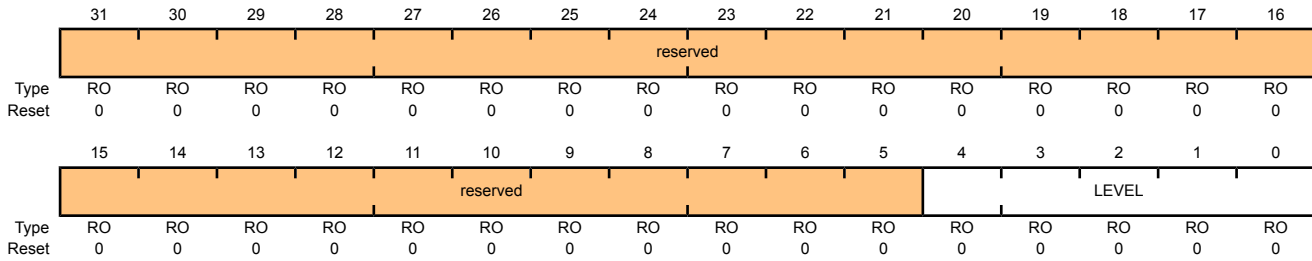
Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	FFI	RO	0	Transmit FIFO Service Request Interrupt  When clear, this bit indicates that the FIFO Level is equal to or above the FIFO Limit.  When set, this bit indicates that the FIFO Level is below the FIFO Limit.
15:1	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.
0	FFM	R/W	0	FIFO Interrupt Mask  When clear, this bit causes the FIFO interrupt to be masked and not sent to the CPU.  When set, this bit allows the FIFO interrupt to be sent to the CPU.

### Register 6: I<sup>2</sup>S Transmit FIFO Level (I2STXLEV), offset 0x018

The number of samples in the transmit FIFO can be read using the **I2STXLEV** register. The value ranges from 0 to 16. Stereo and Compact Stereo sample-pairs are counted as two. Mono samples also increment the count by two. For example, the LEVEL field is set to eight if there are four Mono samples.

#### I2S Transmit FIFO Level (I2STXLEV)

Base 0x4005.4000  
 Offset 0x018  
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4:0	LEVEL	RO	0x00	Number of Audio Samples This field contains the number of samples in the FIFO.

## Register 7: I<sup>2</sup>S Receive FIFO Data (I2SRXFIFO), offset 0x800

This register is the 32-bit serial audio receive data register. In Stereo mode, the data is read left, right, left, right, and so on. The `LRS` bit in the **I<sup>2</sup>S Receive FIFO Configuration (I2SRXFIFOCFG)** register can be read to verify the next position expected. In Compact 16-bit mode, bits [31:16] contain the right sample, and bits [15:0] contain the left sample. In Compact 8-bit mode, bits [15:8] contain the right sample, and bits [7:0] contain the left sample. In Mono mode, each 32-bit entry is a single sample. If the FIFO is empty, a read of this register returns a value of 0x0000.0000 and generates a receive FIFO read error.

### I2S Receive FIFO Data (I2SRXFIFO)

Base 0x4005.4000  
Offset 0x800  
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RXFIFO															
Type	RO	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
	RXFIFO															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

31:0	RXFIFO	RO	0x0000.0000	RX Data
------	--------	----	-------------	---------

Serial audio sample data received.

The read of an empty FIFO will return a value of 0x0.

### Register 8: I<sup>2</sup>S Receive FIFO Configuration (I2SRXFIFOCFG), offset 0x804

This register configures the sample for dual-channel operation. In Stereo mode, the LRS bit toggles between Left and Right as the samples are read from the receive FIFO. In Mono mode, both the left and right samples are stored in the FIFO. The FMM bit can be used to read only the left or right sample as determined by the LRP bit. In Compact Stereo 8- or 16-bit mode, both the left and right samples are read in one access from the FIFO.

#### I2S Receive FIFO Configuration (I2SRXFIFOCFG)

Base 0x4005.4000  
 Offset 0x804  
 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													FMM	CSS	LRS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	FMM	R/W	0	<p>FIFO Mono Mode</p> <p>When clear, this bit configures the receiver in Stereo Mode.</p> <p>When set, this bit configures the receiver in Mono mode. In this case, the LRP bit in the <b>I2SRXCFG</b> register specifies whether data is read while the I2S0RXWS signal is high or low (Right or Left Channel) as follows:</p> <p style="margin-left: 20px;">LRP I2S0RXWS</p> <p style="margin-left: 20px;">0 Low (Right)</p> <p style="margin-left: 20px;">1 High (Left)</p>
1	CSS	R/W	0	<p>Compact Stereo Sample Size</p> <p>When clear, this bit selects Compact 16-bit Stereo Mode, and programs the sample size to 16 bits.</p> <p>When set, this bit selects Compact 8-bit Stereo Mode, and programs the sample size to 8 bits.</p>
0	LRS	R/W	0	<p>Left-Right Sample Indicator</p> <p>When clear, this bit indicates that the left sample is the next position to be read.</p> <p>When set, this bit indicates that the right sample is the next position to be read.</p> <p>This bit is only meaningful in Compact Stereo Mode.</p>

## Register 9: I<sup>2</sup>S Receive Module Configuration (I2SRXCFG), offset 0x808

This register controls the configuration of the receive module.

### I2S Receive Module Configuration (I2SRXCFG)

Base 0x4005.4000

Offset 0x808

Type R/W, reset 0x1400.7DF0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved		JST	DLY	SCP	LRP	reserved	RM	reserved	MSL	reserved						
Type	RO	RO	R/W	R/W	R/W	R/W	RO	R/W	RO	R/W	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	1	0	1	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	
	SSZ						SDSZ						reserved				
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	RO	RO	RO	RO
Reset	0	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0	

Bit/Field	Name	Type	Reset	Description
31:30	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
29	JST	R/W	0	Justification of Input Data When clear, this bit configures the data to be Left-Justified. When set, this bit configures the data to be Right-Justified.
28	DLY	R/W	1	Data Delay When clear, data is latched on the next latching edge of I2S0RXSCK as defined by the SCP bit. This bit should be clear in Left-Justified or Right-Justified mode. When set, this bit causes a one-I2S0RXSCK delay from the edge of I2S0RXWS before data is latched. This bit should be set in I <sup>2</sup> S mode.
27	SCP	R/W	0	SCLK Polarity When clear, this bit causes data to be latched on the falling edge of I2S0RXSCK. When set, this bit causes data to be latched on the rising edge of I2S0RXSCK.
26	LRP	R/W	1	Left/Right Clock Polarity When clear, this bit causes I2S0RXWS to be high during the transmission of the left channel data. When set, this bit causes I2S0RXWS to be high during the transmission of the right channel data.
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.

Bit/Field	Name	Type	Reset	Description
24	RM	R/W	0	<p>Read Mode</p> <p>This bit field selects the mode in which the receive data is received and stored in the FIFO.</p> <p>Value Description</p> <p>0 Stereo/Mono mode</p> <p><b>I2SRXFIFOCFG</b> FMM bit specifies Stereo or Mono FIFO read behavior.</p> <p>1 Compact Stereo mode</p> <p>Left/Right sample packed. Refer to <b>I2SRXFIFOCFG</b> for 8/16-bit sample size selection.</p>
23	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22	MSL	R/W	0	<p>SCLK Master/Slave</p> <p>Source of serial bit clock (<i>I2S0RXSCK</i>) and Word Select (<i>I2S0RXWS</i>).</p> <p>When clear, this bit configures the receiver as a slave using the externally driven <i>I2S0RXSCK</i> and <i>I2S0RXWS</i> signals.</p> <p>When set, this bit configures the receiver as a master using the internally generated <i>I2S0RXSCK</i> and <i>I2S0RXWS</i> signals.</p>
21:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:10	SSZ	R/W	0x1F	<p>Sample Size</p> <p>This field contains the number of bits minus one in the sample.</p>
9:4	SDSZ	R/W	0x1F	<p>System Data Size</p> <p>This field contains the number of bits minus one during the high or low phase of the <i>I2S0RXWS</i> signal.</p>
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 10: I<sup>2</sup>S Receive FIFO Limit (I2SRXLIMIT), offset 0x80C

This register sets the upper FIFO limit at which a FIFO service request is issued.

### I2S Receive FIFO Limit (I2SRXLIMIT)

Base 0x4005.4000

Offset 0x80C

Type R/W, reset 0x0000.7FFF

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:5	reserved	RO	0x7FF	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4:0	LIMIT	R/W	0x1F	FIFO Limit  This field sets the FIFO level at which a FIFO service request is issued, generating an interrupt or a $\mu$ DMA transfer request.  The receive FIFO generates a service request when the number of items in the FIFO is greater than the level specified by the <code>LIMIT</code> field. For example, if the <code>LIMIT</code> field is set to 4, then a service request is generated when there are less than 4 samples remaining in the transmit FIFO.

### Register 11: I<sup>2</sup>S Receive Interrupt Status and Mask (I2SRXISM), offset 0x810

This register indicates the receive interrupt status and interrupt masking control.

#### I2S Receive Interrupt Status and Mask (I2SRXISM)

Base 0x4005.4000  
 Offset 0x810  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															FFI
Type	RO	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															FFM
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
31:17	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	FFI	RO	0	Receive FIFO Service Request Interrupt  When clear, this bit indicates that the FIFO Level is equal to or below the FIFO Limit.  When set, this bit indicates that the FIFO Level is above the FIFO Limit.
15:1	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.
0	FFM	R/W	0	FIFO Interrupt Mask  When clear, this bit causes the FIFO interrupt to be masked and not sent to the CPU.  When set, this bit allows the FIFO interrupt to be sent to the CPU.

## Register 12: I<sup>2</sup>S Receive FIFO Level (I2SRXLEV), offset 0x818

The number of samples in the receive FIFO can be read using the **I2SRXLEV** register. The value ranges from 0 to 16. Stereo and Compact Stereo sample pairs are counted as two. Mono samples also increment the count by two. For example, the LEVEL field is set to eight if there are four Mono samples.

### I2S Receive FIFO Level (I2SRXLEV)

Base 0x4005.4000  
Offset 0x818  
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												LEVEL			
Type	RO	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
31:5	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4:0	LEVEL	RO	0x00	Number of Audio Samples This field contains the number of samples in the FIFO.

### Register 13: I<sup>2</sup>S Module Configuration (I2SCFG), offset 0xC00

This register enables the transmit and receive serial engines and sets the source of the I2S0TXMCLK and I2S0RXMCLK signals.

#### I2S Module Configuration (I2SCFG)

Base 0x4005.4000  
 Offset 0xC00  
 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											RXSLV	TXSLV	reserved		RXEN	TXEN
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	RO	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	RXSLV	R/W	0	Use External I2S0RXMCLK  When set, this bit configures the receiver to use the externally driven I2S0RXMCLK signal.  When clear, this bit configures the receiver to use the internally generated MCLK as the I2S0RXMCLK signal. See "Clock Control" on page 642 for information on how to program the I2S0RXMCLK.
4	TXSLV	R/W	0	Use External I2S0TXMCLK  When set, this bit configures the transmitter to use the externally driven I2S0TXMCLK signal.  When clear, this bit configures the transmitter to use the internally generated MCLK as the I2S0TXMCLK signal. See "Clock Control" on page 642 for information on how to program the I2S0TXMCLK.
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	RXEN	R/W	0	Serial Receive Engine Enable  When clear, this bit disables the serial receive engine.  When set, this bit enables the serial receive engine.
0	TXEN	R/W	0	Serial Transmit Engine Enable  When clear, this bit disables the serial transmit engine.  When set, this bit enables the serial transmit engine.

## Register 14: I<sup>2</sup>S Interrupt Mask (I2SIM), offset 0xC10

This register masks the interrupts to the CPU.

### I2S Interrupt Mask (I2SIM)

Base 0x4005.4000  
Offset 0xC10  
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											RXRE	RXFSR	reserved		TXWE	TXFSR
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	RO	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	RXRE	R/W	0	Receive FIFO Read Error  When clear, this bit causes the receive FIFO read error interrupt to be masked and not sent to the CPU.  When set, this bit allows the receive FIFO read error interrupt to be sent to the CPU.
4	RXFSR	R/W	0	Receive FIFO Service Request  When clear, this bit causes the receive FIFO service request interrupt to be masked and not sent to the CPU.  When set, this bit allows the receive FIFO service request interrupt to be sent to the CPU.
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	TXWE	R/W	0	Transmit FIFO Write Error  When clear, this bit causes the transmit FIFO write error interrupt to be masked and not sent to the CPU.  When set, this bit allows the transmit FIFO write error interrupt to be sent to the CPU.
0	TXFSR	R/W	0	Transmit FIFO Service Request  When clear, this bit causes the transmit FIFO service request interrupt to be masked and not sent to the CPU.  When set, this bit allows the transmit FIFO service request interrupt to be sent to the CPU.

## Register 15: I<sup>2</sup>S Raw Interrupt Status (I2SRIS), offset 0xC14

This register reads the unmasked interrupt status.

### I2S Raw Interrupt Status (I2SRIS)

Base 0x4005.4000  
 Offset 0xC14  
 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											RXRE	RXF5R	reserved	TXWE	TXFSR
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	RXRE	RO	0	Receive FIFO Read Error  When set, this bit indicates that a receive FIFO read error interrupt has occurred.  When clear, this bit indicates that no interrupt has occurred.  This bit is cleared by setting the <b>RXRE</b> bit in the <b>I2SIC</b> register.
4	RXF5R	RO	0	Receive FIFO Service Request  When set, this bit indicates that a receive FIFO service request interrupt has occurred.  When clear, this bit indicates that no interrupt has occurred.  This bit is cleared when the level in the receive FIFO has risen to a value greater than the value programmed in the <b>LIMIT</b> field in the <b>I2SRXLIMIT</b> register.
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	TXWE	RO	0	Transmit FIFO Write Error  When set, this bit indicates that a transmit FIFO write error interrupt has occurred.  When clear, this bit indicates that no interrupt has occurred.  This bit is cleared by setting the <b>TXWE</b> bit in the <b>I2SIC</b> register.

Bit/Field	Name	Type	Reset	Description
0	TXFSR	RO	0	<p>Transmit FIFO Service Request</p> <p>When set, this bit indicates that a transmit FIFO service request interrupt has occurred.</p> <p>When clear, this bit indicates that no interrupt has occurred.</p> <p>This bit is cleared when the level in the transmit FIFO has fallen to a value less than the value programmed in the <code>LIMIT</code> field in the <code>I2STXLIMIT</code> register.</p>

## Register 16: I<sup>2</sup>S Masked Interrupt Status (I2SMIS), offset 0xC18

This register reads the masked interrupt status. The mask is defined in the I2SIM register.

### I2S Masked Interrupt Status (I2SMIS)

Base 0x4005.4000  
 Offset 0xC18  
 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											RXRE	RXF5R	reserved	TXWE	TXFSR
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	RXRE	RO	0	Receive FIFO Read Error  When set, this bit indicates that a receive FIFO read error interrupt has occurred and has been sent to the CPU.  When clear, this bit indicates that no interrupt has occurred or that the interrupt is masked.
4	RXF5R	RO	0	Receive FIFO Service Request  When set, this bit indicates that a receive FIFO service request interrupt has occurred and has been sent to the CPU.  When clear, this bit indicates that no interrupt has occurred or that the interrupt is masked.
3:2	reserved	RO	0s0	Software should not rely on the value of 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	TXWE	RO	0	Transmit FIFO Write Error  When set, this bit indicates that a transmit FIFO write error interrupt has occurred and has been sent to the CPU.  When clear, this bit indicates that no interrupt has occurred or that the interrupt is masked.
0	TXFSR	RO	0	Transmit FIFO Service Request  When set, this bit indicates that a transmit FIFO service request interrupt has occurred and has been sent to the CPU.  When clear, this bit indicates that no interrupt has occurred or that the interrupt is masked.



## Register 17: I<sup>2</sup>S Interrupt Clear (I2SIC), offset 0xC1C

Setting a bit in this register clears the corresponding interrupt.

### I2S Interrupt Clear (I2SIC)

Base 0x4005.4000  
Offset 0xC1C  
Type WO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:6	reserved	WO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	RXRE	WO	0	Receive FIFO Read Error  When set, this bit clears the receive FIFO read error interrupt bit (RXRE) in the I2SRIS register.
4:2	reserved	WO	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	TXWE	WO	0	Transmit FIFO Write Error  When set, this bit clears the transmit FIFO write error interrupt bit (TXWE) in the I2SRIS register.
0	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.

## 18 Controller Area Network (CAN) Module

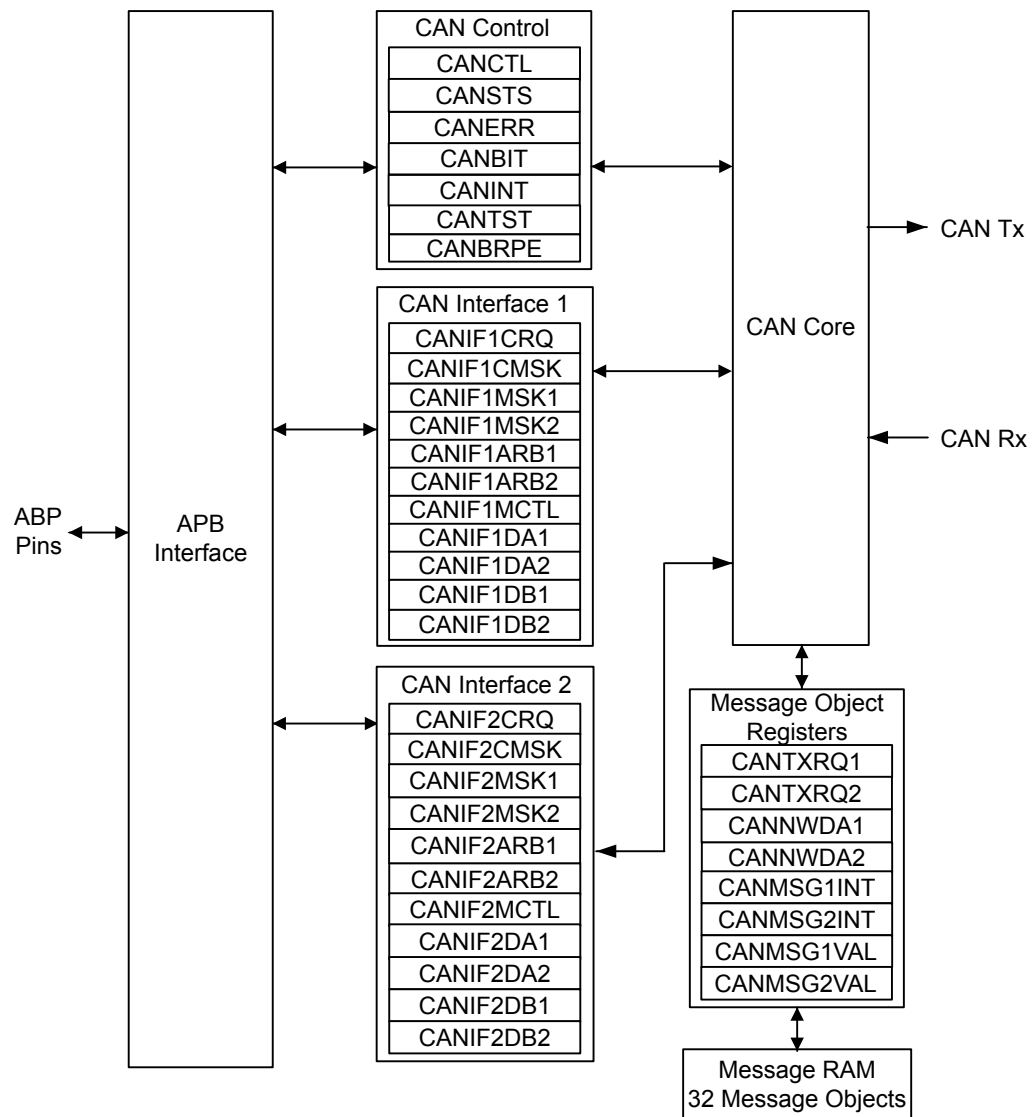
Controller Area Network (CAN) is a multicast, shared serial bus standard for connecting electronic control units (ECUs). CAN was specifically designed to be robust in electromagnetically-noisy environments and can utilize a differential balanced line like RS-485 or a more robust twisted-pair wire. Originally 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 less than 40 meters. Decreased bit rates allow longer network distances (for example, 125 Kbps at 500 meters).

Each Stellaris<sup>®</sup> CAN controller supports the following features:

- CAN protocol version 2.0 part A/B
- Bit rates up to 1 Mbps
- 32 message objects with individual identifier masks
- Maskable interrupt
- Disable Automatic Retransmission mode for Time-Triggered CAN (TTCAN) applications
- Programmable Loopback mode for self-test operation
- Programmable FIFO mode enables storage of multiple message objects
- Gluelessly attaches to an external CAN transceiver through the `CANnTX` and `CANnRX` signals

## 18.1 Block Diagram

Figure 18-1. CAN Controller Block Diagram



## 18.2 Functional Description

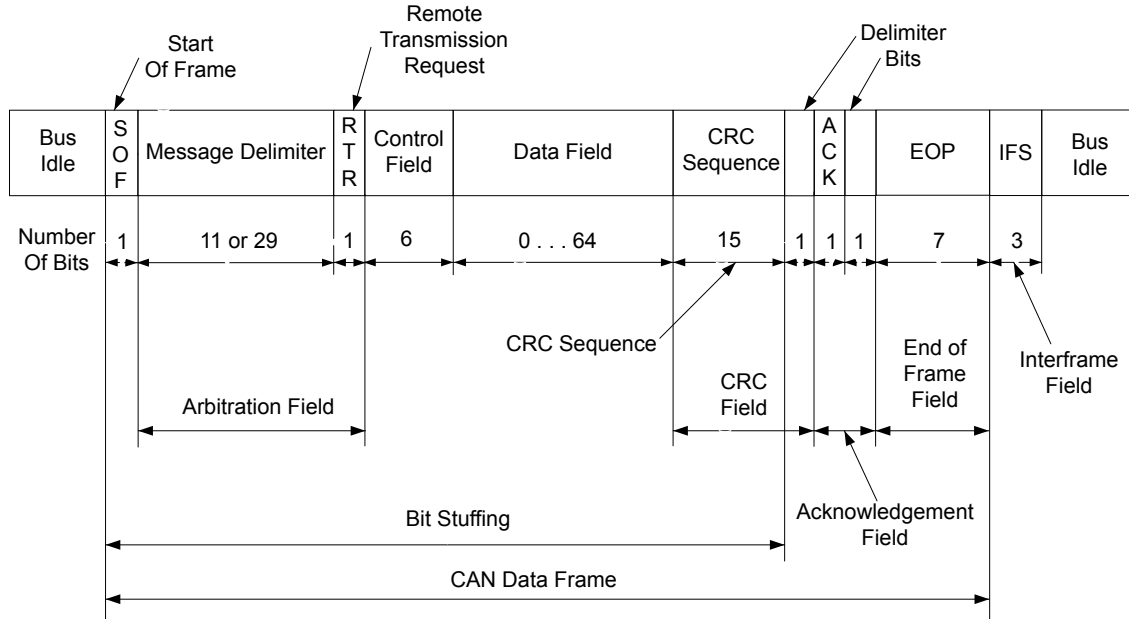
The Stellaris<sup>®</sup> CAN controller conforms to the CAN protocol version 2.0 (parts A and B). Message transfers that include data, remote, error, and overload frames with an 11-bit identifier (standard) or a 29-bit identifier (extended) are supported. Transfer rates can be programmed up to 1 Mbps.

The CAN module consists of three major parts:

- CAN protocol controller and message handler
- Message memory
- CAN register interface

A data frame contains data for transmission, whereas a remote frame contains no data and is used to request the transmission of a specific message object. The CAN data/remote frame is constructed as shown in Figure 18-2 on page 672.

**Figure 18-2. CAN Data/Remote Frame**



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 either of the CAN message object register interfaces.

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 via two CAN interface register sets for communicating with the message objects. As 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 must be processed. In general, one interface is used for transmit data and one for receive data.

### 18.2.1 Initialization

To use the CAN controller, the peripheral clock must be enabled using the **RCGC0** register (see page 154). In addition, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register. See page 175. To find out which GPIO port to enable, refer to Table 24-5 on page 937.

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 **CANnTX** signal is held 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 while 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, label it as not valid by clearing the `MSGVAL` bit in the **CAN IFn Arbitration 2 (CANIFnARB2)** register. Otherwise, the whole message object must be initialized, as the fields of the message object may not have valid information, causing unexpected results. Both the `INIT` and `CCE` bits in the **CANCTL** register must be set in order to access the **CANBIT** register and the **CAN Baud Rate Prescaler Extension (CANBRPE)** register to configure the bit timing. 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 (indicating a bus idle condition) before it takes part in bus activities and starts message transfers. Message object initialization does not require the CAN to be in the initialization state and can be done on the fly. However, message objects should all be configured to particular identifiers or set to not valid before message transfer starts. To change the configuration of a message object during normal operation, clear the `MSGVAL` bit in the **CANIFnARB2** register to indicate that the message object is not valid during the change. When the configuration is completed, set the `MSGVAL` bit again to indicate that the message object is once again valid.

## 18.2.2 Operation

There are two sets of CAN Interface Registers (**CANIF1x** and **CANIF2x**), 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 two sets are independent and identical and can be used to queue transactions. Generally, one interface is used to transmit data and one is used to receive data.

Once the CAN module is initialized and the `INIT` bit in the **CANCTL** register is cleared, the CAN module synchronizes itself to the CAN bus and starts the message transfer. As each message is received, it goes through the message handler's filtering process, and if it passes through the filter, is stored in the message object specified by the `MNUM` bit in the **CAN IFn Command Request (CANIFnCRQ)** register. 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 **CAN IFn Mask 1** and **CAN IFn Mask 2 (CANIFnMSKn)** registers) is used, the arbitration bits that are masked to "don't care" may be overwritten in the message object.

The CPU may read or write each message at any time via the CAN Interface Registers. The message handler guarantees data consistency in case of concurrent accesses.

The transmission of message objects is 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. At the start of transmission, the appropriate `TXRQST` bit in the **CAN Transmission Request n (CANTXRQn)** register and the `NEWDAT` bit in the **CAN New Data n (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 (`MNUM`) for the message object, with 1 being the highest priority and 32 being the lowest priority. 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.

Transmission can be automatically started by the reception of a matching remote frame. To enable this mode, set the `RMTEN` bit in the **CAN IFn Message Control (CANIFnMCTL)** register. A matching received remote frame causes the `TXRQST` bit to be set and the message object automatically transfers its data or generates an interrupt indicating a remote frame was requested. This can be strictly a single message identifier, or it can be a range of values specified in the message object. The CAN mask registers, **CANIFnMSKn**, configure which groups of frames are identified as remote frame requests. The `UMASK` bit in the **CANIFnMCTL** register enables the `MSK` bits in the **CANIFnMSKn** register to filter which frames are identified as a remote frame request. The `MXTD` bit in the **CANIFnMSK2** register should be set if a remote frame request is expected to be triggered by 29-bit extended identifiers.

### 18.2.3 Transmitting Message Objects

If the internal transmit shift register of the CAN module is ready for loading, and if there is no data transfer occurring between the CAN Interface Registers and message RAM, the valid message object with the highest priority 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 in the **CANNWDAn** register is cleared. 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 **CANTXRQn** register is cleared. If the CAN controller is set up to interrupt upon a successful transmission of a message object, (the `TXIE` bit in the **CAN IFn Message Control (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.

### 18.2.4 Configuring a Transmit Message Object

The following steps illustrate how to configure a transmit message object.

1. In the **CAN IFn Command Mask (CANIFnCMASK)** register:
  - Set the `WRNRD` bit to specify a write to the **CANIFnCMASK** register; specify whether to transfer the `IDMASK`, `DIR`, and `MXTD` of the message object into the **CAN IFn** registers using the `MASK` bit
  - Specify whether to transfer the `ID`, `DIR`, `XTD`, and `MSGVAL` of the message object into the interface registers using the `ARB` bit
  - Specify whether to transfer the control bits into the interface registers using the `CONTROL` bit
  - Specify whether to clear the `INTPND` bit in the **CANIFnMCTL** register using the `CLRINTPND` bit
  - Specify whether to clear the `NEWDAT` bit in the **CANNWDAn** register using the `NEWDAT` bit
  - Specify which bits to transfer using the `DATAA` and `DATAB` bits
2. In the **CANIFnMSK1** register, use the `MSK[15:0]` bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. Note that `MSK[15:0]` in this register are used for bits [15:0] of the 29-bit message identifier and are not used for an 11-bit

identifier. A value of 0x00 enables all messages to pass through the acceptance filtering. Also note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the `UMASK` bit in the **CANIFnMCTL** register.

3. In the **CANIFnMSK2** register, use the `MSK[12:0]` bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. Note that `MSK[12:0]` are used for bits [28:16] of the 29-bit message identifier; whereas `MSK[12:2]` are used for bits [10:0] of the 11-bit message identifier. Use the `MXTD` and `MDIR` bits to specify whether to use `XTD` and `DIR` for acceptance filtering. A value of 0x00 enables all messages to pass through the acceptance filtering. Also note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the `UMASK` bit in the **CANIFnMCTL** register.
4. For a 29-bit identifier, configure `ID[15:0]` in the **CANIFnARB1** register to are used for bits [15:0] of the message identifier and `ID[12:0]` in the **CANIFnARB2** register to are used for bits [28:16] of the message identifier. Set the `XTD` bit to indicate an extended identifier; set the `DIR` bit to indicate transmit; and set the `MSGVAL` bit to indicate that the message object is valid.
5. For an 11-bit identifier, disregard the **CANIFnARB1** register and configure `ID[12:2]` in the **CANIFnARB2** register to are used for bits [10:0] of the message identifier. Clear the `XTD` bit to indicate a standard identifier; set the `DIR` bit to indicate transmit; and set the `MSGVAL` bit to indicate that the message object is valid.
6. In the **CANIFnMCTL** register:
  - Optionally set the `UMASK` bit to enable the mask (`MSK`, `MXTD`, and `MDIR` specified in the **CANIFnMSK1** and **CANIFnMSK2** registers) for acceptance filtering
  - Optionally set the `TXIE` bit to enable the `INTPND` bit to be set after a successful transmission
  - Optionally set the `RMTEN` bit to enable the `TXRQST` bit to be set upon the reception of a matching remote frame allowing automatic transmission
  - Set the `EOB` bit for a single message object;
  - Set the `DLC[3:0]` field to specify the size of the data frame. Take care during this configuration not to set the `NEWDAT`, `MSGLST`, `INTPND` or `TXRQST` bits.
7. Load the data to be transmitted into the CAN IFn Data (**CANIFnDA1**, **CANIFnDA2**, **CANIFnDB1**, **CANIFnDB2**) or (**CANIFnDATAA** and **CANIFnDATAB**) registers. Byte 0 of the CAN data frame is stored in `DATA[7:0]` in the **CANIFnDA1** register.
8. Program the number of the message object to be transmitted in the `MNUM` field in the **CAN IFn Command Request (CANIFnCRQ)** register.
9. When everything is properly configured, set the `TXRQST` bit in the **CANIFnMCTL** register. Once this bit is set, the message object is available to be transmitted, depending on priority and bus availability. Note that setting the `RMTEN` bit in the **CANIFnMCTL** register can also start message transmission if a matching remote frame has been received.

### 18.2.5 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` bit in the **CANIFnARB2** register nor the `TXRQST` bits in the **CANIFnMCTL** register have to be cleared before the update.

Even if only some of the data bytes are to be updated, all four bytes of the corresponding **CANIFnDAn/CANIFnDBn** register have to be valid before the content of that register is transferred to the message object. Either the CPU must write all four bytes into the **CANIFnDAn/CANIFnDBn** register or the message object is transferred to the **CANIFnDAn/CANIFnDBn** register before the CPU writes the new data bytes.

In order to only update the data in a message object, the **WRNRD**, **DATAA** and **DATAB** bits in the **CANIFnMSKn** register are set, followed by writing the updated data into **CANIFnDA1**, **CANIFnDA2**, **CANIFnDB1**, and **CANIFnDB2** registers, and then the number of the message object is written to the **MNUM** field in the **CAN IFn Command Request (CANIFnCRQ)** register. To begin transmission of the new data as soon as possible, set the **TXRQST** bit in the **CANIFnMSKn** register.

To prevent the clearing of the **TXRQST** bit in the **CANIFnMCTL** register at the end of a transmission that may already be in progress while the data is updated, the **NEWDAT** and **TXRQST** bits have to be set at the same time in the **CANIFnMCTL** register. When these bits are set at the same time, **NEWDAT** is cleared as soon as the new transmission has started.

### 18.2.6 Accepting Received Message Objects

When the arbitration and control field (the **ID** and **XTD** bits in the **CANIFnARB2** and the **RMTEN** and **DLC[3:0]** bits of the **CANIFnMCTL** register) of an incoming message is completely shifted into the CAN controller, the message handling capability of the controller starts scanning the message RAM for a matching valid message object. To scan the message RAM for a matching message object, the controller uses the acceptance filtering programmed through the mask bits in the **CANIFnMSKn** register and enabled using the **UMASK** bit in the **CANIFnMCTL** register. Each valid message object, starting with object 1, is compared with the incoming message to locate a matching message object in the message RAM. If a match occurs, the scanning is stopped and the message handler proceeds depending on whether it is a data frame or remote frame that was received.

### 18.2.7 Receiving a Data Frame

The message handler stores the message from the CAN controller receive shift register into the matching message object in the message RAM. The data bytes, all arbitration bits, and the **DLC** bits are all stored into the corresponding message object. In this manner, the data bytes are connected with the identifier even if arbitration masks are used. The **NEWDAT** bit of the **CANIFnMCTL** register is set to indicate that new data has been received. The CPU should clear this bit 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 **NEWDAT** bit is already set, the **MSGLST** bit in the **CANIFnMCTL** register is set to indicate that the previous data was lost. If the system requires an interrupt upon successful reception of a frame, the **RXIE** bit of the **CANIFnMCTL** register should be set. In this case, the **INTPND** bit of the same register is set, causing the **CANINT** register to point to the message object that just received a message. The **TXRQST** bit of this message object should be cleared to prevent the transmission of a remote frame.

### 18.2.8 Receiving a Remote Frame

A remote frame contains no data, but instead specifies which object should be transmitted. When a remote frame is received, three different configurations of the matching message object have to be considered:



Configuration in CANIFnMCTL	Description
<ul style="list-style-type: none"> <li>■ DIR = 1 (direction = transmit); programmed in the <b>CANIFnARB2</b> register</li> <li>■ RMTEN = 1 (set the TXRQST bit of the <b>CANIFnMCTL</b> register at reception of the frame to enable transmission)</li> <li>■ UMASK = 1 or 0</li> </ul>	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, and the controller automatically transfers the data in the message object as soon as possible.
<ul style="list-style-type: none"> <li>■ DIR = 1 (direction = transmit); programmed in the <b>CANIFnARB2</b> register</li> <li>■ RMTEN = 0 (do not change the TXRQST bit of the <b>CANIFnMCTL</b> register at reception of the frame)</li> <li>■ UMASK = 0 (ignore mask in the <b>CANIFnMSK<sub>n</sub></b> register)</li> </ul>	At the reception of a matching remote frame, the TXRQST bit of this message object remains unchanged, and the remote frame is ignored. This remote frame is disabled, the data is not transferred and there is no indication that the remote frame ever happened.
<ul style="list-style-type: none"> <li>■ DIR = 1 (direction = transmit); programmed in the <b>CANIFnARB2</b> register</li> <li>■ RMTEN = 0 (do not change the TXRQST bit of the <b>CANIFnMCTL</b> register at reception of the frame)</li> <li>■ UMASK = 1 (use mask (MSK, MXTD, and MDIR in the <b>CANIFnMSK<sub>n</sub></b> register) for acceptance filtering)</li> </ul>	At the reception of a matching remote frame, the TXRQST bit of this message object is cleared. 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.

## 18.2.9 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.

## 18.2.10 Configuring a Receive Message Object

The following steps illustrate how to configure a receive message object.

1. Program the **CAN IF<sub>n</sub> Command Mask (CANIFnCMASK)** register as described in the “Configuring a Transmit Message Object” on page 674 section, except that the WRNRD bit is set to specify a write to the message RAM.
2. Program the **CANIFnMSK1** and **CANIFnMSK2** registers as described in the “Configuring a Transmit Message Object” on page 674 section to configure which bits are used for acceptance filtering. Note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the **CANIFnMCTL** register.
3. In the **CANIFnMSK2** register, use the MSK[12:0] bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. Note that MSK[12:0] are used for bits [28:16] of the 29-bit message identifier; whereas MSK[12:2] are used for bits [10:0] of the 11-bit message identifier. Use the MXTD and MDIR bits to specify whether to use XTD and DIR for acceptance filtering. A value of 0x00 enables all messages to pass through the

acceptance filtering. Also note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the `UMASK` bit in the **CANIFnMCTL** register.

4. Program the **CANIFnARB1** and **CANIFnARB2** registers as described in the “Configuring a Transmit Message Object” on page 674 section to program `XTD` and `ID` bits for the message identifier to be received; set the `MSGVAL` bit to indicate a valid message; and clear the `DIR` bit to specify receive.
5. In the **CANIFnMCTL** register:
  - Optionally set the `UMASK` bit to enable the mask (`MSK`, `MXTD`, and `MDIR` specified in the **CANIFnMSK1** and **CANIFnMSK2** registers) for acceptance filtering
  - Optionally set the `RXIE` bit to enable the `INTPND` bit to be set after a successful reception
  - Clear the `RMTEN` bit to leave the `TXRQST` bit unchanged
  - Set the `EOB` bit for a single message object
  - Set the `DLC[3:0]` field to specify the size of the data frame

Take care during this configuration not to set the `NEWDAT`, `MSGLST`, `INTPND` or `TXRQST` bits.

6. Program the number of the message object to be received in the `MNUM` field in the **CAN IFn Command Request (CANIFnCRQ)** register. Reception of the message object begins as soon as a matching frame is available on the CAN bus.

When the message handler stores a data frame in the message object, it stores the received Data Length Code and eight data bytes in the **CANIFnDA1**, **CANIFnDA2**, **CANIFnDB1**, and **CANIFnDB2** register. Byte 0 of the CAN data frame is stored in `DATA[7:0]` in the **CANIFnDA1** register. If the Data Length Code is less than 8, the remaining bytes of the message object are overwritten by unspecified values.

The CAN mask registers can be used to allow groups of data frames to be received by a message object. The CAN mask registers, **CANIFnMSKn**, configure which groups of frames are received by a message object. The `UMASK` bit in the **CANIFnMCTL** register enables the `MSK` bits in the **CANIFnMSKn** register to filter which frames are received. The `MXTD` bit in the **CANIFnMSK2** register should be set if only 29-bit extended identifiers are expected by this message object.

### 18.2.11 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 **CANIFnCMSK** register and then writes the number of the message object to the **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 generated by this message object.

If the message object uses masks for acceptance filtering, the **CANIFnARBn** registers show the full, unmasked ID for the received message.

The `NEWDAT` bit in the **CANIFnMCTL** register shows whether a new message has been received since the last time this message object was read. The `MSGLST` bit in the **CANIFnMCTL** register shows whether more than one message has been received since the last time this message object

was read. `MSGLST` is not automatically cleared, and should be cleared by software after reading its status.

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.

### 18.2.11.1 Configuration of a FIFO Buffer

With the exception of the `EOB` bit in the `CANIFnMCTL` register, the configuration of receive message objects belonging to a FIFO buffer is the same as the configuration of a single receive message object (see "Configuring a Receive Message Object" on page 677). To concatenate two or more message objects into a FIFO buffer, the identifiers and masks (if used) of these message objects have to be programmed to matching values. Due to the implicit priority of the message objects, the message object with the lowest message object number is the first message object in a FIFO buffer. The `EOB` bit of all message objects of a FIFO buffer except the last one must be cleared. The `EOB` bit of the last message object of a FIFO buffer is set, indicating it is the last entry in the buffer.

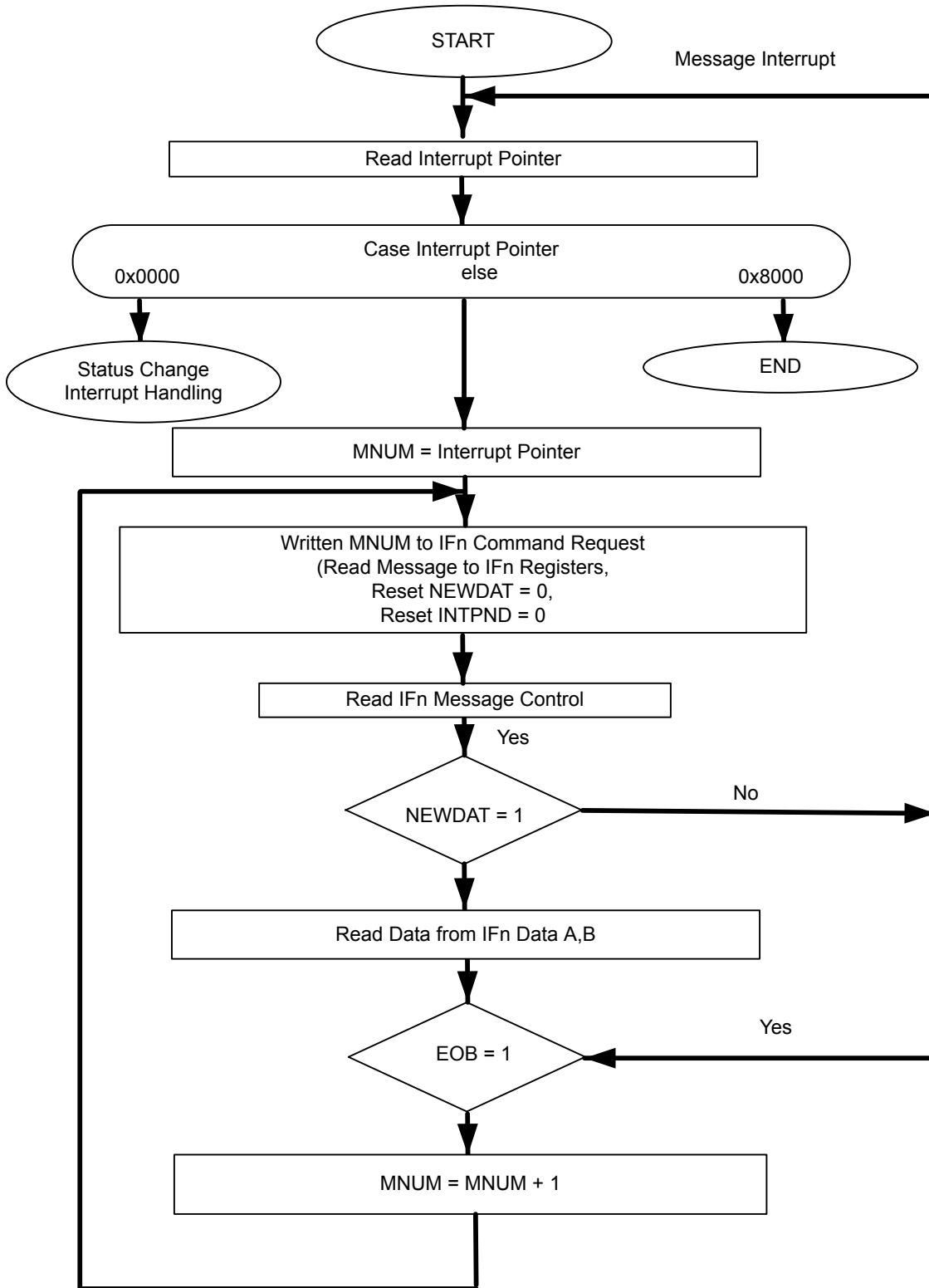
### 18.2.11.2 Reception of Messages with FIFO Buffers

Received messages with identifiers matching to a FIFO buffer are stored starting with the message object with the lowest message number. When a message is stored into a message object of a FIFO buffer, the `NEWDAT` of the `CANIFnMCTL` register bit of this message object is set. By setting `NEWDAT` while `EOB` is clear, the message object is locked and cannot be written to by the message handler until the CPU has cleared the `NEWDAT` bit. Messages are stored into a FIFO buffer until the last message object of this FIFO buffer is reached. If none of the preceding message objects has been released by clearing the `NEWDAT` bit, all further messages for this FIFO buffer will be written into the last message object of the FIFO buffer and therefore overwrite previous messages.

### 18.2.11.3 Reading from a FIFO Buffer

When the CPU transfers the contents of a message object from a FIFO buffer by writing its number to the `CANIFnCRQ` register, the `TXRQST` and `CLRINTPND` bits in the `CANIFnCMSK` register should be set such that the `NEWDAT` and `INTPEND` bits in the `CANIFnMCTL` register are cleared after the read. The values of these bits in the `CANIFnMCTL` register always reflect the status of the message object before the bits are cleared. To assure the correct function of a FIFO buffer, the CPU should read out the message objects starting with the message object with the lowest message number. Figure 18-3 on page 680 shows how a set of message objects which are concatenated to a FIFO Buffer can be handled by the CPU.

Figure 18-3. Message Objects in a FIFO Buffer



## 18.2.12 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. The status interrupt has the highest priority. Among the message interrupts, the message object's interrupt with the lowest message number has the highest priority. A message interrupt is cleared by clearing the message object's **INTPND** bit in the **CANIFnMCTL** register or by reading the **CAN Status (CANSTS)** register. The status Interrupt is cleared by reading the **CANSTS** register.

The interrupt identifier **INTID** in the **CANINT** register indicates the cause of the interrupt. When no interrupt is pending, the register reads as 0x0000. If the value of the **INTID** field 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 the **INTID** field is 0, meaning that all interrupt sources have been cleared (the cause of the interrupt is reset), or until **IE** is cleared, which disables interrupts from the CAN controller.

The **INTID** field of the **CANINT** register points to the pending message interrupt with the highest interrupt priority. The **SIE** bit in the **CANCTL** register controls whether a change of the **RXOK**, **TXOK**, and **LEC** bits in the **CANSTS** register can cause an interrupt. The **EIE** bit in the **CANCTL** register controls whether a change of the **BOFF** and **EWARN** bits in the **CANSTS** register can cause an interrupt. The **IE** 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** register is updated even when the **IE** bit in the **CANCTL** register is clear, but the interrupt will not be indicated to the CPU.

A value of 0x8000 in the **CANINT** register indicates that an interrupt is pending because the CAN module has updated, but not necessarily changed, the **CANSTS** register, indicating that either an error or status interrupt has been generated. A write access to the **CANSTS** register can clear the **RXOK**, **TXOK**, and **LEC** bits in that same register; however, the only way to clear the source of a status interrupt is to read the **CANSTS** register.

There are two ways to determine the source of an interrupt during interrupt handling. The first is to read the **INTID** bit in the **CANINT** 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 clear the message object's **INTPND** bit at the same time by setting the **CLRINTPND** bit in the **CANIFnCMSK** register. Once the **INTPND** bit has been cleared, the **CANINT** register contains the message number for the next message object with a pending interrupt.

## 18.2.13 Test Mode

A Test Mode is provided, which allows various diagnostics to be performed. Test Mode is entered by setting the **TEST** bit **CANCTL** register. Once in Test Mode, the **TX[1:0]**, **LBACK**, **SILENT** and **BASIC** bits in the **CAN Test (CANTST)** register can be used to put the CAN controller into the various diagnostic modes. The **RX** bit in the **CANTST** register allows monitoring of the **CANnRX** signal. All **CANTST** register functions are disabled when the **TEST** bit is cleared.

### 18.2.13.1 Silent Mode

Silent Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits (Acknowledge Bits, Error Frames). The CAN Controller is put in Silent Mode setting the **SILENT** bit in the **CANTST** register. In Silent Mode, the CAN controller is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. If the CAN Controller is required to send a dominant bit (ACK bit, overload flag,

or active error flag), the bit is rerouted internally so that the CAN Controller monitors this dominant bit, although the CAN bus remains in recessive state.

### 18.2.13.2 Loopback Mode

Loopback mode is useful for self-test functions. In Loopback Mode, the CAN Controller internally routes the `CANnTX` signal on to the `CANnRX` signal and treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into the message buffer. The CAN Controller is put in Loopback Mode by setting the `LBACK` bit in the `CANTST` register. To be independent from external stimulation, the CAN Controller ignores acknowledge errors (a recessive bit sampled in the acknowledge slot of a data/remote frame) in Loopback Mode. The actual value of the `CANnRX` signal is disregarded by the CAN Controller. The transmitted messages can be monitored on the `CANnTX` signal.

### 18.2.13.3 Loopback Combined with Silent Mode

Loopback Mode and Silent Mode can be combined to allow the CAN Controller to be tested without affecting a running CAN system connected to the `CANnTX` and `CANnRX` signals. In this mode, the `CANnRX` signal is disconnected from the CAN Controller and the `CANnTX` signal is held recessive. This mode is enabled by setting both the `LBACK` and `SILENT` bits in the `CANTST` register.

### 18.2.13.4 Basic Mode

Basic Mode allows the CAN Controller to be operated without the Message RAM. In Basic Mode, The CANIF1 registers are used as the transmit buffer. The transmission of the contents of the IF1 registers is requested by setting the `BUSY` bit of the `CANIF1CRQ` register. The CANIF1 registers are locked while the `BUSY` bit is set. The `BUSY` bit indicates that a transmission is pending. As soon the CAN bus is idle, the CANIF1 registers are loaded into the shift register of the CAN Controller and transmission is started. When the transmission has completed, the `BUSY` bit is cleared and the locked CANIF1 registers are released. A pending transmission can be aborted at any time by clearing the `BUSY` bit in the `CANIF1CRQ` register while the CANIF1 registers are locked. If the CPU has cleared the `BUSY` bit, a possible retransmission in case of lost arbitration or an error is disabled.

The CANIF2 Registers are used as a receive buffer. After the reception of a message, the contents of the shift register is stored into the CANIF2 registers, without any acceptance filtering. Additionally, the actual contents of the shift register can be monitored during the message transfer. Each time a read message object is initiated by setting the `BUSY` bit of the `CANIF2CRQ` register, the contents of the shift register are stored into the CANIF2 registers.

In Basic Mode, all message-object-related control and status bits and of the control bits of the `CANIFnCMSK` registers are not evaluated. The message number of the `CANIFnCRQ` registers is also not evaluated. In the `CANIF2MCTL` register, the `NEWDAT` and `MSGST` bits retain their function, the `DLC[3:0]` field shows the received DLC, the other control bits are cleared.

Basic Mode is enabled by setting the `BASIC` bit in the `CANTST` register.

### 18.2.13.5 Transmit Control

Software can directly override control of the `CANnTX` signal in four different ways.

- `CANnTX` is controlled by the CAN Controller
- The sample point is driven on the `CANnTX` signal to monitor the bit timing
- `CANnTX` drives a low value

- CANnTX drives a high value

The last two functions, combined with the readable CAN receive pin CANnRX, can be used to check the physical layer of the CAN bus.

The Transmit Control function is enabled by programming the TX[1:0] field in the CANTST register. The three test functions for the CANnTX signal interfere with all CAN protocol functions. TX[1:0] must be cleared when CAN message transfer or Loopback Mode, Silent Mode, or Basic Mode are selected.

### 18.2.14 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.

### 18.2.15 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 18-4 on page 684): 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 18-1 on page 684). 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):

$$t_q = BRP / f_{sys}$$

The CAN module's system clock  $f_{sys}$  is the frequency of its CAN module clock 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.

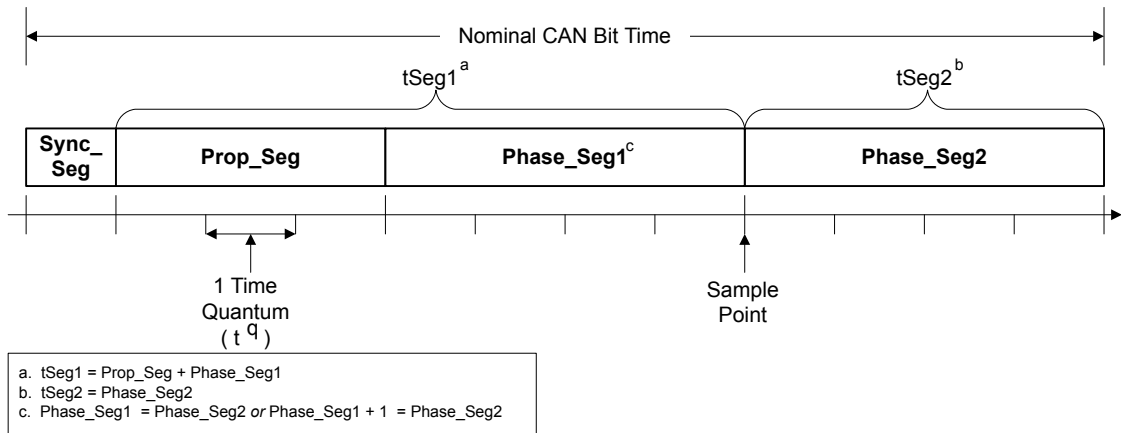
The Propagation Time Segment  $Prop\_Seg$  is intended to compensate for the physical delay times within the CAN network.

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 18-4. CAN Bit Time**



**Table 18-1. CAN Protocol Ranges<sup>a</sup>**

Parameter	Range	Remark
BRP	[1 .. 32]	Defines the length of the time quantum t <sub>q</sub>
Sync_Seg	1 t <sub>q</sub>	Fixed length, synchronization of bus input to system clock
Prop_Seg	[1 .. 8] t <sub>q</sub>	Compensates for the physical delay times
Phase_Seg1	[1 .. 8] t <sub>q</sub>	May be lengthened temporarily by synchronization
Phase_Seg2	[1 .. 8] t <sub>q</sub>	May be shortened temporarily by synchronization
SJW	[1 .. 4] t <sub>q</sub>	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):

$$[TSEG1 + TSEG2 + 3] \times t_q$$

or (functional values):

$$[Sync\_Seg + Prop\_Seg + Phase\_Seg1 + Phase\_Seg2] \times t_q$$

The data in the **CANBIT** register is the configuration input of the CAN protocol controller. The baud rate prescaler (configured by the BRP field) 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.

## 18.2.16 Calculating the Bit Timing Parameters

Usually, the calculation of the bit timing configuration starts with a required 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 required 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  $(\text{bit time} - \text{Prop\_Seg} - 1) t_q$  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 IPT, which is  $t_q$ .

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
- `fosc` = Actual oscillator frequency
- `fnom` = Nominal oscillator frequency

Maximum frequency tolerance must take into account the following formulas:

$$df \leq \frac{(\text{Phase\_Seg1}, \text{Phase\_Seg2}) \min}{2 \times (13 \times \text{tbit} - \text{Phase\_Seg2})}$$

$$df_{max} = 2 \times df \times f_{nom}$$

where:

- Phase\_Seg1 and Phase\_Seg2 are from Table 18-1 on page 684
- tbit = Bit Time
- dfmax = 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.

### 18.2.16.1 Example for Bit Timing at High Baud Rate

In this example, the frequency of CAN clock is 25 MHz, and the bit rate is 1 Mbps.

```

tq 200 ns = (BRP + 1)/CAN Clock
delay of bus driver 50 ns
delay of receiver circuit 30 ns
delay of bus line (40m) 220 ns
tProp 400 ns = 2 × tq
tSJW 200 ns = 1 × tq
tTSeg1 600 ns = tProp + tSJW
tTSeg2 200 ns = (Information Processing Time + 1) × tq
tSync-Seg 200 ns = 1 × tq
bit time 1000 ns = tSync-Seg + tTSeg1 + tTSeg2
    
```

In the above example, the bit field values for the **CANBIT** register are: TSEG2=1, TSEG1=2, SJW =0 and BRP=3. This makes the final value programmed into the **CANBIT** register = 0x3FC0.

### 18.2.16.2 Example for Bit Timing at Low Baud Rate

In this example, the frequency of the CAN clock is 50 MHz, and the bit rate is 100 Kbps.

```

tq 500 ns = (BRP + 1)/CAN clock
delay of bus driver 200 ns
delay of receiver circuit 80 ns
delay of bus line (40m) 220 ns
tProp 4.5 μs = 9 × tq
tSJW 2 μs = 4 × tq
tTSeg1 6.5 μs = tProp + tSJW
tTSeg2 3 μs = (Information Processing Time + 6) × tq
tSync-Seg 500 ns = 1 × tq
bit time 10 μs = tSync-Seg + tTSeg1 + tTSeg2
    
```

In the above example, the bit field values for the **CANBIT** register are: TSEG2=5, TSEG1=12, SJW =3 and BRP=24. This makes the final value programmed into the **CANBIT** register = 0x5CD8.

## 18.3 Register Map

Table 18-2 on page 687 lists the registers. All addresses given are relative to the CAN base address of:

- CAN0: 0x4004.0000
- CAN1: 0x4004.1000

Note that the CAN controller clock must be enabled before the registers can be programmed (see page 154).

**Table 18-2. CAN Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	CANCTL	R/W	0x0000.0001	CAN Control	689
0x004	CANSTS	R/W	0x0000.0000	CAN Status	691
0x008	CANERR	RO	0x0000.0000	CAN Error Counter	694
0x00C	CANBIT	R/W	0x0000.2301	CAN Bit Timing	695
0x010	CANINT	RO	0x0000.0000	CAN Interrupt	697
0x014	CANTST	R/W	0x0000.0000	CAN Test	698
0x018	CANBRPE	R/W	0x0000.0000	CAN Baud Rate Prescaler Extension	700
0x020	CANIF1CRQ	R/W	0x0000.0001	CAN IF1 Command Request	701
0x024	CANIF1CMSK	R/W	0x0000.0000	CAN IF1 Command Mask	702
0x028	CANIF1MSK1	R/W	0x0000.FFFF	CAN IF1 Mask 1	704
0x02C	CANIF1MSK2	R/W	0x0000.FFFF	CAN IF1 Mask 2	705
0x030	CANIF1ARB1	R/W	0x0000.0000	CAN IF1 Arbitration 1	706
0x034	CANIF1ARB2	R/W	0x0000.0000	CAN IF1 Arbitration 2	707
0x038	CANIF1MCTL	R/W	0x0000.0000	CAN IF1 Message Control	709
0x03C	CANIF1DA1	R/W	0x0000.0000	CAN IF1 Data A1	711
0x040	CANIF1DA2	R/W	0x0000.0000	CAN IF1 Data A2	711
0x044	CANIF1DB1	R/W	0x0000.0000	CAN IF1 Data B1	711
0x048	CANIF1DB2	R/W	0x0000.0000	CAN IF1 Data B2	711
0x080	CANIF2CRQ	R/W	0x0000.0001	CAN IF2 Command Request	701
0x084	CANIF2CMSK	R/W	0x0000.0000	CAN IF2 Command Mask	702
0x088	CANIF2MSK1	R/W	0x0000.FFFF	CAN IF2 Mask 1	704
0x08C	CANIF2MSK2	R/W	0x0000.FFFF	CAN IF2 Mask 2	705
0x090	CANIF2ARB1	R/W	0x0000.0000	CAN IF2 Arbitration 1	706
0x094	CANIF2ARB2	R/W	0x0000.0000	CAN IF2 Arbitration 2	707
0x098	CANIF2MCTL	R/W	0x0000.0000	CAN IF2 Message Control	709

Offset	Name	Type	Reset	Description	See page
0x09C	CANIF2DA1	R/W	0x0000.0000	CAN IF2 Data A1	711
0x0A0	CANIF2DA2	R/W	0x0000.0000	CAN IF2 Data A2	711
0x0A4	CANIF2DB1	R/W	0x0000.0000	CAN IF2 Data B1	711
0x0A8	CANIF2DB2	R/W	0x0000.0000	CAN IF2 Data B2	711
0x100	CANTXRQ1	RO	0x0000.0000	CAN Transmission Request 1	712
0x104	CANTXRQ2	RO	0x0000.0000	CAN Transmission Request 2	712
0x120	CANNWDA1	RO	0x0000.0000	CAN New Data 1	713
0x124	CANNWDA2	RO	0x0000.0000	CAN New Data 2	713
0x140	CANMSG1INT	RO	0x0000.0000	CAN Message 1 Interrupt Pending	714
0x144	CANMSG2INT	RO	0x0000.0000	CAN Message 2 Interrupt Pending	714
0x160	CANMSG1VAL	RO	0x0000.0000	CAN Message 1 Valid	715
0x164	CANMSG2VAL	RO	0x0000.0000	CAN Message 2 Valid	715

## 18.4 CAN 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 that 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 clearing `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 cleared, each time a sequence of 11 High bits has been monitored, a `BITERROR0` code is written to the **CANSTS** register (the `LEC` field = 0x5), enabling the CPU to readily check whether the CAN bus is stuck Low or continuously disturbed, and to monitor the proceeding of the bus-off recovery sequence.

### CAN Control (CANCTL)

CAN0 base: 0x4004.0000

CAN1 base: 0x4004.1000

Offset 0x000

Type R/W, reset 0x0000.0001

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	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 <code>INIT</code> 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.

Bit/Field	Name	Type	Reset	Description
3	EIE	R/W	0	Error Interrupt Enable 0: Disabled. No error status interrupt is generated. 1: Enabled. A change in the <i>BOFF</i> or <i>EWARN</i> bits in the <b>CANSTS</b> register generates an interrupt.
2	SIE	R/W	0	Status Interrupt Enable 0: Disabled. No status interrupt is generated. 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 <i>TXOK</i> , <i>RXOK</i> or <i>LEC</i> bits in the <b>CANSTS</b> register generates an interrupt.
1	IE	R/W	0	CAN Interrupt Enable 0: Interrupts disabled. 1: Interrupts enabled.
0	INIT	R/W	1	Initialization 0: Normal operation. 1: Initialization started.

## 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 when a message has been transferred (reception or transmission) without error. The unused error code 7 may be written by the CPU to manually set this field to an invalid error so that it can be checked for a change later.

An error interrupt is generated by the BOFF and EWARN bits and a status interrupt is generated by the RXOK, TXOK, and LEC bits, if 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  
CAN1 base: 0x4004.1000  
Offset 0x004  
Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	BOFF	RO	0	Bus-Off Status 0: CAN controller is not in bus-off state. 1: CAN controller 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 cleared, no message has been successfully received.</p> <p>1: Since this bit was last cleared, a message has been successfully received, independent of the result of the acceptance filtering.</p> <p>This bit is never cleared by the CAN module.</p>
3	TXOK	R/W	0	<p>Transmitted a Message Successfully</p> <p>0: Since this bit was last cleared, no message has been successfully transmitted.</p> <p>1: Since this bit was last cleared, a message has been successfully transmitted error-free and acknowledged by at least one other node.</p> <p>This bit is never cleared by the CAN module.</p>



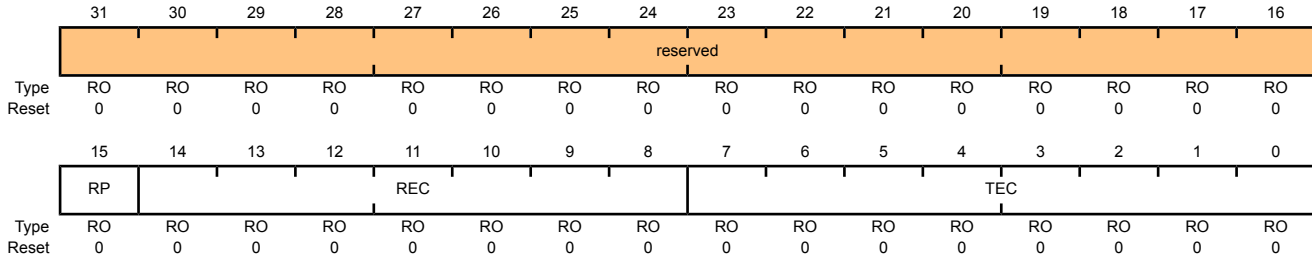
Bit/Field	Name	Type	Reset	Description
2:0	LEC	R/W	0x0	<p>Last Error Code</p> <p>This is the type of the last error to occur on the CAN bus.</p> <p>Value Definition</p> <p>0x0 No Error</p> <p>0x1 Stuff Error</p> <p>More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.</p> <p>0x2 Format Error</p> <p>A fixed format part of the received frame has the wrong format.</p> <p>0x3 ACK Error</p> <p>The message transmitted was not acknowledged by another node.</p> <p>0x4 Bit 1 Error</p> <p>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.</p> <p>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).</p> <p>0x5 Bit 0 Error</p> <p>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).</p> <p>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.</p> <p>0x6 CRC Error</p> <p>The CRC checksum was incorrect in the received message, indicating that the calculated value received did not match the calculated CRC of the data.</p> <p>0x7 Unused</p> <p>When the LEC bit shows this value, no CAN bus event was detected since the CPU wrote this value to LEC.</p>

### 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  
 CAN1 base: 0x4004.1000  
 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	0x00	Receive Error Counter  State of the receiver error counter (0 to 127).
7:0	TEC	RO	0x00	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 programmed to the system clock frequency. This register is write-enabled by setting the `CCE` and `INIT` bits in the `CANCTL` register. See “Bit Time and Bit Rate” on page 683 for more information.

### CAN Bit Timing (CANBIT)

CAN0 base: 0x4004.0000  
CAN1 base: 0x4004.1000  
Offset 0x00C  
Type R/W, reset 0x0000.2301

	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	TSEG2			TSEG1				SJW		BRP					
Type	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	1

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	Time Segment after Sample Point  0x00-0x07: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.  So, for example, a reset value of 0x2 defines that there is 3 (2+1) bit time quanta defined for <code>Phase_Seg2</code> (see Figure 18-4 on page 684). The bit time quanta is defined by the <code>BRP</code> field.
11:8	TSEG1	R/W	0x3	Time Segment Before Sample Point  0x00-0x0F: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.  So, for example, the reset value of 0x3 defines that there is 4 (3+1) bit time quanta defined for <code>Phase_Seg1</code> (see Figure 18-4 on page 684). The bit time quanta is define by the <code>BRP</code> field.
7:6	SJW	R/W	0x0	(Re)Synchronization Jump Width  0x00-0x03: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.  During the start of frame (SOF), if the CAN controller detects a phase error (misalignment), it can adjust the length of <code>TSEG2</code> or <code>TSEG1</code> by the value in <code>SJW</code> . So the reset value of 0 adjusts the length by 1 bit time quanta.

Bit/Field	Name	Type	Reset	Description
5:0	BRP	R/W	0x1	<p>Baud Rate Prescaler</p> <p>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.</p> <p>0x00-0x03F: 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>CANBRPE</b> register can be used to further divide the bit time.</p>

## Register 5: CAN Interrupt (CANINT), offset 0x010

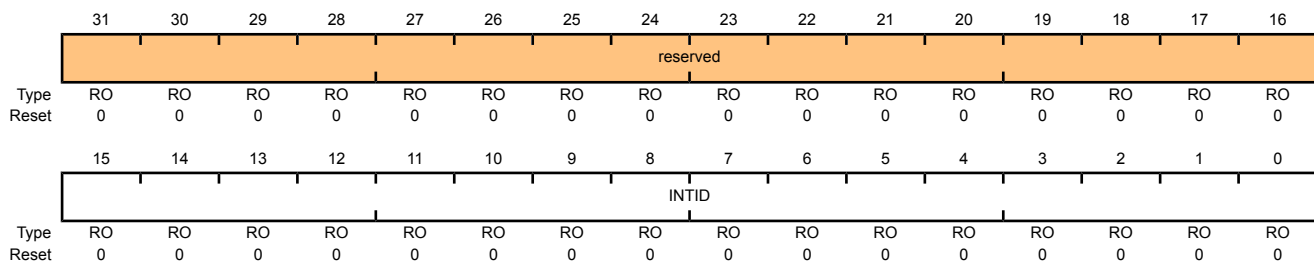
This register indicates the source of the interrupt.

If several interrupts are pending, the **CAN Interrupt (CANINT)** register points to the pending interrupt with the highest priority, disregarding the order in which the interrupts occurred. An interrupt remains pending until the CPU has cleared it. If the **INTID** field 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** field is cleared by reading the **CANSTS** register, or until the **IE** bit in the **CANCTL** register is cleared.

**Note:** Reading the **CAN Status (CANSTS)** register clears the **CAN Interrupt (CANINT)** register, if it is pending.

### CAN Interrupt (CANINT)

CAN0 base: 0x4004.0000  
CAN1 base: 0x4004.1000  
Offset 0x010  
Type RO, reset 0x0000.0000



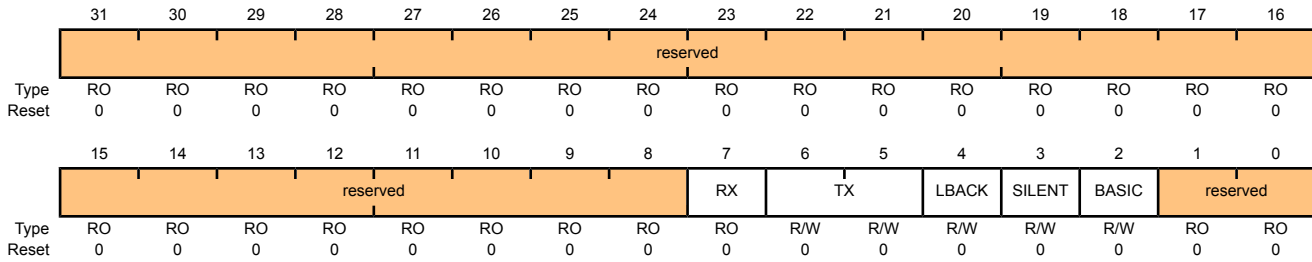
Bit/Field	Name	Type	Reset	Description												
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.												
15:0	INTID	RO	0x0000	Interrupt Identifier The number in this field indicates the source of the interrupt.												
				<table border="1"> <thead> <tr> <th>Value</th> <th>Definition</th> </tr> </thead> <tbody> <tr> <td>0x0000</td> <td>No interrupt pending</td> </tr> <tr> <td>0x0001-0x0020</td> <td>Number of the message object that caused the interrupt</td> </tr> <tr> <td>0x0021-0x7FFF</td> <td>Unused</td> </tr> <tr> <td>0x8000</td> <td>Status Interrupt</td> </tr> <tr> <td>0x8001-0xFFFF</td> <td>Unused</td> </tr> </tbody> </table>	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
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 setting the TEST bit in the CANCTL register. Different test functions may be combined, however, CAN transfers will be affected if the TX bits in this register are not zero.

#### CAN Test (CANTST)

CAN0 base: 0x4004.0000  
 CAN1 base: 0x4004.1000  
 Offset 0x014  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	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.  Value Description 0x0 CANnTx is controlled by the CAN module; default operation 0x1 The sample point is driven on the CANnTx signal. This mode is useful to monitor bit timing. 0x2 CANnTx drives a low value. This mode is useful for checking the physical layer of the CAN bus. 0x3 CANnTx drives a high value. This mode is useful for checking the physical layer of the CAN bus.
4	LBACK	R/W	0	Loopback Mode 0: Disabled. 1: Enabled. In loopback mode, the data from the transmitter is routed into the receiver. Any data on the receive input is ignored.
3	SILENT	R/W	0	Silent Mode Do not transmit data; monitor the bus. Also known as Bus Monitor mode. 0: Disabled. 1: Enabled.

---

Bit/Field	Name	Type	Reset	Description
2	BASIC	R/W	0	Basic Mode 0: Disabled. 1: Use <b>CANIF1</b> registers as transmit buffer, and use <b>CANIF2</b> registers as receive buffer.
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 Prescaler 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 by setting the `CCE` bit in the `CANCTL` register.

#### CAN Baud Rate Prescaler Extension (CANBRPE)

CAN0 base: 0x4004.0000  
 CAN1 base: 0x4004.1000  
 Offset 0x018  
 Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	BRPE	R/W	0x0	Baud Rate Prescaler Extension  0x00-0x0F: Extend the <code>BRP</code> bit in the <code>CANBIT</code> register to values up to 1023. The actual interpretation by the hardware is one more than the value programmed by <code>BRPE</code> (MSBs) and <code>BRP</code> (LSBs).



**Register 8: CAN IF1 Command Request (CANIF1CRQ), offset 0x020****Register 9: CAN IF2 Command Request (CANIF2CRQ), offset 0x080**

A message transfer is started as soon as there is a write of the message object number to the `MNUM` field when the `TXRQST` bit in the `CANIF1MCTL` register is set. With this write operation, the `BUSY` bit is automatically set to indicate that a transfer between the CAN Interface Registers and the internal message RAM 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 clears the `BUSY` bit.

## CAN IF1 Command Request (CANIF1CRQ)

CAN0 base: 0x4004.0000  
 CAN1 base: 0x4004.1000  
 Offset 0x020  
 Type R/W, reset 0x0000.0001

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	BUSY	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	0	Busy Flag 0: Cleared 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. <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x00</td> <td>0 is not a valid message number; it is interpreted as 0x20, or object 32.</td> </tr> <tr> <td>0x01-0x20</td> <td>Indicates specified message object 1 to 32.</td> </tr> <tr> <td>0x21-0x3F</td> <td>Not a valid message number; values are shifted and it is interpreted as 0x01-0x1F.</td> </tr> </tbody> </table>	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.
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**

Reading the Command Mask registers provides status for various functions. Writing to the Command Mask registers specifies the transfer direction and selects which buffer registers are the source or target of the data transfer.

Note that when a read from the message object buffer occurs when the WRNRD bit is clear and the CLRINTPND and/or NEWDAT bits are set, the interrupt pending and/or new data flags in the message object buffer are cleared.

CAN IF1 Command Mask (CANIF1CMSK)

CAN0 base: 0x4004.0000  
 CAN1 base: 0x4004.1000  
 Offset 0x024  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								WRNRD	MASK	ARB	CONTROL	CLRINTPND	NEWDAT / TXRQST	DATAA	DATAB
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	WRNRD	R/W	0	Write, Not Read  Transfer the message object address specified by the <b>CAN Command Request (CANIFnCRQ)</b> register to the CAN message buffer registers.  <b>Note:</b> Interrupt pending and new data conditions in the message buffer can be cleared by reading from the buffer (WRNRD = 0) when the CLRINTPND and/or NEWDAT bits are set.
6	MASK	R/W	0	Access Mask Bits  0: Mask bits unchanged.  1: Transfer IDMASK + DIR + MXTD of the message object into the Interface registers.
5	ARB	R/W	0	Access Arbitration Bits  0: Arbitration bits unchanged.  1: Transfer ID + DIR + XTD + MSGVAL of the message object into the Interface registers.
4	CONTROL	R/W	0	Access Control Bits  0: Control bits unchanged.  1: Transfer control bits from the <b>CANIFnMCTL</b> register into the Interface registers.

Bit/Field	Name	Type	Reset	Description
3	CLRINTPND	R/W	0	<p>Clear Interrupt Pending Bit</p> <p>If WRNRD is set, this bit controls whether the INTPND bit in the <b>CANIFnMCTL</b> register is changed.</p> <p>0: The INTPND bit in the message object remains unchanged.</p> <p>1: The INTPND bit is cleared in the message object.</p> <p>If WRNRD is clear and this bit is clear, the interrupt pending status is transferred from the message buffer into the <b>CANIFnMCTL</b> register.</p> <p>If WRNRD is clear and this bit is set, the interrupt pending status is cleared in the message buffer. Note that the value of this bit that is transferred to the <b>CANIFnMCTL</b> register always reflects the status of the bits before clearing.</p>
2	NEWDAT / TXRQST	R/W	0	<p>NEWDAT / TXRQST Bit</p> <p>If WRNRD is set, this bit can act as a TXRQST bit and request a transmission. Note that when this bit is set, the TXRQST bit in the <b>CANIFnMCTL</b> register is ignored.</p> <p>0: Transmission is not requested</p> <p>1: Begin a transmission</p> <p>If WRNRD is clear and this bit is clear, the value of the new data status is transferred from the message buffer into the <b>CANIFnMCTL</b> register.</p> <p>If WRNRD is clear and this bit is set, the new data status is cleared in the message buffer. Note that the value of this bit that is transferred to the <b>CANIFnMCTL</b> register always reflects the status of the bits before clearing.</p>
1	DATAA	R/W	0	<p>Access Data Byte 0 to 3</p> <p>When WRNRD = 1:</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> <p>When WRNRD = 0:</p> <p>0: Data bytes 0-3 are unchanged.</p> <p>1: Transfer data bytes 0-3 in <b>CANIFnDA1</b> and <b>CANIFnDA2</b> to the message object.</p>
0	DATAB	R/W	0	<p>Access Data Byte 4 to 7</p> <p>When WRNRD = 1:</p> <p>0: Data bytes 4-7 are unchanged.</p> <p>1: Transfer data bytes 4-7 in message object to <b>CANIFnDB1</b> and <b>CANIFnDB2</b>.</p> <p>When WRNRD = 0:</p> <p>0: Data bytes 4-7 are unchanged.</p> <p>1: Transfer data bytes 4-7 in <b>CANIFnDB1</b> and <b>CANIFnDB2</b> to the message object.</p>

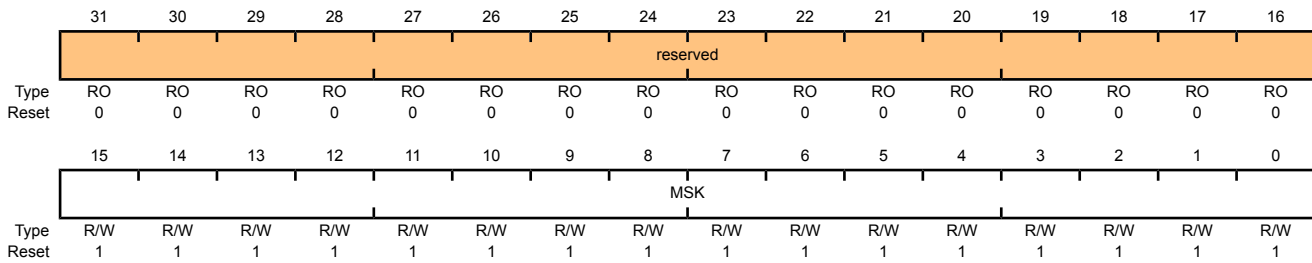
**Register 12: CAN IF1 Mask 1 (CANIF1MSK1), offset 0x028**

**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  
 CAN1 base: 0x4004.1000  
 Offset 0x028  
 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	MSK	R/W	0xFFFF	Identifier Mask  When using a 29-bit identifier, these bits are used for bits [15:0] of the ID. The <b>MSK</b> field in the <b>CANIFnMSK2</b> register are used for bits [28:16] of the ID. When using an 11-bit identifier, these bits are ignored.  0: The corresponding identifier field ( <b>ID</b> ) in the message object cannot inhibit the match in acceptance filtering.  1: The corresponding identifier field ( <b>ID</b> ) is used for acceptance filtering.

**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

CAN1 base: 0x4004.1000

Offset 0x02C

Type R/W, reset 0x0000.FFFF

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	MXTD	R/W	0x1	Mask Extended Identifier  0: The extended identifier bit ( <i>XTD</i> in the <b>CANIFnARB2</b> register) has no effect on the acceptance filtering.  1: The extended identifier bit <i>XTD</i> is used for acceptance filtering.
14	MDIR	R/W	0x1	Mask Message Direction  0: The message direction bit ( <i>DIR</i> in the <b>CANIFnARB2</b> register) has no effect for acceptance filtering.  1: The message direction bit <i>DIR</i> is used for acceptance filtering.
13	reserved	RO	0x1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12:0	MSK	R/W	0xFF	Identifier Mask  When using a 29-bit identifier, these bits are used for bits [28:16] of the ID. The <i>MSK</i> field in the <b>CANIFnMSK1</b> register are used for bits [15:0] of the ID. When using an 11-bit identifier, <i>MSK</i> [12:2] are used for bits [10:0] of the ID.  0: The corresponding identifier field ( <i>ID</i> ) in the message object cannot inhibit the match in acceptance filtering.  1: The corresponding identifier field ( <i>ID</i> ) is used for acceptance filtering.

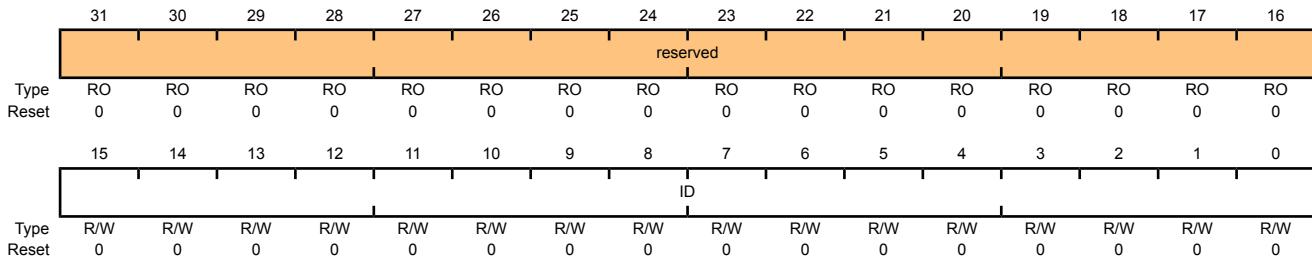
**Register 16: CAN IF1 Arbitration 1 (CANIF1ARB1), offset 0x030**

**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  
 CAN1 base: 0x4004.1000  
 Offset 0x030  
 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	ID	R/W	0x0000	<p>Message Identifier</p> <p>This bit field is used with the <code>ID</code> field in the <code>CANIFnARB2</code> register to create the message identifier.</p> <p>When using a 29-bit identifier, bits 15:0 of the <code>CANIFnARB1</code> register are [15:0] of the ID, while bits 12:0 of the <code>CANIFnARB2</code> register are [28:16] of the ID.</p> <p>When using an 11-bit identifier, these bits are not used.</p>

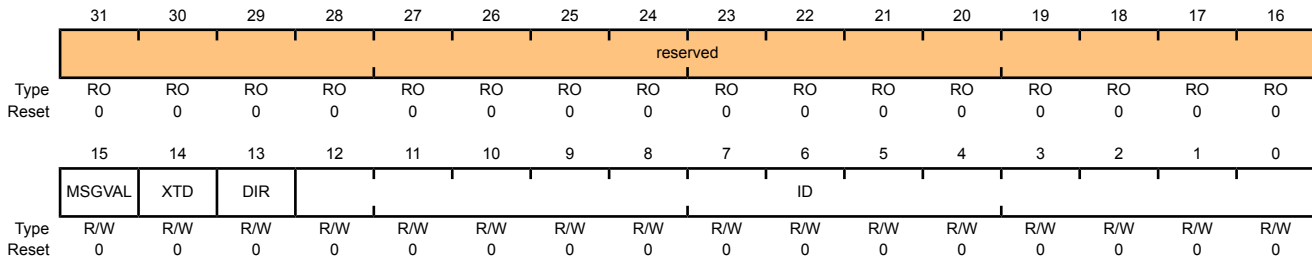
**Register 18: CAN IF1 Arbitration 2 (CANIF1ARB2), offset 0x034**

**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  
 CAN1 base: 0x4004.1000  
 Offset 0x034  
 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	MSGVAL	R/W	0	<p>Message Valid</p> <p>0: The message object is ignored by the message handler.</p> <p>1: The message object is configured and ready to 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> 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> field in the <b>CANIFnMCTL</b> register.</p>
14	XTD	R/W	0	<p>Extended Identifier</p> <p>0: An 11-bit Standard Identifier is used for this message object.</p> <p>1: A 29-bit Extended Identifier is used for this message object.</p>
13	DIR	R/W	0	<p>Message Direction</p> <p>0: Receive. When the <b>TXRQST</b> bit in the <b>CANIFnMCTL</b> register is set, a remote frame with the identifier of this message object is received. On reception of a data frame with matching identifier, that message is stored in this message object.</p> <p>1: Transmit. When the <b>TXRQST</b> bit in the <b>CANIFnMCTL</b> register is set, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the <b>TXRQST</b> bit of this message object is set (if <b>RMTEN</b>=1).</p>

Bit/Field	Name	Type	Reset	Description
12:0	ID	R/W	0x000	<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.</p> <p>When using a 29-bit identifier, <code>ID[15:0]</code> of the <b>CANIFnARB1</b> register are [15:0] of the ID, while these bits, <code>ID[12:0]</code>, are [28:16] of the ID.</p> <p>When using an 11-bit identifier, <code>ID[12:2]</code> are used for bits [10:0] of the ID. The <code>ID</code> field in the <b>CANIFnARB1</b> register is ignored.</p>



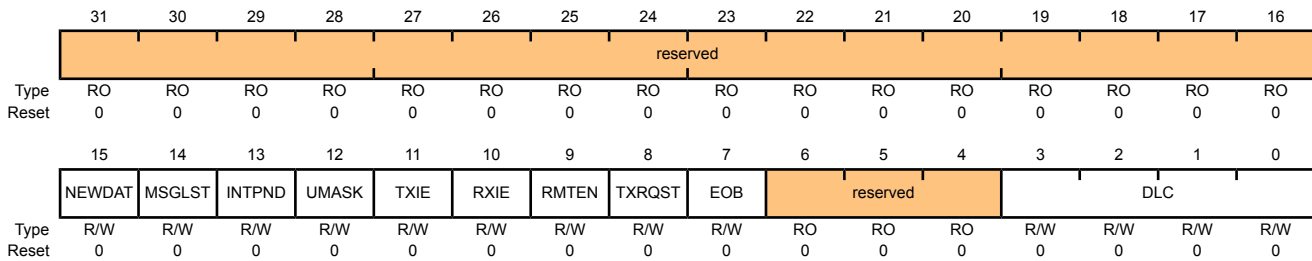
**Register 20: CAN IF1 Message Control (CANIF1MCTL), offset 0x038**

**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  
 CAN1 base: 0x4004.1000  
 Offset 0x038  
 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	NEWDAT	R/W	0	New Data  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.  1: The message handler or the CPU has written new data into the data portion of this message object.
14	MSGLST	R/W	0	Message Lost  0 : No message was lost since the last time this bit was cleared by the CPU.  1: The message handler stored a new message into this object when NEWDAT was set; the CPU has lost a message.  This bit is only valid for message objects when the DIR bit in the <b>CANIFnARB2</b> register clear (receive).
13	INTPND	R/W	0	Interrupt Pending  0: This message object is not the source of an interrupt.  1: This message object is the source of an interrupt. The interrupt identifier in the <b>CANINT</b> register points to this message object if there is not another interrupt source with a higher priority.
12	UMASK	R/W	0	Use Acceptance Mask  0: Mask ignored.  1: Use mask (MSK, MXTD, and MDIR bits in the <b>CANIFnMSKn</b> registers) for acceptance filtering.

Bit/Field	Name	Type	Reset	Description						
11	TXIE	R/W	0	<p>Transmit Interrupt Enable</p> <p>0: The <code>INTPND</code> bit in the <code>CANIFnMCTL</code> register is unchanged after a successful transmission of a frame.</p> <p>1: The <code>INTPND</code> bit in the <code>CANIFnMCTL</code> register is set after a successful transmission of a frame.</p>						
10	RXIE	R/W	0	<p>Receive Interrupt Enable</p> <p>0: The <code>INTPND</code> bit in the <code>CANIFnMCTL</code> register is unchanged after a successful reception of a frame.</p> <p>1: The <code>INTPND</code> bit in the <code>CANIFnMCTL</code> register is set after a successful reception of a frame.</p>						
9	RMTEN	R/W	0	<p>Remote Enable</p> <p>0: At the reception of a remote frame, the <code>TXRQST</code> bit in the <code>CANIFnMCTL</code> register is left unchanged.</p> <p>1: At the reception of a remote frame, the <code>TXRQST</code> bit in the <code>CANIFnMCTL</code> register is set.</p>						
8	TXRQST	R/W	0	<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	0	<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.</p>						
6:4	reserved	RO	0x0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>						
3:0	DLC	R/W	0x0	<p>Data Length Code</p> <table border="1"> <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> field in the <code>CANIFnMCTL</code> 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**

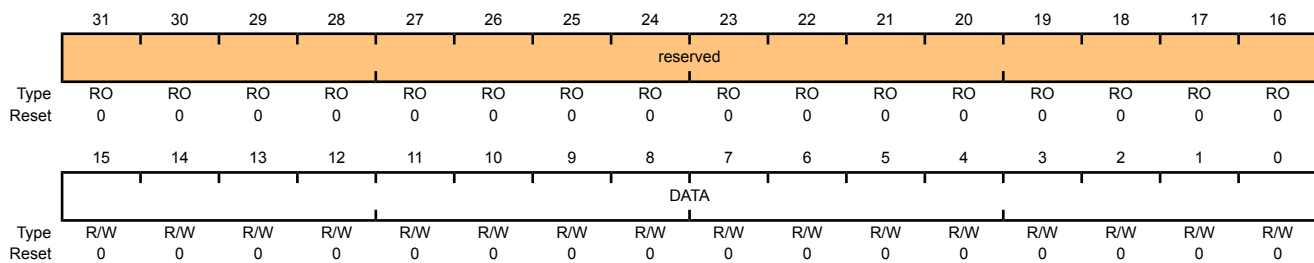
**Register 28: CAN IF2 Data B1 (CANIF2DB1), offset 0x0A4**

**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  
CAN1 base: 0x4004.1000  
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	0x0000	Data

The **CANIFnDA1** registers contain data bytes 1 and 0; **CANIFnDA2** data bytes 3 and 2; **CANIFnDB1** data bytes 5 and 4; and **CANIFnDB2** data bytes 7 and 6.

**Register 30: CAN Transmission Request 1 (CANTXRQ1), offset 0x100**

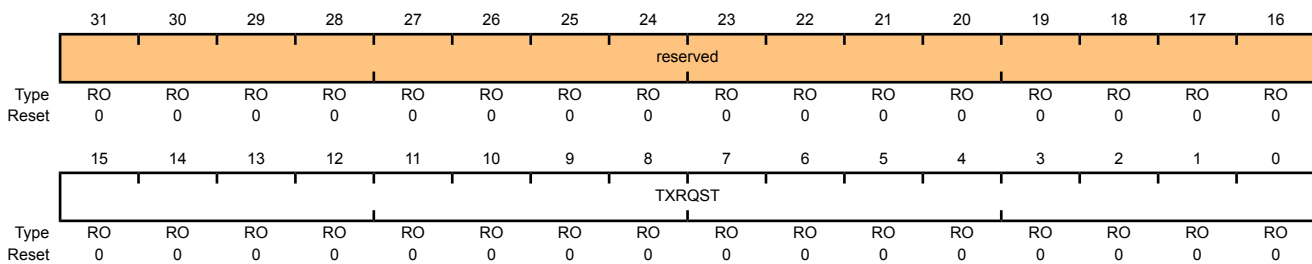
**Register 31: CAN Transmission Request 2 (CANTXRQ2), offset 0x104**

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 **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** bits of the first 16 message objects in the message RAM; the **CANTXRQ2** register contains the **TXRQST** bits of the second 16 message objects.

CAN Transmission Request 1 (CANTXRQ1)

CAN0 base: 0x4004.0000  
 CAN1 base: 0x4004.1000  
 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	0x0000	Transmission Request Bits 0: The corresponding message object is not waiting for transmission. 1: The transmission of the corresponding message object is requested and is not yet done.

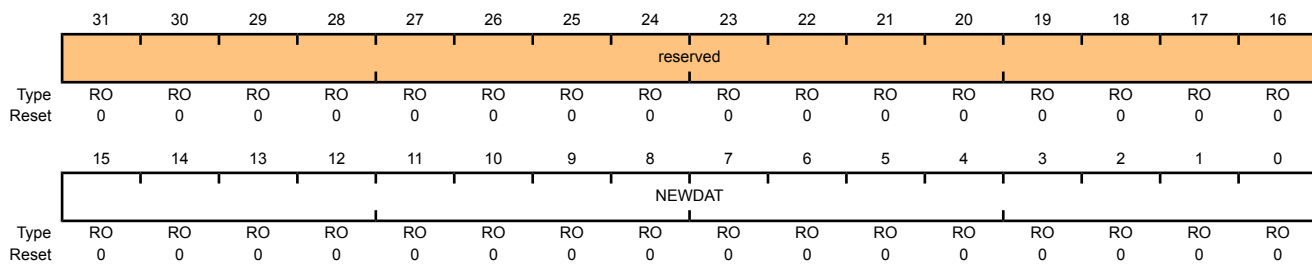
**Register 32: CAN New Data 1 (CANNWDA1), offset 0x120****Register 33: CAN New Data 2 (CANNWDA2), offset 0x124**

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 **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** bits of the first 16 message objects in the message RAM; the **CANNWDA2** register contains the **NEWDAT** bits of the second 16 message objects.

## CAN New Data 1 (CANNWDA1)

CAN0 base: 0x4004.0000  
 CAN1 base: 0x4004.1000  
 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	0x0000	New Data Bits  0: No new data has been written into the data portion of the corresponding message object by the message handler since the last time this flag was cleared by the CPU.  1: The message handler or the CPU has written new data into the data portion of the corresponding message object.

**Register 34: CAN Message 1 Interrupt Pending (CANMSG1INT), offset 0x140**

**Register 35: CAN Message 2 Interrupt Pending (CANMSG2INT), offset 0x144**

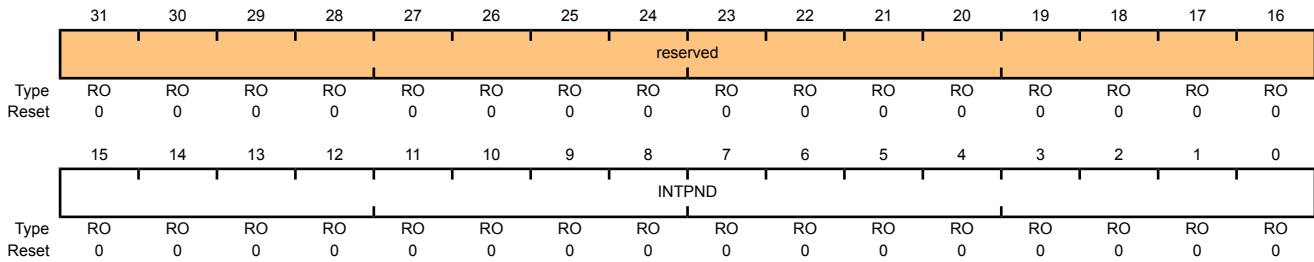
The **CANMSG1INT** and **CANMSG2INT** registers hold the **INTPND** bits of the 32 message objects. By reading these bits, the CPU can check which message object has an interrupt pending. The **INTPND** bit of a specific message object can be changed through two sources: (1) the CPU via the **CANIFnMCTL** register, or (2) the message handler state machine after the reception or transmission of a frame.

This field is also encoded in the **CANINT** register.

The **CANMSG1INT** register contains the **INTPND** bits of the first 16 message objects in the message RAM; the **CANMSG2INT** register contains the **INTPND** bits of the second 16 message objects.

CAN Message 1 Interrupt Pending (CANMSG1INT)

CAN0 base: 0x4004.0000  
 CAN1 base: 0x4004.1000  
 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	0x0000	Interrupt Pending Bits 0: The corresponding message object is not the source of an interrupt. 1: The corresponding 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 **CANIFnMCTL** register.

The **CANMSG1VAL** register contains the **MSGVAL** bits of the first 16 message objects in the message RAM; the **CANMSG2VAL** register contains the **MSGVAL** bits of the second 16 message objects in the message RAM.

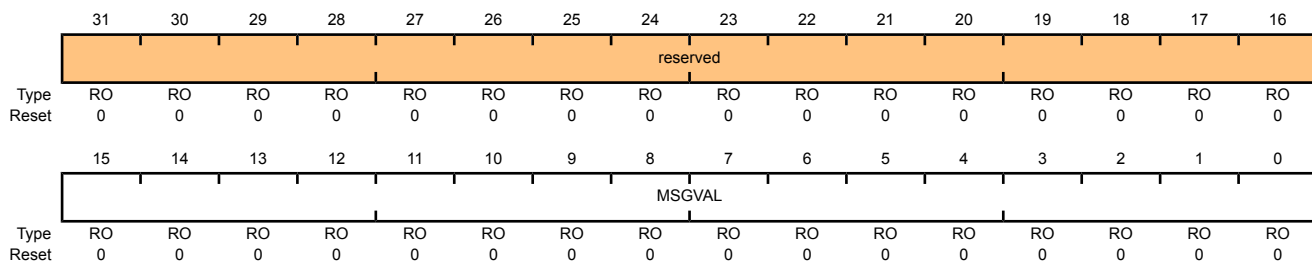
**CAN Message 1 Valid (CANMSG1VAL)**

CAN0 base: 0x4004.0000

CAN1 base: 0x4004.1000

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	0x0000	Message Valid Bits 0: The corresponding message object is not configured and is ignored by the message handler. 1: The corresponding message object is configured and should be considered by the message handler.

## 19 Universal Serial Bus (USB) Controller

The Stellaris<sup>®</sup> USB controller operates as a full-speed or low-speed function controller during point-to-point communications with USB host, device, or OTG functions. The controller complies with the USB 2.0 standard, which includes suspend and resume signaling. Eight endpoints including two hard-wired for control transfers (one endpoint for IN and one endpoint for OUT) plus six endpoints defined by firmware along with a dynamic sizable FIFO support multiple packet queueing.  $\mu$ DMA access to the FIFO allows minimal interference from system software. Software-controlled connect and disconnect allows flexibility during USB device start-up. The controller complies with OTG standard's session request protocol (SRP) and host negotiation protocol (HNP).

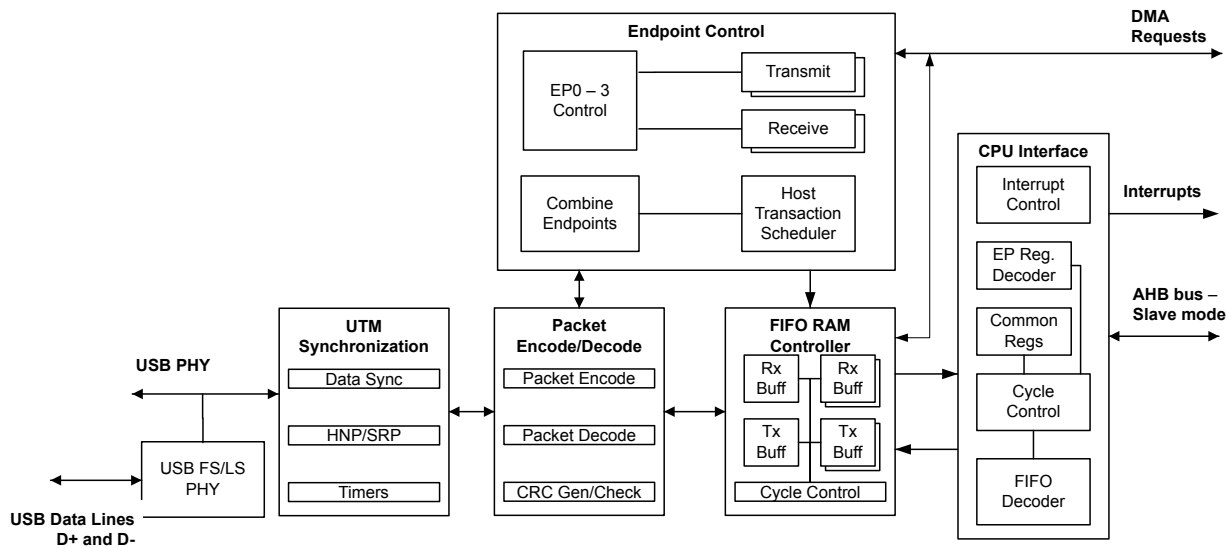
The Stellaris<sup>®</sup> USB module has the following features:

- Complies with USB-IF certification standards
- USB 2.0 full-speed (12 Mbps) and low-speed (1.5 Mbps) operation
- Integrated PHY
- 4 transfer types: Control, Interrupt, Bulk, and Isochronous
- 16 endpoints
  - 1 dedicated control IN endpoint and 1 dedicated control OUT endpoint
  - 7 configurable IN endpoints and 7 configurable OUT endpoints
- 4 KB dedicated endpoint memory - one endpoint may be defined for double-buffered 1023-byte isochronous packet size
- Efficient transfers using Micro Direct Memory Access Controller ( $\mu$ DMA)
  - Separate channels for transmit and receive for up to 3 IN endpoints and 3 OUT endpoints
  - Burst requests
  - Channel requests asserted when FIFO contains required amount of data



## 19.1 Block Diagram

Figure 19-1. USB Module Block Diagram



## 19.2 Functional Description

**Note:** A 9.1-k $\Omega$  resistor should be connected between the USB0RBIAS and ground. The 9.1-k $\Omega$  resistor should have a 1% tolerance and should be located in close proximity to the USB0RBIAS pin. Power dissipation in the resistor is low, so a chip resistor of any geometry may be used.

The Stellaris<sup>®</sup> USB controller provides full OTG negotiation and support for connection to non-OTG peripherals or host controllers. It supports both the session request protocol (SRP) and the host negotiation protocol (HNP) to provide full OTG support. The session request protocol allows devices on the B side of a cable to request that the A side device turn on VBUS. The host negotiation protocol is used after the initial session request protocol has powered the bus and provides a method to determine which end of the cable will act as the host controller. When the device is connected to non-OTG peripherals or devices, the controller can detect which cable end was used and provides a register to indicate if the controller should act as the host or the device controller. This indication and the mode of operation are handled automatically by the USB controller. This auto-detection allows the system to use a single A/B connector instead of having both A and B connectors in the system. It also allows for full OTG negotiations with other OTG devices.

**Note:** When USB is used in the system, the minimum system frequency is 20 MHz.

### 19.2.1 Operation as a Device

This section describes the Stellaris<sup>®</sup> USB controller's actions when it is being used as a USB device. IN endpoints, OUT endpoints, entry into and exit from Suspend mode, and recognition of Start of Frame (SOF) are all described.

When in device mode, IN transactions are controlled by an endpoint's transmit interface and use the transmit endpoint registers for the given endpoint. OUT transactions are handled with an endpoint's receive interface and use the receive endpoint registers for the given endpoint.

When configuring the size of the FIFOs for endpoints, take into account the maximum packet size for an endpoint.

- **Bulk.** Bulk endpoints should be sized to be multiples of the maximum packet size (up to 64 bytes). For instance, if maximum packet size is 64 bytes, the FIFO should be configured to a multiple of 64-byte packets (64, 128, 192, or 256 bytes). This allows for efficient use of double buffering or packet splitting (described further in the following sections).
- **Interrupt.** Interrupt endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used.
- **Isochronous.** Isochronous endpoints are more flexible and can be up to 1023 bytes.
- **Control.** It is also possible to specify a separate control endpoint for a USB device. However, in most cases the USB device should use the dedicated control endpoint on the USB controller's endpoint 0.

### 19.2.1.1 Endpoints

When operating as a device, there are two dedicated control endpoints (IN and OUT) and six configurable endpoints (3 IN and 3 OUT) that can be used for communications with a host controller. The endpoint number associated with an endpoint is directly related to its register designation. For example, when the host is communicating with endpoint 1, all events will occur in the endpoint 1 register interface.

Endpoint 0 is a dedicated control endpoint used for all control transactions to endpoint 0 during enumeration or when any other control requests are made to endpoint 0. Endpoint 0 uses the first 64 bytes of the USB controller's FIFO RAM as a shared memory for both IN and OUT transactions.

The remaining six endpoints can be configured as control, bulk, interrupt, or isochronous endpoints. They should be treated as three configurable IN and three configurable OUT endpoints. The three endpoint pairs (endpoint 1, 2, and 3) are not required to have the same type for their IN and OUT endpoint configuration. For example, the OUT portion of an endpoint pair could be a bulk endpoint, while the IN portion of that endpoint pair could be an interrupt endpoint. The address and size of the FIFOs attached to each endpoint can be modified to fit the application's needs.

### 19.2.1.2 IN Transactions as a Device

When operating as a USB device, data for IN transactions is handled through the FIFOs attached to the transmit endpoints. The sizes of the FIFOs for the three configurable IN endpoints are determined by the **USBTXFIFOADD** register. The maximum size of a data packet that may be placed in a transmit endpoint's FIFO for transmission is programmable and is determined by the value written to the **USBTXMAXPn** register for that endpoint. The endpoint's FIFO can also be configured to use double-packet or single-packet buffering. When double-packet buffering is enabled, two data packets can be buffered in the FIFO, which also requires that the FIFO is at least two packets in size. When double-packet buffering is disabled, only one packet can be buffered, even if the packet size is less than half the FIFO size. The USB controller also supports a special mode for bulk endpoints that allows automatic splitting of a larger FIFO into multiple packets that are maximum packet size transfers.

**Note:** The maximum packet size set for any endpoint must not exceed the FIFO size. The **USBTXMAXPn** register should not be written to while there is data in the FIFO as unexpected results may occur.

### Single-Packet Buffering

If the size of the transmit endpoint's FIFO is less than twice the maximum packet size for this endpoint (as set in the **USBTXFIFOSZ** register), only one packet can be buffered in the FIFO and single-packet buffering is required. When each packet is completely loaded into the transmit FIFO, the **TXRDY** bit in the **USBTXCSRLn** register needs to be set. If the **AUTOSET** bit in the **USBTXCSRHn** register is set, the **TXRDY** bit is automatically set when a maximum sized packet is loaded into the FIFO. For packet sizes less than the maximum, the **TXRDY** bit must be set manually. When the **TXRDY** bit is set, either manually or automatically, the packet is ready to be sent. When the packet has been successfully sent, both **TXRDY** and **FIFONE** are cleared and the appropriate transmit endpoint interrupt signaled. At this point, the next packet can be loaded into the FIFO.

### Double-Packet Buffering

If the size of the transmit endpoint's FIFO is at least twice the maximum packet size for this endpoint, two packets can be buffered in the FIFO and double-packet buffering is allowed. As each packet is loaded into the transmit FIFO, the **TXRDY** bit in the **USBTXCSRLn** register needs to be set. If the **AUTOSET** bit in the **USBTXCSRHn** register is set, the **TXRDY** bit is automatically set when a maximum sized packet is loaded into the FIFO. For packet sizes less than the maximum, **TXRDY** must be set manually. When the **TXRDY** bit is set, either manually or automatically, the packet is ready to be sent. After the first packet is loaded, **TXRDY** is immediately cleared and an interrupt is generated. A second packet can now be loaded into the transmit FIFO and **TXRDY** set again (either manually or automatically if the packet is the maximum size). At this point, both packets are ready to be sent. After each packet has been successfully sent, **TXRDY** is cleared and the appropriate transmit endpoint interrupt signaled to indicate that another packet can now be loaded into the transmit FIFO. The state of the **FIFONE** bit at this point indicates how many packets may be loaded. If the **FIFONE** bit is set, then there is another packet in the FIFO and only one more packet can be loaded. If the **FIFONE** bit is clear, then there are no packets in the FIFO and two more packets can be loaded.

**Note:** Double-packet buffering is disabled if an endpoint's corresponding **EPn** bit is set in the **USBTXDPKTBUFDIS** register. This bit is set by default, so it must be cleared to enable double-packet buffering.

### Special Bulk Handling

The packets transferred in bulk operations are defined by the USB specification to be 8, 16, 32 or 64 bytes in size. For some system designs, however, it may be more convenient for the application software to write larger amounts of data to an endpoint in a single operation than can be transferred in a single USB operation.

To simplify this case, the Stellaris<sup>®</sup> USB controller includes a packet-splitting feature that allows larger data packets to be written to bulk transmit endpoints, which are then split into packets of an appropriate size for transfer across the USB bus. With this option, the **USBTXMAXPn** register uses the bottom 11 bits to define the payload for each individual transfer, while the top 5 bits define a multiplier. The application software can then write data packets of size multiplier × payload to the FIFO, which the USB controller then splits into individual packets of the stated payload for transmission over the USB bus. From the application software's point-of-view, the resulting operation does not differ from the transmission of a single USB packet except in the size of the packet written.

**Note:** Packet-splitting can only be used with bulk endpoints and, in accordance with the USB specification, the payload must be 8, 16, 32, or 64. The payload recorded in the **USBTXMAXPn** register must also match the **wMaxPacketSize** field of the Standard Endpoint Descriptor for the endpoint (see chapter 9 of the USB specification). The associated FIFO must also be large enough to accommodate the data packet prior to being split.

### 19.2.1.3 OUT Transactions as a Device

When in device mode, OUT transactions are handled through the USB controller receive FIFOs. The sizes of the receive FIFOs for the three configurable OUT endpoints are determined by the **USBRXFIFOADD** register. The maximum amount of data received by an endpoint in any packet is determined by the value written to the **USBRXMAXPn** register for that endpoint. When double-packet buffering is enabled, two data packets can be buffered in the FIFO. When double-packet buffering is disabled, only one packet can be buffered even if the packet is less than half the FIFO size. The Stellaris<sup>®</sup> USB controller also supports a special mode for bulk endpoints that allows automatic splitting of a larger FIFO into multiple maximum packet size transfers.

**Note:** In all cases, the maximum packet size must not exceed the FIFO size.

#### **Single-Packet Buffering**

If the size of the receive endpoint FIFO is less than twice the maximum packet size for an endpoint, only one data packet can be buffered in the FIFO and single-packet buffering is required. When a packet is received and placed in the receive FIFO, the **RXRDY** and **FULL** bits in the **USBRXCSSLn** register are set and the appropriate receive endpoint is signaled, indicating that a packet can now be unloaded from the FIFO. After the packet has been unloaded, the **RXRDY** bit needs to be cleared in order to allow further packets to be received. This action also generates the acknowledge signaling to the host controller. If the **AUTOCL** bit in the **USBRXCSSLn** register is set and a maximum-sized packet is unloaded from the FIFO, the **RXRDY** and **FULL** bits are cleared automatically. For packet sizes less than the maximum, **RXRDY** must be cleared manually.

#### **Double-Packet Buffering**

If the size of the receive endpoint FIFO is at least twice the maximum packet size for the endpoint, two data packets can be buffered and double-packet buffering can be used. When the first packet is received and loaded into the receive FIFO, the **RXRDY** bit in the **USBRXCSSLn** register is set and the appropriate receive endpoint interrupt is signaled to indicate that a packet can now be unloaded from the FIFO.

**Note:** The **FULL** bit in **USBRXCSSLn** is not set when the first packet is received. It is only set if a second packet is received and loaded into the receive FIFO.

After each packet has been unloaded, the **RXRDY** bit needs to be cleared in order to allow further packets to be received. If the **AUTOCL** bit in the **USBRXCSSLn** register is set and a maximum-sized packet is unloaded from the FIFO, the **RXRDY** bit is cleared automatically. For packet sizes less than the maximum, **RXRDY** must be cleared manually. If the **FULL** bit was set when **RXRDY** is cleared, the USB controller first clears the **FULL** bit. It then sets **RXRDY** again to indicate that there is another packet waiting in the FIFO to be unloaded.

**Note:** Double-packet buffering is disabled if an endpoint's corresponding **EPn** bit is set in the **USBRXDPKTBUFDIS** register. This bit is set by default, so it must be cleared to enable double-packet buffering.

#### **Special Bulk Handling**

The packets transferred in bulk operations are defined by the USB specification to be 8, 16, 32, or 64 bytes in size. For some system designs, however, it may be more convenient for the application software to read larger amounts of data from an endpoint in a single operation than can be transferred in a single USB operation.

To simplify this case, the Stellaris<sup>®</sup> USB controller includes a packet-combining feature that combines the packets received across the USB bus into larger data packets prior to being read by the application software. With this option, the **USBRXMAXPn** register uses the bottom 11 bits to define

the payload for each individual transfer, while the top 5 bits define a multiplier. The USB controller then combines the appropriate number of USB packets it receives into a single data packet of size multiplier × payload within the FIFO before asserting `RXRDY` to alert the application software that a packet in the FIFO is ready to be read. The size of the resulting packet is reported in the **USBRXCOUNTn** register. From the application software's point-of-view, the resulting operation does not differ from the receipt of a single USB packet except in the size of the packet read.

**Note:** Packet-combining can only be used with bulk endpoints. The payload recorded in the **USBRXMAXPn** register must also match the `wMaxPacketSize` field of the Standard Endpoint Descriptor for the endpoint (see chapter 9 of the USB specification). The associated FIFO must also be large enough to accommodate the combined data packet.

The `RXRDY` bit is only set when either the specified number of packets have been received or a "short" USB packet is received (that is, a packet of less than the specified payload for the endpoint). If a protocol is being used in which the endpoint receives bulk transfers that are a multiple of the recorded payload size with no short packet to terminate it, the **USBRXMAXPn** register should not be programmed to expect more packets than there are in the transfer (otherwise, the software will not be interrupted at the end of the transfer).

#### 19.2.1.4 Scheduling

The device has no control over the scheduling of transactions as this is determined by the host controller. The Stellaris<sup>®</sup> USB controller can set up a transaction at any time. The USB controller will wait for the request from the host controller and generate an interrupt when the transaction is complete or if it was terminated due to some error. If the host controller makes a request and the device controller is not ready, the USB controller sends a busy response (NAK) to all requests until it is ready.

#### 19.2.1.5 Additional Actions

The USB controller responds automatically to certain conditions on the USB bus or actions by the host controller: when the USB controller automatically stalls a control transfer and unexpected zero length OUT data packets.

##### ***Stalled Control Transfer***

The USB controller automatically issues a STALL handshake to a control transfer under the following conditions:

1. The host sends more data during an OUT data phase of a control transfer than was specified in the device request during the SETUP phase. This condition is detected by the USB controller when the host sends an OUT token (instead of an IN token) after the last OUT packet has been unloaded and the `DATAEND` bit in the **USBCSRL0** register has been set.
2. The host requests more data during an IN data phase of a control transfer than was specified in the device request during the SETUP phase. This condition is detected by the USB controller when the host sends an IN token (instead of an OUT token) after the CPU has cleared `TXRDY` and set `DATAEND` in response to the ACK issued by the host to what should have been the last packet.
3. The host sends more than **USBRXMAXPn** bytes of data with an OUT data token.
4. The host sends more than a zero length data packet for the OUT status phase.

### **Zero Length OUT Data Packets**

A zero-length OUT data packet is used to indicate the end of a control transfer. In normal operation, such packets should only be received after the entire length of the device request has been transferred.

However, if the host sends a zero-length OUT data packet before the entire length of device request has been transferred, it is signaling the premature end of the transfer. In this case, the USB controller automatically flushes any IN token ready for the data phase from the FIFO and sets the `SETUP` bit in the `USBCSRL0` register.

### **Setting the Device Address**

When a host device is attempting to enumerate the USB device, it requests that the device change its address from zero to some other value. The address is changed by writing the value that the host requested to the `USBFADDR` register. However, care should be taken when writing to `USBFADDR` to avoid changing the address before the transaction is complete. This register should only be set after the `SET_ADDRESS` command is complete. Like all control transactions, the transaction is only complete after the device has left the STATUS phase. In the case of a `SET_ADDRESS` command, the transaction is completed by responding to the IN request from the host with a zero-byte packet. Once the device has responded to the IN request, the `USBFADDR` register should be programmed to the new value as soon as possible to avoid missing any new commands sent to the new address.

**Note:** If the `USBFADDR` register is set to the new value as soon as the device receives the OUT transaction with the `SET_ADDRESS` command in the packet, it changes the address during the control transfer. In this case, the device does not receive the IN request that allows the USB transaction to exit the STATUS phase of the control transfer because it is sent to the old address. As a result, the host does not get a response to the IN request, and the host fails to enumerate the device.

## **19.2.1.6 Device Mode Suspend**

When no activity has occurred on the USB bus for 3 ms, the USB controller automatically enters Suspend mode. If the Suspend interrupt has been enabled, an interrupt is generated at this time. When in Suspend mode, the PHY also goes into Suspend mode. When Resume signaling is detected, the USB controller exits Suspend mode and takes the PHY out of Suspend. If the Resume interrupt is enabled, an interrupt is generated. The USB controller can also be forced to exit Suspend mode by setting the `RESUME` bit in the `USBPOWER` register. When this bit is set, the USB controller exits Suspend mode and drives Resume signaling onto the bus. The `RESUME` bit is cleared after 10 ms (a maximum of 15 ms) to end Resume signaling.

To meet USB power requirements, the controller can be put into Deep Sleep. This keeps the controller in a static state.

## **19.2.1.7 Start-of-Frame**

When the USB controller is operating in device mode, it receives a Start-Of-Frame packet from the host once every millisecond. When the SOF packet is received, the 11-bit frame number contained in the packet is written into the `USBFAME` register and an SOF interrupt is also signaled and can be handled by the application. Once the USB controller has started to receive SOF packets, it expects one every millisecond. If no SOF packet is received after 1.00358 ms, it is assumed that the packet has been lost and the `USBFAME` register is not updated. The USB controller continues and resynchronizes these pulses to the received SOF packets when these packets are successfully received again.



### 19.2.1.8 USB Reset

When the USB controller is in device mode and a reset condition is detected on the USB bus, the USB controller automatically performs the following actions:

- Clears the **USBFADDR** register.
- Clears the **USBEPIDX** register.
- Flushes all endpoint FIFOs.
- Clears all control/status registers.
- Enables all endpoint interrupts.
- Generates a reset interrupt.

When the application software driving the USB controller receives a reset interrupt, it closes any open pipes and waits for bus enumeration to begin.

### 19.2.1.9 Connect/Disconnect

The USB controller connection to the USB bus is controlled by software. The USB PHY can be switched between normal mode and non-driving mode by setting or clearing the **SOFTCONN** bit of the **USBPOWER** register. When this **SOFTCONN** bit is set, the PHY is placed in its normal mode and the **USB0DP/USB0DM** lines of the USB bus are enabled. At the same time, the USB controller is placed into a state, in which it will not respond to any USB signaling except a USB reset.

When the **SOFTCONN** bit is cleared, the PHY is put into non-driving mode, **USB0DP** and **USB0DM** are tristated, and the USB controller appears to other devices on the USB bus as if it has been disconnected. This is the default so the USB controller appears disconnected until the **SOFTCONN** bit has been set. The application software can then choose when to set the PHY into its normal mode. Systems with a lengthy initialization procedure may use this to ensure that initialization is complete and the system is ready to perform enumeration before connecting to the USB. Once the **SOFTCONN** bit has been set, the USB controller can be disconnected by clearing this bit.

**Note:** The USB controller does not generate an interrupt when the device is connected to the host. However, an interrupt is generated when the host terminates a session.

## 19.2.2 Operation as a Host

When the Stellaris<sup>®</sup> USB controller is operating in host mode, it can either be used for point-to-point communications with another USB device or, when attached to a hub, for communication with multiple devices. Full-speed and low-speed USB devices are supported, both for point-to-point communication and for operation through a hub. The USB controller automatically carries out the necessary transaction translation needed to allow a low-speed or full-speed device to be used with a USB 2.0 hub. Control, bulk, isochronous, and interrupt transactions are supported. This section describes the USB controller's actions when it is being used as a USB host. Configuration of IN endpoints, OUT endpoints, entry into and exit from Suspend mode, and reset are all described.

When in host mode, IN transactions are controlled by an endpoint's receive interface. All IN transactions use the receive endpoint registers and all OUT endpoints use the transmit endpoint registers for a given endpoint. As in device mode, the FIFOs for endpoints should take into account the maximum packet size for an endpoint.

- **Bulk.** Bulk endpoints should be sized to be multiples of the maximum packet size (up to 64 bytes). For instance, if maximum packet size is 64 bytes, the FIFO should be configured to a multiple of 64-byte packets (64, 128, 192, or 256 bytes). This allows for efficient use of double buffering or packet splitting (described further in the following sections).
- **Interrupt.** Interrupt endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used.
- **Isochronous.** Isochronous endpoints are more flexible and can be up to 1023 bytes.
- **Control.** It is also possible to specify a separate control endpoint to communicate with a device. However, in most cases the USB controller should use the dedicated control endpoint to communicate with a device's endpoint 0.

### 19.2.2.1 Endpoints

The endpoint registers are used to control the USB endpoint interfaces used to communicate with device(s) that are connected. There is a dedicated control IN endpoint, a dedicated control OUT endpoint, three configurable OUT endpoints, and three configurable IN endpoints.

The dedicated control interface can only be used for control transactions to endpoint 0 of devices. These control transactions are used during enumeration or other control functions that communicate using endpoint 0 of devices. This control endpoint shares the first 64 bytes of the USB controller's FIFO RAM for IN and OUT transactions. The remaining IN and OUT interfaces can be configured to communicate with control, bulk, interrupt, or isochronous device endpoints.

These USB interfaces can be used to simultaneously schedule as many as three independent OUT and three independent IN transactions to any endpoints on any device. The IN and OUT controls are paired in three sets of registers. However, they can be configured to communicate with different types of endpoints and different endpoints on devices. For example, the first pair of endpoint controls can be split so that the OUT portion is communicating with a device's bulk OUT endpoint 1, while the IN portion is communicating with a device's interrupt IN endpoint 2.

Before accessing any device, whether for point-to-point communications or for communications via a hub, the relevant **USBRXFUNCADDR<sub>n</sub>** or **USBTXFUNCADDR<sub>n</sub>** registers need to be set for each receive or transmit endpoint to record the address of the device being accessed.

The USB controller also supports connections to devices through a USB hub by providing a register that specifies the hub address and port of each USB transfer. The FIFO address and size are customizable and can be specified for each USB IN and OUT transfer. This includes allowing one FIFO per transaction, sharing a FIFO across transactions, and allowing for double-buffered FIFOs.

### 19.2.2.2 IN Transactions as a Host

IN transactions are handled in a similar manner to the way in which OUT transactions are handled when the USB controller is in Device mode except that the transaction first needs to be initiated by setting the **REQPKT** bit in **USBCSRL0**. This indicates to the transaction scheduler that there is an active transaction on this endpoint. The transaction scheduler then sends an IN token to the target device. When the packet is received and placed in the receive FIFO, the **RXRDY** bit in **USBCSRL0** is set and the appropriate receive endpoint interrupt is signaled to indicate that a packet can now be unloaded from the FIFO.

When the packet has been unloaded, **RXRDY** should be cleared. The **AUTOCL** bit in the **USBRXCSRH<sub>n</sub>** register can be used to have **RXRDY** automatically cleared when a maximum-sized packet has been unloaded from the FIFO. There is also an **AUTORQ** bit in **USBRXCSRH<sub>n</sub>** which causes the **REQPKT** bit to be automatically set when the **RXRDY** bit is cleared. The **AUTOCL** and



`AUTORQ` bits can be used with  $\mu$ DMA accesses to perform complete bulk transfers without main processor intervention. When the `RXRDY` bit is cleared, the controller will send an acknowledge to the device. When there is a known number of packets to be transferred, the `USBRQPKTCOUNTn` register associated with the endpoint should be set to the number of packets to be transferred. The USB controller decrements the value in the `USBRQPKTCOUNTn` register following each request. When the `USBRQPKTCOUNTn` value decrements to 0, the `AUTORQ` bit is cleared to prevent any further transactions being attempted. For cases where the size of the transfer is unknown, `USBRQPKTCOUNTn` should be left set to zero. `AUTORQ` then remains set until cleared by the reception of a short packet (that is, less than MaxP) such as may occur at the end of a bulk transfer.

If the device responds to a bulk or interrupt IN token with a NAK, the USB host controller keeps retrying the transaction until any NAK Limit that has been set has been reached. If the target device responds with a STALL, however, the USB host controller does not retry the transaction but interrupts the CPU with the `STALLED` bit in the `USBCSRL0` register set. If the target device does not respond to the IN token within the required time, or there was a CRC or bit-stuff error in the packet, the USB host controller retries the transaction. If after three attempts the target device has still not responded, the USB host controller clears the `REQPKT` bit and interrupts the CPU by setting the `ERROR` bit in the `USBCSRL0` register.

### 19.2.2.3 Out Transactions as a Host

OUT transactions are handled in a similar manner to the way in which IN transactions are handled when the USB controller is in Device mode. The `TXRDY` bit in the `USBTXCSSLn` register needs to be set as each packet is loaded into the transmit FIFO. Again, setting the `AUTOSET` bit in the `USBTXCSRHn` register automatically sets `TXRDY` when a maximum-sized packet has been loaded into the FIFO. Furthermore, `AUTOSET` can be used with a  $\mu$ DMA controller to perform complete bulk transfers without software intervention.

If the target device responds to the OUT token with a NAK, the USB host controller keeps retrying the transaction until the NAK Limit that has been set has been reached. However, if the target device responds with a STALL, the USB controller does not retry the transaction but interrupts the main processor by setting the `STALLED` bit in the `USBTXCSSLn` register. If the target device does not respond to the OUT token within the required time, or there was a CRC or bit-stuff error in the packet, the USB host controller retries the transaction. If after three attempts the target device has still not responded, the USB controller flushes the FIFO and interrupts the main processor by setting the `ERROR` bit in the `USBTXCSSLn` register.

### 19.2.2.4 Transaction Scheduling

Scheduling of transactions is handled automatically by the USB host controller. The host controller allows configuration of the endpoint communication scheduling based on the type of endpoint transaction. Interrupt transactions can be scheduled to occur in the range of every frame to every 255 frames in 1 frame increments. Bulk endpoints do not allow scheduling parameters, but do allow for a NAK timeout in the event an endpoint on a device is not responding. Isochronous endpoints can be scheduled from every frame to every  $2^{16}$  frames, in powers of 2.

The USB controller maintains a frame counter. If the target device is a full-speed device, the USB controller automatically sends an SOF packet at the start of each frame and increments the frame counter. If the target device is a low-speed device, a 'K' state is transmitted on the bus to act as a "keep-alive" to stop the low-speed device from going into Suspend mode.

After the SOF packet has been transmitted, the USB host controller cycles through all the configured endpoints looking for active transactions. An active transaction is defined as a receive endpoint for which the `REQPKT` bit is set or a transmit endpoint for which the `TXRDY` bit and/or the `FIFONE` bit is set.

An active isochronous or interrupt transaction starts only if it is found on the first transaction scheduler cycle of a frame and if the interval counter for that endpoint has counted down to zero. This ensures that only one interrupt or isochronous transaction occurs per endpoint every  $n$  frames, where  $n$  is the interval set via the **USBTXINTERVAL $n$**  or **USBRXINTERVAL $n$**  register for that endpoint.

An active bulk transaction starts immediately, provided there is sufficient time left in the frame to complete the transaction before the next SOF packet is due. If the transaction needs to be retried (for example, because a NAK was received or the target device did not respond), then the transaction is not retried until the transaction scheduler has first checked all the other endpoints for active transactions. This ensures that an endpoint that is sending a lot of NAKs does not block other transactions on the bus. The core also allows the user to specify a limit to the length of time for NAKs to be received from a target device before the endpoint times out.

#### 19.2.2.5 USB Hubs

The following setup requirements apply to the USB host controller only if it is used with a USB hub. When a full- or low-speed device is connected to the USB controller via a USB 2.0 hub, details of the hub address and the hub port also need to be recorded in the corresponding **USBRXHUBADDR $n$**  and **USBRXHUBPORT $n$**  or the **USBTXHUBADDR $n$**  and **USBTXHUBPORT $n$**  registers. In addition, the speed at which the device operates (full or low) needs to be recorded in the **USBTYPEn0** (endpoint 0), **USBTXTYPEn**, or **USBRXTYPEn** registers for each endpoint that is accessed by the device.

For hub communications, the settings in these registers record the current allocation of the endpoints to the attached USB devices. To maximize the number of devices supported, the USB host controller allows this allocation to be changed dynamically by simply updating the address and speed information recorded in these registers. Any changes in the allocation of endpoints to device functions need to be made following the completion of any on-going transactions on the endpoints affected.

#### 19.2.2.6 Babble

The USB host controller does not start a transaction until the bus has been inactive for at least the minimum inter-packet delay. It also does not start a transaction unless it can be finished before the end of the frame. If the bus is still active at the end of a frame, then the USB host controller assumes that the target device to which it is connected has malfunctioned and the USB controller suspends all transactions and generates a babble interrupt.

#### 19.2.2.7 Host Suspend

If the **SUSPEND** bit in the **USBPOWER** register is set, the USB host controller completes the current transaction then stops the transaction scheduler and frame counter. No further transactions are started and no SOF packets are generated.

To exit Suspend mode, the **RESUME** bit is set and the **SUSPEND** bit is cleared. While the **RESUME** bit is High, the USB host controller generates Resume signaling on the bus. After 20 ms, the **RESUME** bit should be cleared, at which point the frame counter and transaction scheduler start. The host supports the detection of a remote wake-up.

#### 19.2.2.8 USB Reset

If the **RESET** bit in the **USBPOWER** register is set, the USB host controller generates USB Reset signaling on the bus. The **RESET** bit should be set for at least 20 ms to ensure correct resetting of the target device. After the CPU has cleared the bit, the USB host controller starts its frame counter and transaction scheduler.

### 19.2.2.9 Connect/Disconnect

A session is started by setting the `SESSION` bit in the **USBDEVCTL** register. This enables the USB controller to wait for a device to be connected. When a device is detected, a connect interrupt is generated. The speed of the device that has been connected can be determined by reading the **USBDEVCTL** register where the `FSDEV` bit is High for a full-speed device and the `LSDEV` bit is High for a low-speed device. The USB controller should generate a reset to the device and then the USB host controller can begin device enumeration. If the device is disconnected while a session is in progress, a disconnect interrupt is generated.

## 19.2.3 OTG Mode

In order to conserve power, the USB On-The-Go (OTG) supplement allows VBus to only be powered up when required and to be turned off when the bus is not in use. VBus is always supplied by the A device on the bus. The USB OTG controller determines whether it is the A device or the B device by sampling the ID input from the PHY. This signal is pulled Low when an A-type plug is sensed (signifying that the USB OTG controller should act as the A device) but taken High when a B-type plug is sensed (signifying that the USB controller is a B device).

### 19.2.3.1 Starting a Session

When the USB OTG controller needs to start a session, the `SESSION` bit should be set in the **USBDEVCTL** register. The USB OTG controller then enables ID pin sensing. The ID input is either taken Low if an A-type connection is detected or High if a B-type connection is detected. The `DEV` bit in the **USBDEVCTL** register is also set to indicate whether the USB OTG controller has adopted the role of the A device or the B device.

If the USB OTG controller is the A device, then the USB OTG controller enters Host mode (the A device is always the default host), turns on VBus, and waits for VBus to go above the VBus Valid threshold, as indicated by the `VBUS` bit in the **USBDEVCTL** register going to 0x3. The USB OTG controller then waits for a peripheral to be connected. When a peripheral is detected, a Connect interrupt is signaled and either the `FSDEV` or `LSDEV` bit in the **USBDEVCTL** register is set, depending whether a full-speed or a low-speed peripheral is detected. The USB controller then issues a reset to the connected device. The `SESSION` bit in the **USBDEVCTL** register is cleared to end a session. The USB OTG controller will also automatically end the session if babble is detected.

If the USB OTG controller is the B device, then the USB OTG controller requests a session using the Session Request Protocol defined in the USB On-The-Go supplement, that is, it will first discharge VBus. Then when VBus has gone below the Session End threshold (`VBUS` bit in the **USBDEVCTL** register goes to 0x0) and the line state has been a single-ended zero for > 2 ms, the USB OTG controller pulses the data line, then pulses VBus. At the end of the session, the `SESSION` bit is cleared either by the USB OTG controller or by the application software. The USB OTG controller then causes the PHY to switch out the pull-up resistor on D+. This signals the A device to end the session.

### 19.2.3.2 Detecting Activity

When the other device of the OTG set-up wishes to start a session, it either raises VBus above the Session Valid threshold if it is the A device, or if it is the B device, it pulses the data line then pulses VBus. Depending on which of these actions happens, the USB controller can determine whether it is the A device or the B device in the current set-up and act accordingly. If VBus is raised above the Session Valid threshold, then the USB controller is the B device. The USB controller sets the `SESSION` bit in the **USBDEVCTL** register. When Reset signaling is detected on the bus, a Reset interrupt is signaled, which is interpreted as the start of a session.

The USB controller is in device mode at this point as the B device is the default mode. At the end of the session, the A device turns off the power to VBus. When VBus drops below the Session Valid threshold, the USB controller detects this and clears the `SESSION` bit to indicate that the session has ended. This causes a disconnect interrupt to be signaled. If data line and VBus pulsing is detected, then the USB controller is the A device. It generates a Session Request interrupt to indicate that the B device is requesting a session. The `SESSION` bit in the **USBDEVCTL** register should then be set to start a session.

### 19.2.3.3 Host Negotiation

When the USB controller is the A device, ID is Low, and it automatically enters Host mode when a session starts. When the USB controller is the B device, ID is High, and it automatically enters Device mode when a session starts. However, the CPU can request that the USB controller become the host by setting the `HOSTREQ` bit in the **USBDEVCTL** register. This bit can be set either at the same time as requesting a Session Start by setting the `SESSION` bit in the **USBDEVCTL** register, or at any time after a session has started. When the USB controller next enters Suspend mode, assuming the `HOSTREQ` bit remains set, it enters Host mode and begins host negotiation (as specified in the USB On-The-Go supplement) by causing the PHY to disconnect the pull-up resistor on the D+ line. This causes the A device to switch to Device mode and connect its own pull-up resistor. When the USB controller detects this, it generates a Connect interrupt. It also sets the `RESET` bit in the **USBPOWER** register to begin resetting the A device. The USB controller begins this reset sequence automatically to ensure that reset is started as required within 1 ms of the A device connecting its pull-up resistor. The main processor should wait at least 20 ms, then clear the `RESET` bit and enumerate the A device.

When the USB OTG controller B device has finished using the bus, it goes into Suspend mode by setting the `SUSPEND` bit in the **USBPOWER** register. The A device detects this and either terminates the session or reverts to Host mode. If the A device is USB OTG controller, it generates a Disconnect interrupt.

### 19.2.4 DMA Operation

The USB peripheral provides an interface connected to the  $\mu$ DMA controller. The DMA operation of the USB is enabled through the **USBTXCSRHn** and **USBRXCSRHn** registers, for the TX and RX channels respectively. When DMA operation is enabled, the USB asserts a DMA request on the enabled receive or transmit channel when the associated FIFO can transfer data. When either FIFO can transfer data, the burst request for that channel is asserted. The  $\mu$ DMA channel must be configured with an arbitration size that matches the size of the USB FIFO, and the size of the  $\mu$ DMA transfer must be restricted to whole multiples of the size of the USB FIFO. Both read and write transfers of the USB FIFOs using  $\mu$ DMA should be configured in this manner. For example, if the USB endpoint is configured with a FIFO size of 64 bytes, the  $\mu$ DMA channel must be configured with an arbitration size of 64. The  $\mu$ DMA channel can be used to transfer 64 bytes to or from the endpoint FIFO. If the number of bytes to transfer is less than 64, then a programmed I/O method must be used to copy the data to or from the FIFO.

If DMA is enabled, then the  $\mu$ DMA controller triggers an interrupt when a transfer is complete. The interrupt occurs on the USB interrupt vector. Therefore, if interrupts are used for USB operation and DMA is enabled, the USB interrupt handler must be designed to handle the  $\mu$ DMA completion interrupt.

Care must be taken when using a DMA to unload the receive FIFO as data is read from the receive FIFO in 4 byte chunks regardless of the `RxMaxP` bit in the **USBRXCSRHn** register. The `RXRDY` bit is cleared as follows.

**Table 19-1. Remainder (RxMaxP/4)**

Value	Description
0	RxMaxP = 64 bytes
1	RxMaxP = 61 bytes
2	RxMaxP = 62 bytes
3	RxMaxP = 63 bytes

**Table 19-2. Actual Bytes Read**

Value	Description
0	RxMaxP
1	RxMaxP+3
2	RxMaxP+2
3	RxMaxP+1

**Table 19-3. Packet Sizes That Will Clear RXRDY**

Value	Description
0	RxMaxP, RxMaxP-1, RxMaxP-2, RxMaxP-3
1	RxMaxP
2	RxMaxP, RxMaxP-1
3	RxMaxP, RxMaxP-1, RxMaxP-2

To enable DMA operation for the endpoint receive channel, the `DMAEN` bit of the **USBXCSRHn** register should be set. To enable DMA operation for the endpoint transmit channel, the `DMAEN` bit of the **USBTXCSRHn** register should be set.

See “Micro Direct Memory Access ( $\mu$ DMA)” on page 219 for more details about programming the  $\mu$ DMA controller.

## 19.3 Initialization and Configuration

To use the USB Controller, the peripheral clock must be enabled by via the **RCGC2** register. See page 175. In addition, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register in the System Control module. See page 175. To find out which GPIO port to enable, refer to Table 24-5 on page 937.

The initial configuration in all cases requires that the processor enable the USB controller and USB controller’s physical layer interface (PHY) before setting any registers. The next step is to enable the USB PLL so that the correct clocking is provided to the PHY. To ensure that voltage is not supplied to the bus incorrectly, the external power control signal, `USB0EPEN`, should be de-asserted on start up. This requires setting the `USB0EPEN` and `USB0PFLT` pins to be controlled by the USB controller and not have their default GPIO behavior.

The VBUS sense and ID pins (`USB0VBUS` and `USB0ID`) do not require any configuration as they are dedicated pins for the USB controller. In OTG mode, these pins directly connect to the USB connector’s VBUS and ID signals. In Host and Device modes, these pins must be tied to appropriate voltage levels. `USB0VBUS` must be tied to 5 V (4.75-5.25V). `USB0ID` must be tied Low for USB Host operation or tied High for USB Device Operation. These pins should not be used as GPIOs while using the USB controller as it may cause unexpected behavior in the controller.

### 19.3.1 Pin Configuration

When using the device controller portion of the USB controller in a system that also provides host functionality, the power to VBUS must be disabled to allow the external host controller to supply power. Usually, the `USB0EPEN` signal is used to control the external regulator and should be de-asserted to avoid having two devices driving the `USB0VBUS` power pin on the USB connector.

When the USB controller is acting as a host, it is in control of two signals that are attached to an external voltage supply that provides power to VBUS. The host controller uses the `USB0EPEN` signal to enable or disable power to the `USB0VBUS` pin on the USB connector. There is also an input pin, `USB0PFLT`, which provides feedback when there has been a power fault on VBUS. The `USB0PFLT` signal can be configured to either automatically de-assert the `USB0EPEN` signal to disable power, and/or it can generate an interrupt to the main processor to allow it to handle the power fault condition. The polarity and actions related to both `USB0EPEN` and `USB0PFLT` are fully configurable in the USB controller. The controller also provides interrupts on device insertion and removal to allow the host controller code to respond to these external events.

### 19.3.2 Endpoint Configuration

In order to start communication on host or device mode, the endpoint registers must first be configured. In Host mode, this provides a connection between an endpoint register and an endpoint on a device. In Device mode, this provides the setup for a given endpoint before enumerating to the host controller.

In both cases, the endpoint 0 configuration is limited as this is a fixed function, fixed FIFO size endpoint. In Device and Host modes, the endpoint requires little setup but does require a software-based state machine to progress through the setup, data, and status phases of a standard control transaction. In Device mode, the configuration of the remaining endpoints is done once before enumerating and then only changed if an alternate configuration is selected by the host controller. In Host mode, the endpoints must be configured to operate as control, bulk, interrupt or isochronous mode. Once the type of endpoint is configured, a FIFO area must be assigned to each endpoint. In the case of bulk, control and interrupt endpoints, each has a maximum of 64 bytes per transaction. Isochronous endpoints can have packets with up to 1023 bytes per packet. In either mode, the maximum packet size for the given endpoint must be set prior to sending or receiving data.

Configuring each endpoint's FIFO involves reserving a portion of the overall USB FIFO RAM to each endpoint. The total FIFO RAM available is 4 Kbytes with the first 64 bytes in use by endpoint 0. The endpoint's FIFO does not have to be the same size as the maximum packet size in all cases as the controller can automatically split for bulk transactions if the FIFO is larger than the maximum packet size. The FIFO can also be configured as a double-buffered FIFO so that interrupts occur at the end of each packet and allow filling the other half of the FIFO.

If operating as a device, the USB device controllers' soft connect should be enabled when the device is ready to start communications. This indicates to the host controller that the device is ready to start the enumeration process. If operating as a host controller, the device soft connect should be disabled and power should be provided to VBUS via the `USB0EPEN` signal.

## 19.4 Register Map

Table 19-4 on page 731 lists the registers. All addresses given are relative to the USB base address of 0x4005.0000. Note that the USB controller clock must be enabled before the registers can be programmed (see page 175).

Table 19-4. Universal Serial Bus (USB) Controller Register Map

Offset	Name	Type	Reset	Description	See page
0x000	USBFADDR	R/W	0x00	USB Device Functional Address	735
0x001	USBPOWER	R/W	0x20	USB Power	736
0x002	USBTXIS	RO	0x0000	USB Transmit Interrupt Status	738
0x004	USBRXIS	RO	0x0000	USB Receive Interrupt Status	739
0x006	USBTXIE	R/W	0x000F	USB Transmit Interrupt Enable	740
0x008	USBRXIE	R/W	0x000E	USB Receive Interrupt Enable	741
0x00A	USBIS	RO	0x00	USB General Interrupt Status	742
0x00B	USBIE	R/W	0x06	USB Interrupt Enable	744
0x00C	USBFFRAME	RO	0x0000	USB Frame Value	746
0x00E	USBEPIDX	R/W	0x00	USB Endpoint Index	747
0x00F	USBTTEST	R/W	0x00	USB Test Mode	748
0x020	USBFIFO0	R/W	0x0000.0000	USB FIFO Endpoint 0	750
0x024	USBFIFO1	R/W	0x0000.0000	USB FIFO Endpoint 1	750
0x028	USBFIFO2	R/W	0x0000.0000	USB FIFO Endpoint 2	750
0x02C	USBFIFO3	R/W	0x0000.0000	USB FIFO Endpoint 3	750
0x060	USBDEVCTL	R/W	0x80	USB Device Control	751
0x062	USBTXFIFOSZ	R/W	0x00	USB Transmit Dynamic FIFO Sizing	754
0x063	USBRXFIFOSZ	R/W	0x00	USB Receive Dynamic FIFO Sizing	754
0x064	USBTXFIFOADD	R/W	0x0000	USB Transmit FIFO Start Address	755
0x066	USBRXFIFOADD	R/W	0x0000	USB Receive FIFO Start Address	755
0x07A	USBCONTIM	R/W	0x5C	USB Connect Timing	756
0x07B	USBVPLEN	R/W	0x3C	USB OTG VBus Pulse Timing	757
0x07D	USBFSEOF	R/W	0x77	USB Full-Speed Last Transaction to End of Frame Timing	758
0x07E	USBLSEOF	R/W	0x72	USB Low-Speed Last Transaction to End of Frame Timing	759
0x080	USBTXFUNCADDR0	R/W	0x00	USB Transmit Functional Address Endpoint 0	760
0x082	USBTXHUBADDR0	R/W	0x00	USB Transmit Hub Address Endpoint 0	761
0x083	USBTXHUBPORT0	R/W	0x00	USB Transmit Hub Port Endpoint 0	762
0x088	USBTXFUNCADDR1	R/W	0x00	USB Transmit Functional Address Endpoint 1	760
0x08A	USBTXHUBADDR1	R/W	0x00	USB Transmit Hub Address Endpoint 1	761
0x08B	USBTXHUBPORT1	R/W	0x00	USB Transmit Hub Port Endpoint 1	762
0x08C	USBRXFUNCADDR1	R/W	0x00	USB Receive Functional Address Endpoint 1	763

Offset	Name	Type	Reset	Description	See page
0x08E	USBRXHUBADDR1	R/W	0x00	USB Receive Hub Address Endpoint 1	764
0x08F	USBRXHUBPORT1	R/W	0x00	USB Receive Hub Port Endpoint 1	765
0x090	USBTXFUNCADDR2	R/W	0x00	USB Transmit Functional Address Endpoint 2	760
0x092	USBTXHUBADDR2	R/W	0x00	USB Transmit Hub Address Endpoint 2	761
0x093	USBTXHUBPORT2	R/W	0x00	USB Transmit Hub Port Endpoint 2	762
0x094	USBRXFUNCADDR2	R/W	0x00	USB Receive Functional Address Endpoint 2	763
0x096	USBRXHUBADDR2	R/W	0x00	USB Receive Hub Address Endpoint 2	764
0x097	USBRXHUBPORT2	R/W	0x00	USB Receive Hub Port Endpoint 2	765
0x098	USBTXFUNCADDR3	R/W	0x00	USB Transmit Functional Address Endpoint 3	760
0x09A	USBTXHUBADDR3	R/W	0x00	USB Transmit Hub Address Endpoint 3	761
0x09B	USBTXHUBPORT3	R/W	0x00	USB Transmit Hub Port Endpoint 3	762
0x09C	USBRXFUNCADDR3	R/W	0x00	USB Receive Functional Address Endpoint 3	763
0x09E	USBRXHUBADDR3	R/W	0x00	USB Receive Hub Address Endpoint 3	764
0x09F	USBRXHUBPORT3	R/W	0x00	USB Receive Hub Port Endpoint 3	765
0x102	USBCSRL0	W1C	0x00	USB Control and Status Endpoint 0 Low	767
0x103	USBCSRH0	W1C	0x00	USB Control and Status Endpoint 0 High	770
0x108	USBCOUNT0	RO	0x00	USB Receive Byte Count Endpoint 0	772
0x10A	USBTTYPE0	R/W	0x00	USB Type Endpoint 0	773
0x10B	USBNAKLMT	R/W	0x00	USB NAK Limit	774
0x110	USBTXMAXP1	R/W	0x0000	USB Maximum Transmit Data Endpoint 1	766
0x112	USBTXCSRL1	R/W	0x00	USB Transmit Control and Status Endpoint 1 Low	775
0x113	USBTXCSRH1	R/W	0x00	USB Transmit Control and Status Endpoint 1 High	778
0x114	USBRXMAXP1	R/W	0x0000	USB Maximum Receive Data Endpoint 1	781
0x116	USBRXCSRL1	R/W	0x00	USB Receive Control and Status Endpoint 1 Low	782
0x117	USBRXCSRH1	R/W	0x00	USB Receive Control and Status Endpoint 1 High	785
0x118	USBRXCOUNT1	RO	0x0000	USB Receive Byte Count Endpoint 1	788
0x11A	USBTXTYPE1	R/W	0x00	USB Host Transmit Configure Type Endpoint 1	789
0x11B	USBTXINTERVAL1	R/W	0x00	USB Host Transmit Interval Endpoint 1	791
0x11C	USBRXTYPE1	R/W	0x00	USB Host Configure Receive Type Endpoint 1	792
0x11D	USBRXINTERVAL1	R/W	0x00	USB Host Receive Polling Interval Endpoint 1	794
0x120	USBTXMAXP2	R/W	0x0000	USB Maximum Transmit Data Endpoint 2	766
0x122	USBTXCSRL2	R/W	0x00	USB Transmit Control and Status Endpoint 2 Low	775
0x123	USBTXCSRH2	R/W	0x00	USB Transmit Control and Status Endpoint 2 High	778



Offset	Name	Type	Reset	Description	See page
0x124	USBRXMAXP2	R/W	0x0000	USB Maximum Receive Data Endpoint 2	781
0x126	USBRXCSSL2	R/W	0x00	USB Receive Control and Status Endpoint 2 Low	782
0x127	USBRXCSSL2H	R/W	0x00	USB Receive Control and Status Endpoint 2 High	785
0x128	USBRXCOUNT2	RO	0x0000	USB Receive Byte Count Endpoint 2	788
0x12A	USBTXTYPE2	R/W	0x00	USB Host Transmit Configure Type Endpoint 2	789
0x12B	USBTXINTERVAL2	R/W	0x00	USB Host Transmit Interval Endpoint 2	791
0x12C	USBRXTYPE2	R/W	0x00	USB Host Configure Receive Type Endpoint 2	792
0x12D	USBRXINTERVAL2	R/W	0x00	USB Host Receive Polling Interval Endpoint 2	794
0x130	USBTXMAXP3	R/W	0x0000	USB Maximum Transmit Data Endpoint 3	766
0x132	USBTXCSSL3	R/W	0x00	USB Transmit Control and Status Endpoint 3 Low	775
0x133	USBTXCSSL3H	R/W	0x00	USB Transmit Control and Status Endpoint 3 High	778
0x134	USBRXMAXP3	R/W	0x0000	USB Maximum Receive Data Endpoint 3	781
0x136	USBRXCSSL3	R/W	0x00	USB Receive Control and Status Endpoint 3 Low	782
0x137	USBRXCSSL3H	R/W	0x00	USB Receive Control and Status Endpoint 3 High	785
0x138	USBRXCOUNT3	RO	0x0000	USB Receive Byte Count Endpoint 3	788
0x13A	USBTXTYPE3	R/W	0x00	USB Host Transmit Configure Type Endpoint 3	789
0x13B	USBTXINTERVAL3	R/W	0x00	USB Host Transmit Interval Endpoint 3	791
0x13C	USBRXTYPE3	R/W	0x00	USB Host Configure Receive Type Endpoint 3	792
0x13D	USBRXINTERVAL3	R/W	0x00	USB Host Receive Polling Interval Endpoint 3	794
0x304	USBRQPKTCOUNT1	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 1	795
0x308	USBRQPKTCOUNT2	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 2	795
0x30C	USBRQPKTCOUNT3	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 3	795
0x340	USBRXDPKTBUFDIS	R/W	0x0000	USB Receive Double Packet Buffer Disable	796
0x342	USBTXDPKTBUFDIS	R/W	0x0000	USB Transmit Double Packet Buffer Disable	797
0x400	USBEPCC	R/W	0x0000.0000	USB External Power Control	798
0x404	USBEPCCRIS	RO	0x0000.0000	USB External Power Control Raw Interrupt Status	801
0x408	USBEPCCIM	R/W	0x0000.0000	USB External Power Control Interrupt Mask	802
0x40C	USBEPCCISC	R/W	0x0000.0000	USB External Power Control Interrupt Status and Clear	803
0x410	USBDRRIS	RO	0x0000.0000	USB Device Resume Raw Interrupt Status	804
0x414	USBDRRIM	R/W	0x0000.0000	USB Device Resume Interrupt Mask	805
0x418	USBDRRISC	W1C	0x0000.0000	USB Device Resume Interrupt Status and Clear	806

Offset	Name	Type	Reset	Description	See page
0x430	USBVDC	R/W	0x0000.0000	USB VBUS Droop Control	807
0x434	USBVDCRIS	RO	0x0000.0000	USB VBUS Droop Control Raw Interrupt Status	808
0x438	USBVDCIM	R/W	0x0000.0000	USB VBUS Droop Control Interrupt Mask	809
0x43C	USBVDCISC	R/W	0x0000.0000	USB VBUS Droop Control Interrupt Status and Clear	810
0x444	USBIDVRIS	RO	0x0000.0000	USB ID Valid Detect Raw Interrupt Status	811
0x448	USBIDVIM	R/W	0x0000.0000	USB ID Valid Detect Interrupt Mask	812
0x44C	USBIDVISC	R/W1C	0x0000.0000	USB ID Valid Detect Interrupt Status and Clear	813
0x450	USBEPS	R/W	0x0000.0321	USB End-Point Select	814

## 19.5 Register Descriptions

The LM3S5791 USB controller is configured to the communication mode specified in the `USB0` bit field in the **DC6** register (page 143):

- On-The-Go (OTG) (`USB0` set to 0x3)

## Register 1: USB Device Functional Address (USBFADDR), offset 0x000

### Device

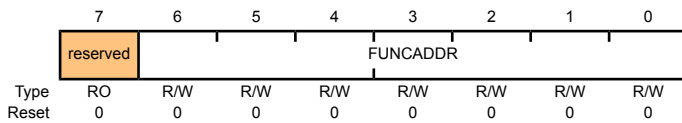
**USBFADDR** is an 8-bit register that should be written with the 7-bit address of the device part of the transaction.

When the USB controller is being used in Device mode (`HOST` bit in **USBDEVCTL** register is 0), this register should be written with the address received through a `SET_ADDRESS` command, which is then used for decoding the function address in subsequent token packets.

**Important:** See the section called “Setting the Device Address” on page 722 for special considerations when writing this register.

### USB Device Functional Address (USBFADDR)

Base 0x4005.0000  
Offset 0x000  
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
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.
6:0	FUNCADDR	R/W	0x00	Function Address Function Address of Device as received through <code>SET_ADDRESS</code> .

## Register 2: USB Power (USBPOWER), offset 0x001

Host

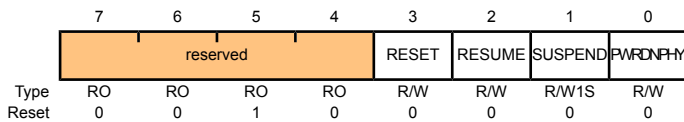
USBPOWER is an 8-bit register that is used for controlling Suspend and Resume signaling, and some basic operational aspects of the USB controller.

Device

### Host Mode

#### USB Power (USBPOWER)

Base 0x4005.0000  
Offset 0x001  
Type R/W, reset 0x20

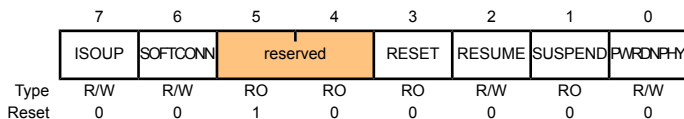


Bit/Field	Name	Type	Reset	Description
7:4	reserved	RO	0x02	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	RESET	R/W	0	Reset  This bit is set to enable Reset signaling on the bus and cleared to end Reset signaling on the bus.
2	RESUME	R/W	0	Resume Signaling  Set by the CPU to generate Resume signaling when the device is in Suspend mode. The CPU should clear this bit after 20 ms.
1	SUSPEND	R/W1S	0	Suspend Mode  This bit is written to 1 by the CPU to enter Suspend mode. Writing a 0 does nothing.
0	PWRDNPHY	R/W	0	Power Down PHY  Set by the CPU to power down the internal USB PHY.

### Device Mode

#### USB Power (USBPOWER)

Base 0x4005.0000  
Offset 0x001  
Type R/W, reset 0x20



Bit/Field	Name	Type	Reset	Description
7	ISOUP	R/W	0	<p>ISO Update</p> <p>When set by the CPU, the USB controller waits for an SOF token from the time <code>TXRDY</code> is set before sending the packet. If an IN token is received before an SOF token, then a zero-length data packet is sent.</p> <p><b>Note:</b> Only valid for isochronous transfers.</p>
6	SOFTCONN	R/W	0	<p>Soft Connect/Disconnect</p> <p>The USB D+/D- lines are enabled when this bit is set by the CPU, and tri-stated when this bit is cleared by the CPU.</p>
5:4	reserved	RO	0x2	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>
3	RESET	RO	0	<p>Reset</p> <p>This bit is set when Reset signaling is present on the bus.</p>
2	RESUME	R/W	0	<p>Resume Signaling</p> <p>Set by the CPU to generate Resume signaling when the device is in Suspend mode. The CPU should clear this bit after 10 ms (a maximum of 15 ms) to end Resume signaling.</p>
1	SUSPEND	RO	0	<p>Suspend Mode</p> <p>This bit is set on entry into Suspend mode. It is cleared when the CPU reads the interrupt register or sets the <code>RESUME</code> bit above.</p>
0	PWRDNPHY	R/W	0	<p>Power Down PHY</p> <p>Set by the CPU to power down the internal USB PHY.</p>

### Register 3: USB Transmit Interrupt Status (USBTXIS), offset 0x002

Host

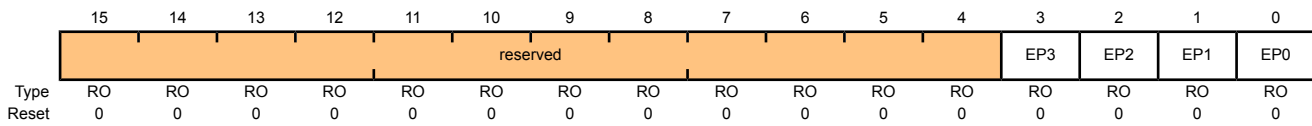
Device

**USBTXIS** is a 16-bit read-only register that indicates which interrupts are currently active for endpoint 0 and the transmit endpoints 1–3. The meaning of the  $EP_n$  bits in this register are based on the mode of the device. For the  $EP_1$ ,  $EP_2$  and  $EP_3$  bits, these bits always indicate that the USB controller is sending data; however, in Host mode, these are the three configurable OUT endpoints; while in device mode, these are the three configurable IN endpoints. The  $EP_0$  bit is special in Host and Device modes and indicates that either a control IN or control OUT endpoint has generated an interrupt.

**Note:** Bits relating to endpoints that have not been configured always return 0. Note also that all active interrupts are cleared when this register is read.

#### USB Transmit Interrupt Status (USBTXIS)

Base 0x4005.0000  
 Offset 0x002  
 Type RO, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:4	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	EP3	RO	0	TX Endpoint 3 Interrupt
2	EP2	RO	0	TX Endpoint 2 Interrupt
1	EP1	RO	0	TX Endpoint 1 Interrupt
0	EP0	RO	0	TX and RX Endpoint 0 Interrupt

**Register 4: USB Receive Interrupt Status (USBRIX), offset 0x004****Host**

**USBRIX** is a 16-bit read-only register that indicates which of the interrupts for receive endpoints 1–3 are currently active.

**Device**

**Note:** Bits relating to endpoints that have not been configured always return 0. Note also that all active interrupts are cleared when this register is read.

## USB Receive Interrupt Status (USBRIX)

Base 0x4005.0000

Offset 0x004

Type RO, reset 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												EP3	EP2	EP1	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
15:4	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	EP3	RO	0	RX Endpoint 3 Interrupt
2	EP2	RO	0	RX Endpoint 2 Interrupt
1	EP1	RO	0	RX Endpoint 1 Interrupt
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: USB Transmit Interrupt Enable (USBTXIE), offset 0x006

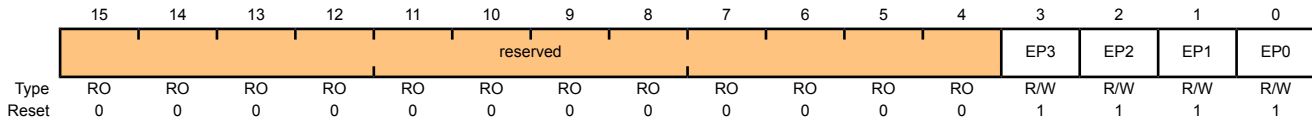
Host

Device

**USBTXIE** is a 16-bit register that provides interrupt enable bits for the interrupts in **USBTXIS**. When a bit in **USBTXIE** is set to 1, the USB interrupt to the processor is asserted when the corresponding interrupt bit in the **USBTXIS** register is set. When a bit is cleared to 0, the interrupt in **USBTXIS** is still set but the USB interrupt to the processor is not asserted. On reset, the bits corresponding to endpoint 0 and transmit endpoints 1-3 are set to 1, while the remaining bits are set to 0.

#### USB Transmit Interrupt Enable (USBTXIE)

Base 0x4005.0000  
 Offset 0x006  
 Type R/W, reset 0x000F



Bit/Field	Name	Type	Reset	Description
15:4	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	EP3	R/W	1	TX Endpoint 3 Interrupt Enable
2	EP2	R/W	1	TX Endpoint 2 Interrupt Enable
1	EP1	R/W	1	TX Endpoint 1 Interrupt Enable
0	EP0	R/W	1	TX and RX Endpoint 0 Interrupt Enable



## Register 6: USB Receive Interrupt Enable (USBRXIE), offset 0x008

Host

**USBRXIE** is a 16-bit register that provides interrupt enable bits for the interrupts in **USBRXIS**. When a bit in **USBRXIE** is set to 1, the USB interrupt to the processor is asserted when the corresponding interrupt bit in the **USBRXIS** register is set. When a bit is cleared to 0, the interrupt in **USBRXIS** is still set but the USB interrupt to the processor is not asserted. On reset, the bits corresponding to receive endpoints 1-3 are set to 1, while the remaining bits are set to 0.

Device

### USB Receive Interrupt Enable (USBRXIE)

Base 0x4005.0000

Offset 0x008

Type R/W, reset 0x000E

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved												EP3	EP2	EP1	reserved	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

Bit/Field	Name	Type	Reset	Description
15:4	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	EP3	R/W	1	RX Endpoint 3 Interrupt Enable
2	EP2	R/W	1	RX Endpoint 2 Interrupt Enable
1	EP1	R/W	1	RX Endpoint 1 Interrupt Enable
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: USB General Interrupt Status (USBIS), offset 0x00A

OTG

**USBIS** is an 8-bit read-only register that indicates which USB interrupts are currently active. All active interrupts are cleared when this register is read.

Host

Device

### Host Mode

#### USB General Interrupt Status (USBIS)

Base 0x4005.0000

Offset 0x00A

Type RO, reset 0x00

	7	6	5	4	3	2	1	0
	VBUSERR	SESREQ	DISCON	CONN	SOF	BABBLE	RESUME	reserved
Type	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	VBUSERR	RO	0	<p>VBus Error</p> <p>Set when VBus drops below the VBus Valid threshold during a session.</p> <p><b>Note:</b> Only valid when the USB controller is an OTG A device.</p>
6	SESREQ	RO	0	<p>Session Request</p> <p>Set when Session Request signaling has been detected.</p> <p><b>Note:</b> Only valid when the USB controller is an OTG A device.</p>
5	DISCON	RO	0	<p>Session Disconnect</p> <p>Set when a device disconnect is detected.</p>
4	CONN	RO	0	<p>Session Connect</p> <p>Set when a device connection is detected.</p>
3	SOF	RO	0	<p>Start of Frame</p> <p>Set when a new frame starts.</p>
2	BABBLE	RO	0	<p>Babble Detected</p> <p>Set when babble is detected. Only active after first SOF has been sent.</p>
1	RESUME	RO	0	<p>Resume Signal Detected</p> <p>Set when Resume signaling is detected on the bus while the USB controller is in Suspend mode.</p> <p>This can only be used if the USB's system clock is enabled. If the user disables the clock programming, the <b>USBDRCRIS</b>, <b>USBDRCIM</b>, and <b>USBISC</b> registers should be used.</p>

Bit/Field	Name	Type	Reset	Description
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.

## Device Mode

### USB General Interrupt Status (USBIS)

Base 0x4005.0000

Offset 0x00A

Type RO, reset 0x00

	7	6	5	4	3	2	1	0
	VBUSERR	SESREQ	DISCON	reserved	SOF	RESET	RESUME	SUSPEND
Type	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	VBUSERR	RO	0	<p>VBus Error</p> <p>Set when VBus drops below the VBus Valid threshold during a session.</p> <p><b>Note:</b> Only valid when the USB controller is an OTG A device.</p>
6	SESREQ	RO	0	<p>Session Request</p> <p>Set when Session Request signaling has been detected.</p> <p><b>Note:</b> Only valid when the USB controller is an OTG A device.</p>
5	DISCON	RO	0	<p>Session Disconnect</p> <p>Set when a session ends. Valid at all transaction speeds.</p>
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	SOF	RO	0	<p>Start of Frame</p> <p>Set when a new frame starts.</p>
2	RESET	RO	0	<p>Reset Signal Detected</p> <p>Set when Reset signaling is detected on the bus.</p>
1	RESUME	RO	0	<p>Resume Signal Detected</p> <p>Set when Resume signaling is detected on the bus while the USB controller is in Suspend mode.</p> <p>This can only be used if the USB's system clock is enabled. If the user disables the clock programming, the <b>USBDRCRIS</b>, <b>USBDRCIM</b>, and <b>USBISC</b> registers should be used.</p>
0	SUSPEND	RO	0	<p>Suspend Signal Detected</p> <p>Set when Suspend signaling is detected on the bus.</p>

## Register 8: USB Interrupt Enable (USBIE), offset 0x00B

Host

**USBIE** is an 8-bit register that provides interrupt enable bits for each of the interrupts in **USBIS**. By default, interrupt 1 and 2 are enabled.

Device

### Host Mode

#### USB Interrupt Enable (USBIE)

Base 0x4005.0000  
 Offset 0x00B  
 Type R/W, reset 0x06

	7	6	5	4	3	2	1	0
	VBUSERR	SESREQ	DISCON	CONN	SOF	RESET	RESUME	SUSPND
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

Bit/Field	Name	Type	Reset	Description
7	VBUSERR	R/W	0	Enable VBUS Error Interrupt Set by CPU to enable <code>VBUSERR</code> in <b>USBIS</b> .
6	SESREQ	R/W	0	Enable Session Request Set by CPU to enable <code>SESREQ</code> in <b>USBIS</b> .
5	DISCON	R/W	0	Enable Disconnect Interrupt Set by CPU to enable <code>DISCON</code> in <b>USBIS</b> .
4	CONN	R/W	0	Enable Connect Interrupt Set by CPU to enable <code>CONN</code> in <b>USBIS</b> .
3	SOF	R/W	0	Enable Start-of-Frame Interrupt Set by CPU to enable <code>SOF</code> in <b>USBIS</b> .
2	RESET	R/W	1	Enable Reset Interrupt Set by CPU to enable <code>RESET</code> in <b>USBIS</b> .
1	RESUME	R/W	1	Enable Resume Interrupt Set by CPU to enable <code>RESUME</code> in <b>USBIS</b> .
0	SUSPND	R/W	0	Enable Suspend Interrupt Set by CPU to enable <code>SUSPEND</code> in <b>USBIS</b> .

## Device Mode

### USB Interrupt Enable (USBIE)

Base 0x4005.0000  
 Offset 0x00B  
 Type R/W, reset 0x06

	7	6	5	4	3	2	1	0
	VBUSERR	SESREQ	DISCON	CONN	SOF	BABBLE	RESUME	SUSPND
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	1	1	0

Bit/Field	Name	Type	Reset	Description
7	VBUSERR	R/W	0	Enable VBUS Error Interrupt Set by CPU to enable <code>VBUSERR</code> in <b>USBIS</b> .
6	SESREQ	R/W	0	Enable Session Request Interrupt Set by CPU to enable <code>SESREQ</code> in <b>USBIS</b> .
5	DISCON	R/W	0	Enable Disconnect Interrupt Set by CPU to enable <code>DISCON</code> in <b>USBIS</b> .
4	CONN	R/W	0	Enable Connect Interrupt Set by CPU to enable <code>CONN</code> in <b>USBIS</b> .
3	SOF	R/W	0	Enable Start-of-Frame Interrupt Set by CPU to enable <code>SOF</code> in <b>USBIS</b> .
2	BABBLE	R/W	1	Enable Babble Interrupt Set by CPU to enable <code>BABBLE</code> in <b>USBIS</b> .
1	RESUME	R/W	1	Enable Resume Interrupt Set by CPU to enable <code>RESUME</code> in <b>USBIS</b> .
0	SUSPND	R/W	0	Enable Suspend Interrupt Set by CPU to enable <code>SUSPEND</code> in <b>USBIS</b> .

**Register 9: USB Frame Value (USBFRAME), offset 0x00C**

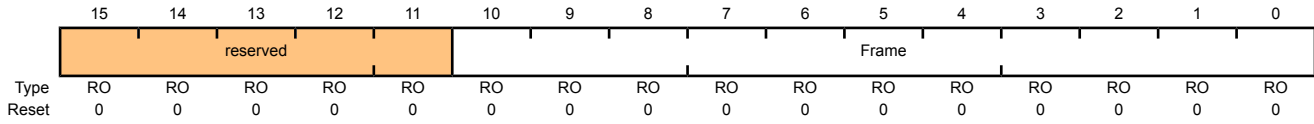
Host

**USBFRAME** is a 16-bit read-only register that holds the last received frame number.

Device

USB Frame Value (USBFRAME)

Base 0x4005.0000  
 Offset 0x00C  
 Type RO, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:11	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.
10:0	Frame	RO	0x00	Frame Number

**Register 10: USB Endpoint Index (USBEPIDX), offset 0x00E****Host**

Each endpoint's buffer can be accessed by configuring a FIFO size and starting address. The **USBEPIDX** 16-bit register is used with the **USBTXFIFOSZ**, **USBRXFIFOSZ**, **USBTXFIFOADD**, and **USBRXFIFOADD** registers.

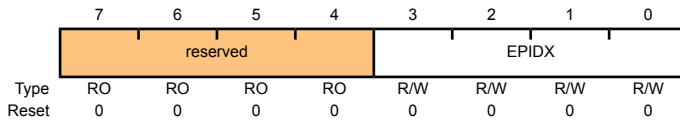
**Device**

USB Endpoint Index (USBEPIDX)

Base 0x4005.0000

Offset 0x00E

Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	EPIDX	R/W	0x00	Endpoint Index  This sets which endpoint is accessed when reading or writing to one of the USB controller's indexed registers.

### Register 11: USB Test Mode (USBTEST), offset 0x00F

Host

**USBTESTMODE** is an 8-bit register that is primarily used to put the USB controller into one of the four test modes for operation described in the *USB 2.0 specification*, in response to a SET FEATURE: USBTESTMODE command. It is not used in normal operation.

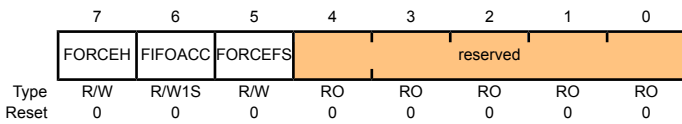
Device

**Note:** Only one of these bits should be set at any time.

#### Host Mode

##### USB Test Mode (USBTEST)

Base 0x4005.0000  
 Offset 0x00F  
 Type R/W, reset 0x00

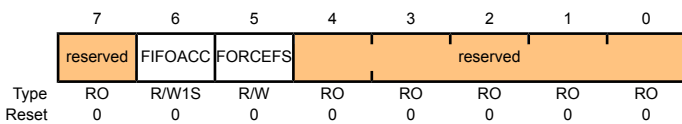


Bit/Field	Name	Type	Reset	Description
7	FORCEH	R/W	0	<p>Force Host Mode</p> <p>The CPU sets this bit to instruct the core to enter Host mode when the Session bit is set, regardless of whether it is connected to any peripheral. The state of the USB<sup>D+</sup> and USB<sup>D-</sup> are ignored. The core then remains in Host mode until the SESSION bit is cleared, even if a device is disconnected, and if the FORCEH bit remains set, re-enters Host mode the next time the SESSION bit is set.</p> <p>While in this mode, status of the bus connection may be read from the DEV bit of the <b>USBDEVCTL</b> register. The operating speed is determined from the FORCEFS bit.</p>
6	FIFOACC	R/W1S	0	<p>FIFO Access</p> <p>The CPU sets this bit to transfer the packet in the endpoint 0 transmit FIFO to the endpoint 0 receive FIFO. It is cleared automatically.</p>
5	FORCEFS	R/W	0	<p>Force Full-Speed Mode</p> <p>The CPU sets this bit to force the USB controller into Full-Speed mode when it receives a USB reset. When 0, the USB controller operates at Low Speed.</p>
4:0	reserved	RO	0x00	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>

#### Device Mode

##### USB Test Mode (USBTEST)

Base 0x4005.0000  
 Offset 0x00F  
 Type R/W, reset 0x00





---

Bit/Field	Name	Type	Reset	Description
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	FIFOACC	R/W1S	0	FIFO Access The CPU sets this bit to transfer the packet in the endpoint 0 transmit FIFO to the endpoint 0 receive FIFO. It is cleared automatically.
5	FORCEFS	R/W	0	Force Full Speed The CPU sets this bit to force the USB controller into Full-Speed mode when it receives a USB reset. When 0, the USB controller operates at Low Speed.
4:0	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 12: USB FIFO Endpoint 0 (USBFIFO0), offset 0x020**

**Register 13: USB FIFO Endpoint 1 (USBFIFO1), offset 0x024**

**Register 14: USB FIFO Endpoint 2 (USBFIFO2), offset 0x028**

**Register 15: USB FIFO Endpoint 3 (USBFIFO3), offset 0x02C**

Host

These 32-bit registers provide an address for CPU access to the FIFOs for each endpoint. Writing to these addresses loads data into the Transmit FIFO for the corresponding endpoint. Reading from these addresses unloads data from the Receive FIFO for the corresponding endpoint.

Device

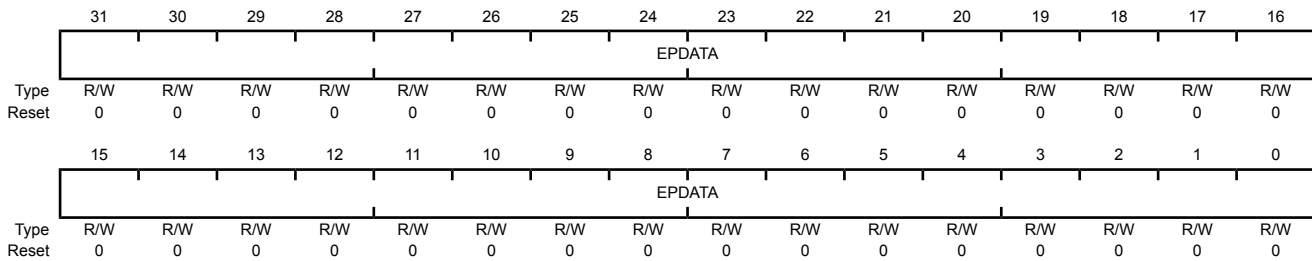
Transfers to and from FIFOs may be 8-bit, 16-bit or 32-bit as required, and any combination of access is allowed provided the data accessed is contiguous. All transfers associated with one packet must be of the same width so that the data is consistently byte-, word- or double-word-aligned. However, the last transfer may contain fewer bytes than the previous transfers in order to complete an odd-byte or odd-word transfer.

Depending on the size of the FIFO and the expected maximum packet size, the FIFOs support either single-packet or double-packet buffering. Burst writing of multiple packets is not supported as flags need to be set after each packet is written.

Following a STALL response or a transmit error on endpoint 1–3, the associated FIFO is completely flushed.

USB FIFO Endpoint 0 (USBFIFO0)

Base 0x4005.0000  
 Offset 0x020  
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	EPDATA	R/W	0x00	Endpoint Data

Writing to this register loads the data into the Transmit FIFO and reading unloads data from the Receive FIFO.

**Register 16: USB Device Control (USBDEVCTL), offset 0x060****OTG**

**USBDEVCTL** is an 8-bit register used for controlling and monitoring the USB VBus line. If the PHY is suspended, no PHY clock is received and the VBus is not sampled.

**Host**

**USBDEVCTL** provides the status information for the current operating mode (host or device) of the USB controller. If the USB controller is in host mode, this register also indicates if a full- or low-speed device has been connected.

**Device****Host Mode**

## USB Device Control (USBDEVCTL)

Base 0x4005.0000

Offset 0x060

Type R/W, reset 0x80

	7	6	5	4	3	2	1	0
	DEV	FSDEV	LSDEV	VBUS		HOST	HOSTREQ	SESSION
Type	RO	RO	RO	RO	RO	RO	R/W	R/W
Reset	1	0	0	0	0	0	0	0

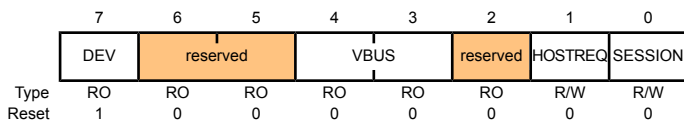
Bit/Field	Name	Type	Reset	Description						
7	DEV	RO	1	<p>Device Mode</p> <p>This read-only bit indicates whether the USB controller is operating as the OTG A device or the OTG B device.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>A device</td> </tr> <tr> <td>1</td> <td>B device</td> </tr> </tbody> </table> <p><b>Note:</b> This value is only valid while a session is in progress.</p>	Value	Description	0	A device	1	B device
Value	Description									
0	A device									
1	B device									
6	FSDEV	RO	0	<p>Full-Speed Device Detected</p> <p>This read-only bit is set when a full-speed device has been detected on the port.</p>						
5	LSDEV	RO	0	<p>Low-Speed Device Detected</p> <p>This read-only bit is set when a low-speed device has been detected on the port.</p>						

Bit/Field	Name	Type	Reset	Description										
4:3	VBUS	RO	0x00	<p>VBus Level</p> <p>These read-only bits encode the current VBus level as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Below SessionEnd VBUS is detected as under 0.5 V.</td> </tr> <tr> <td>0x1</td> <td>Above SessionEnd, below AValid VBUS is detected as above 0.5 V and under 1.5 V.</td> </tr> <tr> <td>0x2</td> <td>Above AValid, below VBusValid VBUS is detected as above 1.5 V and below 4.5 V.</td> </tr> <tr> <td>0x3</td> <td>Above VBusValid VBUS is detected as above 4.5 V.</td> </tr> </tbody> </table>	Value	Description	0x0	Below SessionEnd VBUS is detected as under 0.5 V.	0x1	Above SessionEnd, below AValid VBUS is detected as above 0.5 V and under 1.5 V.	0x2	Above AValid, below VBusValid VBUS is detected as above 1.5 V and below 4.5 V.	0x3	Above VBusValid VBUS is detected as above 4.5 V.
Value	Description													
0x0	Below SessionEnd VBUS is detected as under 0.5 V.													
0x1	Above SessionEnd, below AValid VBUS is detected as above 0.5 V and under 1.5 V.													
0x2	Above AValid, below VBusValid VBUS is detected as above 1.5 V and below 4.5 V.													
0x3	Above VBusValid VBUS is detected as above 4.5 V.													
2	HOST	RO	0	<p>Host Mode</p> <p>This read-only bit is set when the USB controller is acting as a Host.</p>										
1	HOSTREQ	R/W	0	<p>Host Request</p> <p>When set, the USB controller initiates the Host Negotiation when Suspend mode is entered. It is cleared when Host Negotiation is completed.</p>										
0	SESSION	R/W	0	<p>Session Start/End</p> <p><i>When operating as an OTG A device, this bit is set or cleared by the CPU to start or end a session.</i></p> <p><i>When operating as an OTG B device, this bit is set or cleared by the USB controller when a session starts or ends. It is also set by the CPU to initiate the Session Request Protocol. When the USB controller is in Suspend mode, the bit may be cleared by the CPU to perform a software disconnect.</i></p> <p><b>Note:</b> Clearing this bit when the core is not suspended will result in undefined behavior.</p>										

### Device Mode

#### USB Device Control (USBDEVCTL)

Base 0x4005.0000  
 Offset 0x060  
 Type R/W, reset 0x80



Bit/Field	Name	Type	Reset	Description										
7	DEV	RO	1	<p>Device Mode</p> <p>This read-only bit indicates whether the USB controller is operating as the OTG A device or the OTG B device.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>A device</td> </tr> <tr> <td>1</td> <td>B device</td> </tr> </tbody> </table> <p><b>Note:</b> This value is only valid while a session is in progress.</p>	Value	Description	0	A device	1	B device				
Value	Description													
0	A device													
1	B device													
6:5	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.										
4:3	VBUS	RO	0x00	<p>VBus Level</p> <p>These read-only bits encode the current VBus level as follows.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Below SessionEnd VBUS is detected as under 0.5 V.</td> </tr> <tr> <td>0x1</td> <td>Above SessionEnd, below AValid VBUS is detected as above 0.5 V and under 1.5 V.</td> </tr> <tr> <td>0x2</td> <td>Above AValid, below VBusValid VBUS is detected as above 1.5 V and below 4.5 V.</td> </tr> <tr> <td>0x3</td> <td>Above VBusValid VBUS is detected as above 4.5 V.</td> </tr> </tbody> </table>	Value	Description	0x0	Below SessionEnd VBUS is detected as under 0.5 V.	0x1	Above SessionEnd, below AValid VBUS is detected as above 0.5 V and under 1.5 V.	0x2	Above AValid, below VBusValid VBUS is detected as above 1.5 V and below 4.5 V.	0x3	Above VBusValid VBUS is detected as above 4.5 V.
Value	Description													
0x0	Below SessionEnd VBUS is detected as under 0.5 V.													
0x1	Above SessionEnd, below AValid VBUS is detected as above 0.5 V and under 1.5 V.													
0x2	Above AValid, below VBusValid VBUS is detected as above 1.5 V and below 4.5 V.													
0x3	Above VBusValid VBUS is detected as above 4.5 V.													
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	HOSTREQ	R/W	0	<p>Host Request</p> <p>When set, the USB controller initiates the Host Negotiation when Suspend mode is entered. It is cleared when Host Negotiation is completed.</p>										
0	SESSION	R/W	0	<p>Session Start/End</p> <p><i>When operating as an OTG A device</i>, this bit is set or cleared by the CPU to start or end a session.</p> <p><i>When operating as an OTG B device</i>, this bit is set or cleared by the USB controller when a session starts or ends. It is also set by the CPU to initiate the Session Request Protocol. When the USB controller is in Suspend mode, the bit may be cleared by the CPU to perform a software disconnect.</p> <p><b>Note:</b> Clearing this bit when the core is not suspended will result in undefined behavior.</p>										

**Register 17: USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ), offset 0x062**

**Register 18: USB Receive Dynamic FIFO Sizing (USBRXFIFOSZ), offset 0x063**

**Host**

These 8-bit registers allow the selected TX/RX endpoint FIFOs to be dynamically sized. **USBEPIDX** is used to configure each transmit endpoint's FIFO size.

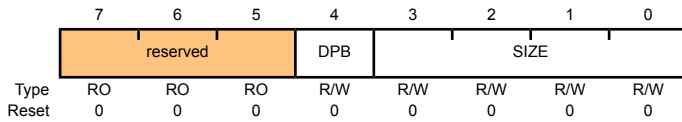
**Device**

USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ)

Base 0x4005.0000

Offset 0x062

Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:5	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	DPB	R/W	0	Double Packet Buffer Support  Defines whether double-packet buffering is supported. When 1, double-packet buffering is supported. When 0, only single-packet buffering is supported.
3:0	SIZE	R/W	0x0	Max Packet Size  Maximum packet size to be allowed for ( <i>before</i> any splitting within the FIFO of bulk/high-bandwidth packets prior to transmission.  If <b>DPB</b> = 0, the FIFO also is this size; if <b>DPB</b> = 1, the FIFO is twice this size.
	Value	Packet Size (Bytes)		
	0x0	8		
	0x1	16		
	0x2	32		
	0x3	64		
	0x4	128		
	0x5	256		
	0x6	512		
	0x7	1024		
	0x8	2048		
	0x9-0xF	Reserved		

**Register 19: USB Transmit FIFO Start Address (USBTXFIFOADD), offset 0x064**

**Register 20: USB Receive FIFO Start Address (USBRXFIFOADD), offset 0x066**

**Host**

**USBTXFIFOADD** is a 16-bit register that controls the start address of the selected transmit endpoint FIFO. **USBRXFIFOADD** is a 14-bit register that controls the start address of the selected receive endpoint FIFO.

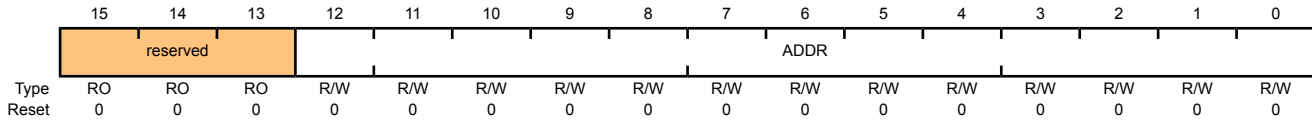
**Device**

USB Transmit FIFO Start Address (USBTXFIFOADD)

Base 0x4005.0000

Offset 0x064

Type R/W, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:13	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.
12:0	ADDR	R/W	0x00	Transmit/Receive Start Address

Start address of the endpoint FIFO in units of 8 bytes.

Value	Start Address
0x0	0
0x1	8
0x2	16
0x3	32
0x4	64
0x5	128
0x6	256
0x7	512
0x8	1024
0x9	2048
0xA-0x1FFF	Reserved

### Register 21: USB Connect Timing (USBCONTIM), offset 0x07A

OTG

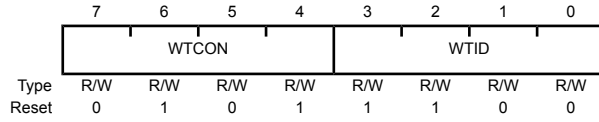
This 8-bit configuration register allows some delays to be specified.

Host

USB Connect Timing (USBCONTIM)

Base 0x4005.0000  
 Offset 0x07A  
 Type R/W, reset 0x5C

Device



Bit/Field	Name	Type	Reset	Description
7:4	WTCON	R/W	0x5	Connect Wait  Sets the wait to be applied to allow for the user's connect/disconnect filter, in units of 533.3 ns. (The default setting corresponds to 2.667µs.)
3:0	WTID	R/W	0xC	Wait ID  Set the delay to be applied from the enable of the ID detection to when the ID value is valid, in units of 4.369 ms. (The default setting corresponds to 52.43 ms.)

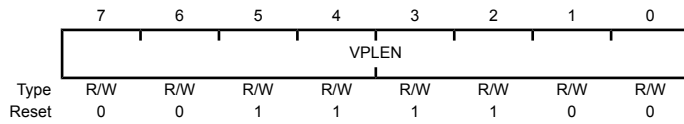


**Register 22: USB OTG VBus Pulse Timing (USBVPLEN), offset 0x07B****OTG**

This 8-bit configuration register sets the duration of the VBus pulsing charge.

## USB OTG VBus Pulse Timing (USBVPLEN)

Base 0x4005.0000  
 Offset 0x07B  
 Type R/W, reset 0x3C



Bit/Field	Name	Type	Reset	Description
7:0	VPLEN	R/W	0x3C	VBus Pulse Length

Sets the duration of the VBus pulsing charge in units of 546.1  $\mu$ s. (The default setting corresponds to 32.77 ms.)

### Register 23: USB Full-Speed Last Transaction to End of Frame Timing (USBFSEOF), offset 0x07D

**Host**

This 8-bit configuration register sets the minimum time gap that is to be allowed between the start of the last transaction and the EOF for full-speed transactions.

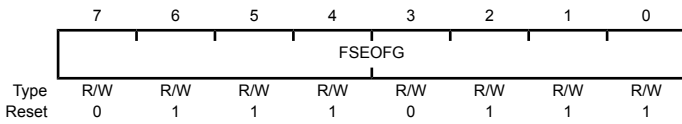
**Device**

USB Full-Speed Last Transaction to End of Frame Timing (USBFSEOF)

Base 0x4005.0000

Offset 0x07D

Type R/W, reset 0x77



Bit/Field	Name	Type	Reset	Description
7:0	FSEOFG	R/W	0x77	Full-Speed End-of-Frame Gap

Used during full-speed transactions, to set the gap between the last transaction and the End-of-Frame (EOF), in units of 533.3 ns. The default corresponds to 63.46  $\mu$ s.

## Register 24: USB Low-Speed Last Transaction to End of Frame Timing (USBLSEOF), offset 0x07E

### Host

This 8-bit configuration register sets the minimum time gap that is to be allowed between the start of the last transaction and the EOF for low-speed transactions.

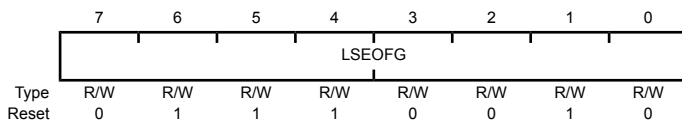
### Device

USB Low-Speed Last Transaction to End of Frame Timing (USBLSEOF)

Base 0x4005.0000

Offset 0x07E

Type R/W, reset 0x72



Bit/Field	Name	Type	Reset	Description
7:0	LSEOFG	R/W	0x72	Low-Speed End-of-Frame Gap

Used during low-speed transactions, to set the gap between the last transaction and the End-of-Frame (EOF), in units of 1.067  $\mu$ s. The default corresponds to 121.6  $\mu$ s.

**Register 25: USB Transmit Functional Address Endpoint 0 (USBTXFUNCADDR0), offset 0x080**

**Register 26: USB Transmit Functional Address Endpoint 1 (USBTXFUNCADDR1), offset 0x088**

**Register 27: USB Transmit Functional Address Endpoint 2 (USBTXFUNCADDR2), offset 0x090**

**Register 28: USB Transmit Functional Address Endpoint 3 (USBTXFUNCADDR3), offset 0x098**

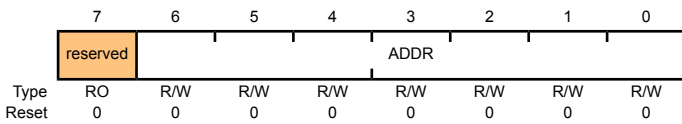
Host

USBTXFUNCADDR<sub>n</sub> is an 8-bit read/write register that records the address of the target function that is to be accessed through the associated endpoint (EP<sub>n</sub>). USBTXFUNCADDR<sub>n</sub> needs to be defined for each transmit endpoint that is used.

**Note:** USBTXFUNCADDR0 is used for both receive and transmit for endpoint 0.

USB Transmit Functional Address Endpoint 0 (USBTXFUNCADDR0)

Base 0x4005.0000  
 Offset 0x080  
 Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
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.
6:0	ADDR	R/W	0x00	Device Address USB bus address for the target device.

**Register 29: USB Transmit Hub Address Endpoint 0 (USBTXHUBADDR0), offset 0x082**

**Register 30: USB Transmit Hub Address Endpoint 1 (USBTXHUBADDR1), offset 0x08A**

**Register 31: USB Transmit Hub Address Endpoint 2 (USBTXHUBADDR2), offset 0x092**

**Register 32: USB Transmit Hub Address Endpoint 3 (USBTXHUBADDR3), offset 0x09A**

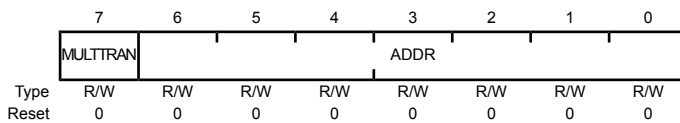
Host

**USBTXHUBADDR<sub>n</sub>** is an 8-bit read/write register that, like **USBTXHUBPORT<sub>n</sub>**, only needs to be written when a USB device is connected to transmit endpoint **EP<sub>n</sub>** via a USB 2.0 hub. This register records the address of that USB 2.0 hub through which the target associated with the endpoint is accessed. This information, together with the hub port in **USBTXHUBPORT<sub>n</sub>**, allows the USB controller to support split transactions.

**Note:** **USBTXHUBADDR0** is used for both receive and transmit for endpoint 0.

#### USB Transmit Hub Address Endpoint 0 (USBTXHUBADDR0)

Base 0x4005.0000  
Offset 0x082  
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	MULTTRAN	R/W	0	Multiple Translators  Indicates whether the hub has multiple transaction translators. Clear to 0 if single transaction translator; set to 1 if multiple transaction translators.
6:0	ADDR	R/W	0x00	Hub Address  USB bus address for the USB 2.0 hub.

**Register 33: USB Transmit Hub Port Endpoint 0 (USBTXHUBPORT0), offset 0x083**

**Register 34: USB Transmit Hub Port Endpoint 1 (USBTXHUBPORT1), offset 0x08B**

**Register 35: USB Transmit Hub Port Endpoint 2 (USBTXHUBPORT2), offset 0x093**

**Register 36: USB Transmit Hub Port Endpoint 3 (USBTXHUBPORT3), offset 0x09B**

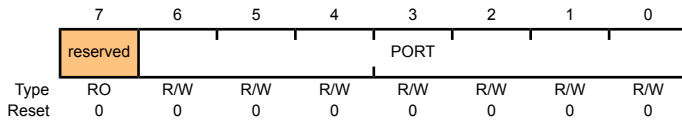
Host

USBTXHUBPORTn is an 8-bit read/write register that, like USBTXHUBADDRn, only needs to be written when a full- or low-speed device is connected to transmit endpoint EPn via a USB 2.0 hub. This register records the port of that USB 2.0 hub through which the target associated with the endpoint is accessed. This information, together with the hub address in USBTXHUBADDRn, allows the USB controller to support split transactions.

**Note:** USBTXHUBPORT0 is used for both receive and transmit for endpoint 0.

USB Transmit Hub Port Endpoint 0 (USBTXHUBPORT0)

Base 0x4005.0000  
 Offset 0x083  
 Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
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.
6:0	PORT	R/W	0x00	Hub Port USB hub port number.

### Register 37: USB Receive Functional Address Endpoint 1 (USBXFUNCAADDR1), offset 0x08C

### Register 38: USB Receive Functional Address Endpoint 2 (USBXFUNCAADDR2), offset 0x094

### Register 39: USB Receive Functional Address Endpoint 3 (USBXFUNCAADDR3), offset 0x09C

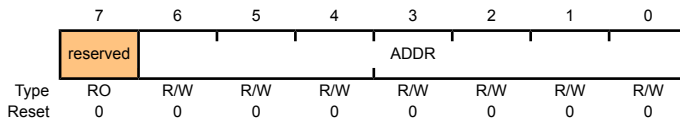
Host

**USBXFUNCAADDR<sub>n</sub>** is an 8-bit read/write register that records the address of the target function that is to be accessed through the associated endpoint (EP<sub>n</sub>). **USBXFUNCAADDR<sub>n</sub>** needs to be defined for each receive endpoint that is used.

**Note:** **USBTXFUNCAADDR0** is used for both receive and transmit for endpoint 0.

#### USB Receive Functional Address Endpoint 1 (USBXFUNCAADDR1)

Base 0x4005.0000  
Offset 0x08C  
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
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.
6:0	ADDR	R/W	0x00	Device Address USB bus address for the target device.

**Register 40: USB Receive Hub Address Endpoint 1 (USBRXHUBADDR1), offset 0x08E**

**Register 41: USB Receive Hub Address Endpoint 2 (USBRXHUBADDR2), offset 0x096**

**Register 42: USB Receive Hub Address Endpoint 3 (USBRXHUBADDR3), offset 0x09E**

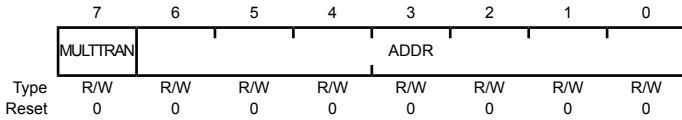
**Host**

**USBRXHUBADDRn** is an 8-bit read/write register that, like **USBRXHUBPORTn**, only needs to be written when a full- or low-speed device is connected to receive endpoint EPn via a USB 2.0 hub. This register records the address of that USB 2.0 hub through which the target associated with the endpoint is accessed. This information, together with the hub port in **USBRXHUBPORTn**, allows the USB controller to support split transactions.

**Note:** **USBTXHUBADDR0** is used for both receive and transmit for endpoint 0.

USB Receive Hub Address Endpoint 1 (USBRXHUBADDR1)

Base 0x4005.0000  
 Offset 0x08E  
 Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	MULTTRAN	R/W	0	Multiple Translators  Indicates whether the hub has multiple transaction translators. Clear to 0 if single transaction translator; set to 1 if multiple transaction translators.
6:0	ADDR	R/W	0x00	Hub Address  USB bus address for the USB 2.0 hub.



**Register 43: USB Receive Hub Port Endpoint 1 (USBRXHUBPORT1), offset 0x08F**

**Register 44: USB Receive Hub Port Endpoint 2 (USBRXHUBPORT2), offset 0x097**

**Register 45: USB Receive Hub Port Endpoint 3 (USBRXHUBPORT3), offset 0x09F**

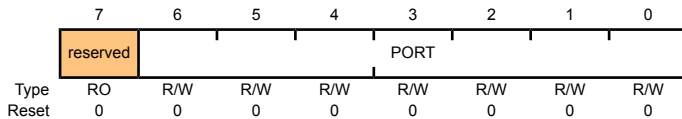
Host

**USBRXHUBPORT<sub>n</sub>** is an 8-bit read/write register that, like **USBRXHUBADDR<sub>n</sub>**, only needs to be written when a full- or low-speed device is connected to receive endpoint EP<sub>n</sub> via a USB 2.0 hub. This register records the port of that USB 2.0 hub through which the target associated with the endpoint is accessed. This information, together with the hub address in **USBTXHUBADDR<sub>n</sub>**, allows the USB controller to support split transactions.

**Note:** **USBTXHUBPORT0** is used for both receive and transmit for endpoint 0.

#### USB Receive Hub Port Endpoint 1 (USBRXHUBPORT1)

Base 0x4005.0000  
Offset 0x08F  
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
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.
6:0	PORT	R/W	0x00	Hub Port USB hub port number.

**Register 46: USB Maximum Transmit Data Endpoint 1 (USBTXMAXP1), offset 0x110**

**Register 47: USB Maximum Transmit Data Endpoint 2 (USBTXMAXP2), offset 0x120**

**Register 48: USB Maximum Transmit Data Endpoint 3 (USBTXMAXP3), offset 0x130**

**Host**

The **USBTXMAXPn** 16-bit register defines the maximum amount of data that can be transferred through the transmit endpoint in a single operation.

**Device**

Bits 10:0 define (in bytes) the maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the *USB Specification* on packet sizes for bulk, interrupt and isochronous transfers in full-speed operation.

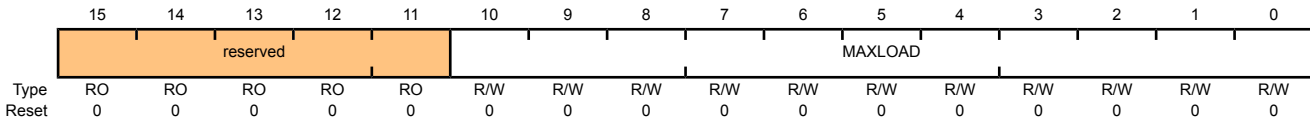
The total amount of data represented by the value written to this register (specified payload × *m*) must not exceed the FIFO size for the transmit endpoint, and should not exceed half the FIFO size if double-buffering is required.

If this register is changed after packets have been sent from the endpoint, the transmit endpoint FIFO should be completely flushed (using the **FLUSH** bit in **USBTXCSRL1n**) after writing the new value to this register.

**Note:** **USBTXMAXPn** must be set to an even number of bytes for proper interrupt generation in  $\mu$ DMA Mode 1.

USB Maximum Transmit Data Endpoint 1 (USBTXMAXP1)

Base 0x4005.0000  
 Offset 0x110  
 Type R/W, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10:0	MAXLOAD	R/W	0x00	Maximum Payload The maximum payload in bytes per transaction.

## Register 49: USB Control and Status Endpoint 0 Low (USBCSRL0), offset 0x102

Host

USBCSRL0 is an 8-bit register that provides control and status bits for endpoint 0.

Device

### Host Mode

#### USB Control and Status Endpoint 0 Low (USBCSRL0)

Base 0x4005.0000

Offset 0x102

Type W1C, reset 0x00

	7	6	5	4	3	2	1	0
Type	R/W0C	R/W	R/W	R/W0C	R/W1S	R/W0C	R/W1S	R/W0C
Reset	0	0	0	0	0	0	0	0

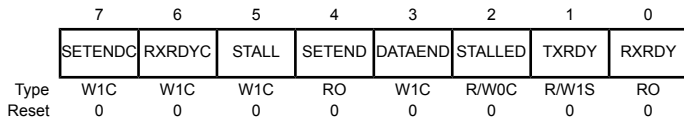
Bit/Field	Name	Type	Reset	Description
7	NAKTO	R/W0C	0	<p>NAK Timeout</p> <p>This bit is set by the USB controller when endpoint 0 is halted following the receipt of NAK responses for longer than the time set by the <b>USBNAKLMT</b> register. The CPU should clear this bit by writing a 0 to it to allow the endpoint to continue.</p>
6	STATUS	R/W	0	<p>Status Packet</p> <p>The CPU sets this bit at the same time as the <b>TXRDY</b> or <b>REQPKT</b> bit is set, to perform a status stage transaction. Setting this bit ensures <b>DT</b> is set to 1 so that a <b>DATA1</b> packet is used for the Status Stage transaction.</p>
5	REQPKT	R/W	0	<p>Request Packet</p> <p>The CPU sets this bit to request an IN transaction. It is cleared when <b>RXRDY</b> is set.</p>
4	ERROR	R/W0C	0	<p>Error</p> <p>This bit is set by the USB controller when three attempts have been made to perform a transaction with no response from the peripheral. The CPU should clear this bit. An interrupt is generated when this bit is set.</p>
3	SETUP	R/W1S	0	<p>Setup Packet</p> <p>The CPU sets this bit, at the same time as the <b>TXRDY</b> bit is set, to send a <b>SETUP</b> token instead of an <b>OUT</b> token for the transaction. This always resets the data toggle and sends a <b>DATA0</b> packet.</p>
2	STALLED	R/W0C	0	<p>Endpoint Stalled</p> <p>This bit is set when a <b>STALL</b> handshake is received. The CPU should clear this bit.</p>

Bit/Field	Name	Type	Reset	Description
1	TXRDY	R/W1S	0	<p>Transmit Packet Ready</p> <p>The CPU sets this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. An interrupt is also generated at this point.</p>
0	RXRDY	R/W0C	0	<p>Receive Packet Ready</p> <p>This bit is set when a data packet has been received. An interrupt is generated when this bit is set. The CPU should clear this bit, by writing a 0 when the packet has been read from the FIFO. This acknowledges that data has been read from the FIFO.</p>

**Device Mode**

USB Control and Status Endpoint 0 Low (USBCSRL0)

Base 0x4005.0000  
 Offset 0x102  
 Type W1C, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	SETENDC	W1C	0	<p>Setup End Clear</p> <p>The CPU writes a 1 to this bit to clear the SETEND bit.</p>
6	RXRDYC	W1C	0	<p>RXRDY Clear</p> <p>The CPU writes a 1 to this bit to clear the RXRDY bit.</p>
5	STALL	W1C	0	<p>Send Stall</p> <p>The CPU writes a 1 to this bit to terminate the current transaction. The STALL handshake is transmitted, and then this bit is cleared automatically.</p>
4	SETEND	RO	0	<p>Setup End</p> <p>This bit is set when a control transaction ends before the DataEnd bit has been set. An interrupt is generated and the FIFO flushed at this time. The bit is cleared by the CPU writing a 1 to the SETENDC bit.</p>
3	DATAEND	W1C	0	<p>Data End</p> <p>The CPU sets this bit:</p> <ul style="list-style-type: none"> <li>■ When setting TXRDY for the last data packet</li> <li>■ When clearing RXRDY after unloading the last data packet</li> <li>■ When setting TXRDY for a zero-length data packet</li> </ul> <p>It is cleared automatically.</p>

---

Bit/Field	Name	Type	Reset	Description
2	STALLED	R/W0C	0	Endpoint Stalled  This bit is set when a STALL handshake is transmitted. The CPU should clear this bit by writing a 0. This bit can only be cleared. Setting this bit does nothing.
1	TXRDY	R/W1S	0	Transmit Packet Ready  The CPU writes a 1 to this bit after loading a data packet into the FIFO. It is cleared automatically when the data packet has been transmitted. An interrupt is also generated at this point.
0	RXRDY	RO	0	Receive Packet Ready  This bit is set when a data packet has been received. An interrupt is generated when this bit is set. The CPU clears this bit by setting the RXRDYC bit.

## Register 50: USB Control and Status Endpoint 0 High (USBCSRH0), offset 0x103

Host

USBSR0H is an 8-bit register that provides control and status bits for endpoint 0.

Device

### Host Mode

#### USB Control and Status Endpoint 0 High (USBCSRH0)

Base 0x4005.0000  
Offset 0x103  
Type W1C, reset 0x00

	7	6	5	4	3	2	1	0
	reserved					DTWE	DT	FLUSH
Type	RO	RO	RO	RO	RO	W1S	R/W	W1C
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7:3	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	DTWE	W1S	0	Data Toggle Write Enable  The CPU writes a 1 to this bit to enable the current state of the endpoint 0 data toggle to be written (see DT bit). This bit is automatically cleared once the new value is written.
1	DT	R/W	0	Data Toggle  When read, this bit indicates the current state of the endpoint 0 data toggle. If DTWE is High, this bit may be written with the required setting of the data toggle. If DTWE is Low, this cannot be written.
0	FLUSH	W1C	0	Flush FIFO  The CPU writes a 1 to this bit to flush the next packet to be transmitted/read from the endpoint 0 FIFO. The FIFO pointer is reset and the TXRDY/RXRDY bit is cleared.

**Important:** FLUSH should only be used when TXRDY/RXRDY is set. At other times, it may cause data to be corrupted.

### Device Mode

#### USB Control and Status Endpoint 0 High (USBCSRH0)

Base 0x4005.0000  
Offset 0x103  
Type W1C, reset 0x00

	7	6	5	4	3	2	1	0
	reserved							FLUSH
Type	RO	RO	RO	RO	RO	RO	RO	W1S
Reset	0	0	0	0	0	0	0	0

---

Bit/Field	Name	Type	Reset	Description
7:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	FLUSH	W1S	0	Flush FIFO  The CPU writes a 1 to this bit to flush the next packet to be transmitted/read from the endpoint 0 FIFO. The FIFO pointer is reset and the TXRDY/RXRDY bit is cleared.  <b>Important:</b> FLUSH should only be used when TXRDY/RXRDY is set. At other times, it may cause data to be corrupted.

---

**Register 51: USB Receive Byte Count Endpoint 0 (USBCOUNT0), offset 0x108**

**Host**

**USBCOUNT0** is an 8-bit read-only register that indicates the number of received data bytes in the endpoint 0 FIFO. The value returned changes as the contents of the FIFO change and is only valid while **RXRDY** is set.

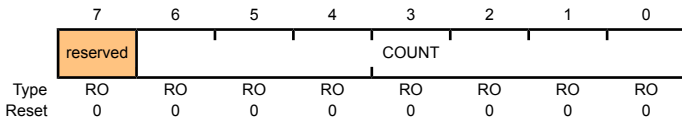
**Device**

USB Receive Byte Count Endpoint 0 (USBCOUNT0)

Base 0x4005.0000

Offset 0x108

Type RO, reset 0x00



Bit/Field	Name	Type	Reset	Description
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.
6:0	COUNT	RO	0x00	Count Count is a read-only value that indicates the number of received data bytes in the endpoint 0 FIFO.



**Register 52: USB Type Endpoint 0 (USBTYPE0), offset 0x10A****Host**

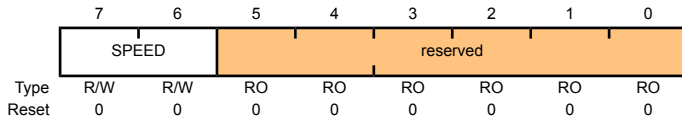
This is an 8-bit register that should be written with the operating speed of the targeted device being communicated with using endpoint 0.

## USB Type Endpoint 0 (USBTYPE0)

Base 0x4005.0000

Offset 0x10A

Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:6	SPEED	R/W	0x00	Operating Speed  Operating speed of the target device. If selected, the target is assumed to have the same connection speed as the core.  Value Description 00 Reserved 01 Reserved 10 Full 11 Low
5:0	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 53: USB NAK Limit (USBNAKLMT), offset 0x10B**

Host

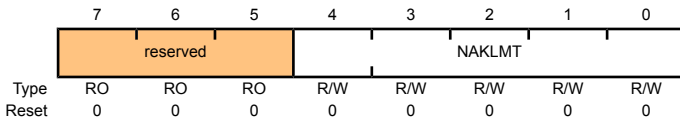
**USBNAKLMT** is an 8-bit register that sets the number of frames after which endpoint 0 should time out on receiving a stream of NAK responses. (Equivalent settings for other endpoints can be made through their **USBTXINTERVALn** and **USBRXINTERVALn** registers.)

The number of frames selected is  $2^{(m-1)}$  (where  $m$  is the value set in the register, with valid values of 2–16). If the host receives NAK responses from the target for more frames than the number represented by the limit set in this register, the endpoint is halted.

**Note:** A value of 0 or 1 disables the NAK timeout function.

USB NAK Limit (USBNAKLMT)

Base 0x4005.0000  
 Offset 0x10B  
 Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:5	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.
4:0	NAKLMT	R/W	0x00	EP0 NAK Limit Number of frames after receiving a stream of NAK responses.

**Register 54: USB Transmit Control and Status Endpoint 1 Low (USBTXCSRL1), offset 0x112**

**Register 55: USB Transmit Control and Status Endpoint 2 Low (USBTXCSRL2), offset 0x122**

**Register 56: USB Transmit Control and Status Endpoint 3 Low (USBTXCSRL3), offset 0x132**

Host

**USBTXCSRLn** is an 8-bit register that provides control and status bits for transfers through the currently selected transmit endpoint.

Device

### Host Mode

USB Transmit Control and Status Endpoint 1 Low (USBTXCSRL1)

Base 0x4005.0000

Offset 0x112

Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	NAKTO / INCTX	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
Type	R/W0C	W1S	R/W0C	R/W	W1C	R/W0C	R/W0C	R/W0C
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	NAKTO / INCTX	R/W0C	0	<p>NAK Timeout / Incomplete TX</p> <p><i>Bulk endpoints only:</i> This bit is set when the transmit endpoint is halted following the receipt of NAK responses for longer than the time set as the NAK Limit by the <b>USBTXINTERVALn</b> register. The CPU should clear this bit to allow the endpoint to continue.</p> <p><i>High-bandwidth interrupt endpoints only:</i> This bit is set if no response is received from the device to which the packet is being sent.</p>
6	CLRDT	W1S	0	<p>Clear Data Toggle</p> <p>The CPU writes a 1 to this bit to reset the endpoint data toggle to 0.</p>
5	STALLED	R/W0C	0	<p>Endpoint Stalled</p> <p>This bit is set when a STALL handshake is received. When this bit is set, any <math>\mu</math>DMA request that is in progress is stopped, the FIFO is completely flushed, and the <b>TXRDY</b> bit is cleared. The CPU should clear this bit.</p>
4	SETUP	R/W	0	<p>Setup Packet</p> <p>The CPU sets this bit, at the same time as the <b>TXRDY</b> bit is set, to send a SETUP token instead of an OUT token for the transaction.</p> <p><b>Note:</b> Setting this bit also clears <b>DT</b>.</p>

Bit/Field	Name	Type	Reset	Description
3	FLUSH	W1C	0	<p>Flush FIFO</p> <p>The CPU writes a 1 to this bit to flush the latest packet from the endpoint transmit FIFO. The FIFO pointer is reset, the <code>TXRDY</code> bit is cleared, and an interrupt is generated. <code>FLUSH</code> may be set simultaneously with <code>TXRDY</code> to abort the packet that is currently being loaded into the FIFO.</p> <p><b>Note:</b> <code>FLUSH</code> should only be used when <code>TXRDY</code> is set. At other times, it may cause data to be corrupted. Also note that, if the FIFO is double-buffered, <code>FLUSH</code> may need to be set twice to completely clear the FIFO.</p>
2	ERROR	R/W0C	0	<p>Error</p> <p>The USB sets this bit when three attempts have been made to send a packet and no handshake packet has been received. When the bit is set, an interrupt is generated, <code>TXRDY</code> is cleared, and the FIFO is completely flushed. The CPU should clear this bit.</p> <p><b>Note:</b> This is valid only when the endpoint is operating in Bulk or Interrupt mode.</p>
1	FIFONE	R/W0C	0	<p>FIFO Not Empty</p> <p>The USB controller sets this bit when there is at least one packet in the transmit FIFO.</p>
0	TXRDY	R/W0C	0	<p>Transmit Packet Ready</p> <p>The CPU sets this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. An interrupt is generated at this point. <code>TXRDY</code> is also automatically cleared prior to loading a second packet into a double-buffered FIFO.</p>

## Device Mode

### USB Transmit Control and Status Endpoint 1 Low (USBTXCSRL1)

Base 0x4005.0000  
 Offset 0x112  
 Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	INCTX	CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
Type	R/W0C	W1S	R/W0C	R/W	W1C	R/W0C	R/W0C	R/W1S
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	INCTX	R/W0C	0	<p>Incomplete Transmit</p> <p>When the endpoint is being used for high-bandwidth isochronous transfers, this bit is set to indicate where a large packet has been split into 2 or 3 packets for transmission but insufficient IN tokens have been received to send all the parts.</p> <p><b>Note:</b> Only valid for isochronous transfers.</p>
6	CLRDT	W1S	0	<p>Clear Data Toggle</p> <p>The CPU writes a 1 to this bit to reset the endpoint data toggle to 0.</p>

Bit/Field	Name	Type	Reset	Description
5	STALLED	R/W0C	0	<p>Endpoint Stalled</p> <p>This bit is set when a STALL handshake is transmitted. The FIFO is flushed and the <code>TXRDY</code> bit is cleared. The CPU should clear this bit.</p>
4	STALL	R/W	0	<p>Send Stall</p> <p>The CPU writes a 1 to this bit to issue a STALL handshake to an IN token. The CPU clears this bit to terminate the stall condition.</p> <p><b>Note:</b> This bit has no effect in isochronous transfers.</p>
3	FLUSH	W1C	0	<p>Flush FIFO</p> <p>The CPU writes a 1 to this bit to flush the latest packet from the endpoint transmit FIFO. The FIFO pointer is reset, the <code>TXRDY</code> bit is cleared, and an interrupt is generated. This bit may be set simultaneously with <code>TXRDY</code> to abort the packet that is currently being loaded into the FIFO.</p> <p><b>Note:</b> <code>FLUSH</code> should only be used when <code>TXRDY</code> is set. At other times, it may cause data to be corrupted. Also note that, if the FIFO is double-buffered, <code>FLUSH</code> may need to be set twice to completely clear the FIFO.</p>
2	UNDRN	R/W0C	0	<p>Underrun</p> <p>The USB controller sets this bit if an IN token is received when <code>TXRDY</code> is not set. The CPU should clear this bit.</p>
1	FIFONE	R/W0C	0	<p>FIFO Not Empty</p> <p>The USB controller sets this bit when there is at least 1 packet in the transmit FIFO.</p>
0	TXRDY	R/W1S	0	<p>Transmit Packet Ready</p> <p>The CPU sets this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. An interrupt is generated at this point. <code>TXRDY</code> is also automatically cleared prior to loading a second packet into a double-buffered FIFO.</p>

**Register 57: USB Transmit Control and Status Endpoint 1 High (USBTXCSRH1), offset 0x113**

**Register 58: USB Transmit Control and Status Endpoint 2 High (USBTXCSRH2), offset 0x123**

**Register 59: USB Transmit Control and Status Endpoint 3 High (USBTXCSRH3), offset 0x133**

Host

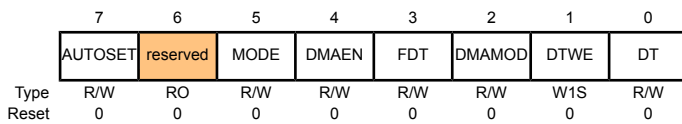
**USBTXCSRHn** is an 8-bit register that provides additional control for transfers through the currently selected transmit endpoint.

Device

**Host Mode**

USB Transmit Control and Status Endpoint 1 High (USBTXCSRH1)

Base 0x4005.0000  
 Offset 0x113  
 Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	AUTOSET	R/W	0	Auto Set  If the CPU sets this bit, <b>TXRDY</b> is automatically set when data of the maximum packet size (value in <b>USBTXMAXPn</b> ) is loaded into the transmit FIFO. If a packet of less than the maximum packet size is loaded, then <b>TXRDY</b> must be set manually.  <b>Note:</b> This bit should not be set for either high-bandwidth isochronous or high-bandwidth interrupt endpoints.
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	MODE	R/W	0	Mode  The CPU sets this bit to enable the endpoint direction as TX, and clears it to enable the endpoint direction as RX.  <b>Note:</b> This bit only has an effect when the same endpoint FIFO is used for both transmit and receive transactions.
4	DMAEN	R/W	0	DMA Request Enable  The CPU sets this bit to enable the $\mu$ DMA request for the transmit endpoint.

Bit/Field	Name	Type	Reset	Description
3	FDT	R/W	0	Force Data Toggle  The CPU sets this bit to force the endpoint data toggle to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This can be used by interrupt transmit endpoints that are used to communicate rate feedback for isochronous endpoints.
2	DMAMOD	R/W	0	DMA Request Mode  The CPU sets this bit to select $\mu$ DMA Request Mode 1 and clears it to select $\mu$ DMA Request Mode 0.  <b>Note:</b> This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.
1	DTWE	W1S	0	Data Toggle Write Enable  The CPU writes a 1 to this bit to enable the current state of the transmit endpoint data toggle to be written (see DT). This bit is automatically cleared once the new value is written.
0	DT	R/W	0	Data Toggle  When read, this bit indicates the current state of the transmit endpoint data toggle. If DTWE is High, this bit may be written with the required setting of the data toggle. If DTWE is Low, any value written to this bit is ignored.

## Device Mode

### USB Transmit Control and Status Endpoint 1 High (USBTXCSRH1)

Base 0x4005.0000  
Offset 0x113  
Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD	reserved	reserved
Type	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	AUTOSET	R/W	0	Auto Set  If the CPU sets this bit, TXRDY is automatically set when data of the maximum packet size (value in <b>USBTXMAXPn</b> ) is loaded into the transmit FIFO. If a packet of less than the maximum packet size is loaded, then TXRDY must be set manually.  <b>Note:</b> This bit should not be set for either high-bandwidth isochronous or high-bandwidth interrupt endpoints.
6	ISO	R/W	0	ISO  The CPU sets this bit to enable the transmit endpoint for isochronous transfers, and clears it to enable the transmit endpoint for bulk or interrupt transfers.

Bit/Field	Name	Type	Reset	Description
5	MODE	R/W	0	<p>Mode</p> <p>The CPU sets this bit to enable the endpoint direction as TX, and clears the bit to enable it as RX.</p> <p><b>Note:</b> This bit only has an effect where the same endpoint FIFO is used for both transmit and receive transactions.</p>
4	DMAEN	R/W	0	<p>DMA Request Enable</p> <p>The CPU sets this bit to enable the <math>\mu</math>DMA request for the transmit endpoint.</p>
3	FDT	R/W	0	<p>Force Data Toggle</p> <p>The CPU sets this bit to force the endpoint data toggle to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This can be used by interrupt transmit endpoints that are used to communicate rate feedback for isochronous endpoints.</p>
2	DMAMOD	R/W	0	<p>DMA Request Mode</p> <p>The CPU sets this bit to select <math>\mu</math>DMA Request Mode 1 and clears it to select <math>\mu</math>DMA Request Mode 0.</p> <p><b>Note:</b> This bit must not be cleared either before or in the same cycle as the above <i>DMAEN</i> bit is cleared.<sup>2</sup></p>
1:0	reserved	RO	0x00	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>



**Register 60: USB Maximum Receive Data Endpoint 1 (USBRXMAXP1), offset 0x114**

**Register 61: USB Maximum Receive Data Endpoint 2 (USBRXMAXP2), offset 0x124**

**Register 62: USB Maximum Receive Data Endpoint 3 (USBRXMAXP3), offset 0x134**

**Host**

The **USBRXMAXPn** 16-bit register defines the maximum amount of data that can be transferred through the selected receive endpoint in a single operation.

**Device**

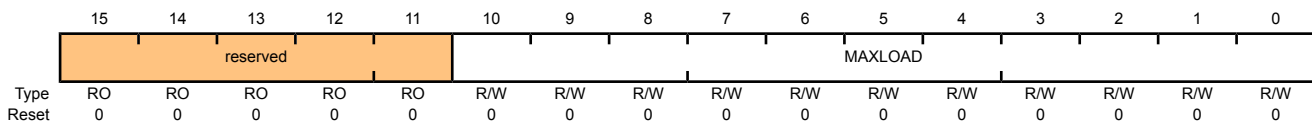
Bits 10:0 define (in bytes) the maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the *USB Specification* on packet sizes for bulk, interrupt and isochronous transfers in full-speed operations.

The total amount of data represented by the value written to this register (specified payload  $\times$  *m*) must not exceed the FIFO size for the receive endpoint, and should not exceed half the FIFO size if double-buffering is required.

**Note:** **USBRXMAXPn** must be set to an even number of bytes for proper interrupt generation in  $\mu$ DMA Mode 1.

USB Maximum Receive Data Endpoint 1 (USBRXMAXP1)

Base 0x4005.0000  
Offset 0x114  
Type R/W, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10:0	MAXLOAD	R/W	0x00	Maximum Payload The maximum payload in bytes per transaction.

**Register 63: USB Receive Control and Status Endpoint 1 Low (USBXCSRL1), offset 0x116**

**Register 64: USB Receive Control and Status Endpoint 2 Low (USBXCSRL2), offset 0x126**

**Register 65: USB Receive Control and Status Endpoint 3 Low (USBXCSRL3), offset 0x136**

Host

**USBXCSRLn** is an 8-bit register that provides control and status bits for transfers through the currently selected receive endpoint.

Device

**Host Mode**

USB Receive Control and Status Endpoint 1 Low (USBXCSRL1)

Base 0x4005.0000  
 Offset 0x116  
 Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
Type	W1S	R/W0C	R/W	W1S	R/W0C	R/W0C	RO	R/W0C
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	CLRDT	W1S	0	Clear Data Toggle The CPU writes a 1 to this bit to reset the endpoint data toggle to 0.
6	STALLED	R/W0C	0	Endpoint Stalled When a STALL handshake is received, this bit is set and an interrupt is generated. The CPU should clear this bit.
5	REQPKT	R/W	0	Request Packet The CPU writes a 1 to this bit to request an IN transaction. It is cleared when RXRDY is set.
4	FLUSH	W1S	0	Flush FIFO The CPU writes a 1 to this bit to flush the next packet to be read from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared.

**Note:** FLUSH should only be used when RXRDY is set. At other times, it may cause data to be corrupted. Also note that, if the FIFO is double-buffered, FLUSH may need to be set twice to completely clear the FIFO.

Bit/Field	Name	Type	Reset	Description
3	DATAERR / NAKTO	R/W0C	0	Data Error / NAK Timeout  When operating in ISO mode, this bit is set when <code>RXRDY</code> is set if the data packet has a CRC or bit-stuff error and cleared when <code>RXRDY</code> is cleared. In Bulk mode, this bit is set when the receive endpoint is halted following the receipt of NAK responses for longer than the time set as the NAK Limit by the <code>USBRXINTERVALn</code> register. The CPU should clear this bit to allow the endpoint to continue.
2	ERROR	R/W0C	0	Error  The USB sets this bit when three attempts have been made to receive a packet and no data packet has been received. The CPU should clear this bit. An interrupt is generated when the bit is set.  <b>Note:</b> This bit is only valid when the receive endpoint is operating in Bulk or Interrupt mode. In ISO mode, it always returns zero.
1	FULL	RO	0	FIFO Full  This bit is set when no more packets can be loaded into the receive FIFO.
0	RXRDY	R/W0C	0	Receive Packet Ready  This bit is set when a data packet has been received. The CPU should clear this bit when the packet has been unloaded from the receive FIFO. An interrupt is generated when the bit is set.

## Device Mode

### USB Receive Control and Status Endpoint 1 Low (USBRXCSSL1)

Base 0x4005.0000  
Offset 0x116  
Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
Type	W1S	R/W0C	R/W	W1S	RO	R/W0C	RO	R/W0C
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	CLRDT	W1S	0	Clear Data Toggle  The CPU writes a 1 to this bit to reset the endpoint data toggle to 0.
6	STALLED	R/W0C	0	Endpoint Stalled  This bit is set when a STALL handshake is transmitted. The CPU should clear this bit.
5	STALL	R/W	0	Send Stall  The CPU writes a 1 to this bit to issue a STALL handshake. The CPU clears this bit to terminate the stall condition.  <b>Note:</b> This bit has no effect where the endpoint is being used for isochronous transfers.

Bit/Field	Name	Type	Reset	Description
4	FLUSH	W1S	0	<p>Flush FIFO</p> <p>The CPU writes a 1 to this bit to flush the next packet to be read from the endpoint receive FIFO. The FIFO pointer is reset and the <code>RXRDY</code> bit is cleared.</p> <p><b>Note:</b> The <code>FLUSH</code> bit should only be used when <code>RXRDY</code> is set. At other times, it may cause data to be corrupted. Also note that, if the FIFO is double-buffered, <code>FLUSH</code> may need to be set twice to completely clear the FIFO.</p>
3	DATAERR	RO	0	<p>Data Error</p> <p>This bit is set when <code>RXRDY</code> is set if the data packet has a CRC or bit-stuff error. It is cleared when <code>RXRDY</code> is cleared.</p> <p><b>Note:</b> This bit is only valid when the endpoint is operating in ISO mode. In Bulk mode, it always returns zero.</p>
2	OVER	R/W0C	0	<p>Overflow</p> <p>This bit is set if an OUT packet cannot be loaded into the receive FIFO. The CPU should clear this bit.</p> <p><b>Note:</b> This bit is only valid when the endpoint is operating in ISO mode. In Bulk mode, it always returns zero.</p>
1	FULL	RO	0	<p>FIFO Full</p> <p>This bit is set when no more packets can be loaded into the receive FIFO.</p>
0	RXRDY	R/W0C	0	<p>Receive Packet Ready</p> <p>This bit is set when a data packet has been received. The CPU should clear this bit when the packet has been unloaded from the receive FIFO. An interrupt is generated when the bit is set.</p>

**Register 66: USB Receive Control and Status Endpoint 1 High (USBXCSRH1), offset 0x117**

**Register 67: USB Receive Control and Status Endpoint 2 High (USBXCSRH2), offset 0x127**

**Register 68: USB Receive Control and Status Endpoint 3 High (USBXCSRH3), offset 0x137**

Host

**USBXCSRHn** is an 8-bit register that provides additional control and status bits for transfers through the currently selected receive endpoint.

Device

### Host Mode

#### USB Receive Control and Status Endpoint 1 High (USBXCSRH1)

Base 0x4005.0000

Offset 0x117

Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	INCRX
Type	R/W	R/W	R/W	RO	R/W	RO	RO	R/W0C
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	AUTOCL	R/W	0	Auto Clear

If the CPU sets this bit, then the `RXRDY` bit is automatically cleared when a packet of `USBXMAXPn` bytes has been unloaded from the receive FIFO. When packets of less than the maximum packet size are unloaded, `RXRDY` must be cleared manually. Care must be taken when using  $\mu$ DMA to unload the receive FIFO as data is read from the receive FIFO in 4 byte chunks regardless of the `RxMaxP` bit.

**Note:** This bit should not be set for high-bandwidth isochronous endpoints.

6	AUTORQ	R/W	0	Auto Request
---	--------	-----	---	--------------

If the CPU sets this bit, the `ReqPkt` bit is automatically set when the `RXRDY` bit is cleared.

**Note:** This bit is automatically cleared when a short packet is received.

5	DMAEN	R/W	0	DMA Request Enable
---	-------	-----	---	--------------------

The CPU sets this bit to enable the  $\mu$ DMA request for the receive endpoint.

4	PIDERR	RO	0	PID Error
---	--------	----	---	-----------

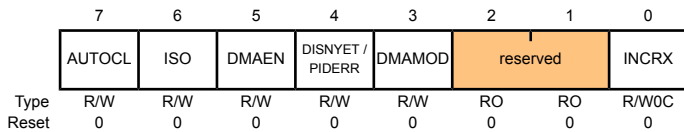
For ISO transactions, the core sets this bit to indicate a PID error in the received packet. This bit is ignored in bulk or interrupt transactions.

Bit/Field	Name	Type	Reset	Description
3	DMAMOD	R/W	0	DMA Request Mode The CPU sets this bit to select $\mu$ DMA Request Mode 1 and clears it to select $\mu$ DMA Request Mode 0.
2	DTWE	RO	0	Data Toggle Write Enable The CPU writes a 1 to this bit to enable the current state of the endpoint 0 data toggle to be written (see DT). This bit is automatically cleared once the new value is written.
1	DT	RO	0	Data Toggle When read, this bit indicates the current state of the endpoint 0 data toggle. If DTWE is High, this bit may be written with the required setting of the data toggle. If DTWE is Low, any value written to this bit is ignored.
0	INCRX	R/W0C	0	Incomplete Receive This bit is set in a high-bandwidth isochronous or interrupt transfer if the packet received is incomplete. It is cleared when RXRDY is cleared. <b>Note:</b> If USB protocols are followed correctly, this bit should never be set. The bit becoming set indicates a failure of the associated peripheral device to behave correctly. (In anything other than isochronous transfer, this bit always returns 0.)

Device Mode

USB Receive Control and Status Endpoint 1 High (USBXRCSRH1)

Base 0x4005.0000  
Offset 0x117  
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	AUTOCL	R/W	0	Auto Clear If the CPU sets this bit, then the RXRDY bit is automatically cleared when a packet of RXMaxP bytes has been unloaded from the receive FIFO. When packets of less than the maximum packet size are unloaded, RXRDY must be cleared manually. Care must be taken when using $\mu$ DMA to unload the receive FIFO as data is read from the receive FIFO in 4 byte chunks regardless of the RXMaxP bit. <b>Note:</b> This bit should not be set for high-bandwidth isochronous endpoints.
6	ISO	R/W	0	ISO The CPU sets this bit to enable the receive endpoint for isochronous transfers, and clears it to enable the receive endpoint for bulk/interrupt transfers.

Bit/Field	Name	Type	Reset	Description
5	DMAEN	R/W	0	<p>DMA Request Enable</p> <p>The CPU sets this bit to enable the <math>\mu</math>DMA request for the receive endpoint.</p>
4	DISNYET / PIDERR	R/W	0	<p>Disable NYET / PID Error</p> <p>For bulk or interrupt transactions, the CPU sets this bit to disable the sending of NYET handshakes. When set, all successfully received packets are acknowledged, including at the point at which the FIFO becomes full.</p> <p>For ISO transactions, the core sets this bit to indicate a PID error in the received packet.</p>
3	DMAMOD	R/W	0	<p>DMA Request Mode</p> <p>The CPU sets this bit to select <math>\mu</math>DMA Request Mode 1 and clears it to select <math>\mu</math>DMA Request Mode 0.</p>
2:1	reserved	RO	0x00	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>
0	INCRX	R/W0C	0	<p>Incomplete Receive</p> <p>This bit is set in a high-bandwidth isochronous/interrupt transfer if the packet in the receive FIFO is incomplete because parts of the data were not received. It is cleared when <math>RXRDY</math> is cleared.</p> <p><b>Note:</b> Only valid for isochronous transfers.</p>

**Register 69: USB Receive Byte Count Endpoint 1 (USBRXCOUNT1), offset 0x118**

**Register 70: USB Receive Byte Count Endpoint 2 (USBRXCOUNT2), offset 0x128**

**Register 71: USB Receive Byte Count Endpoint 3 (USBRXCOUNT3), offset 0x138**

**Host**

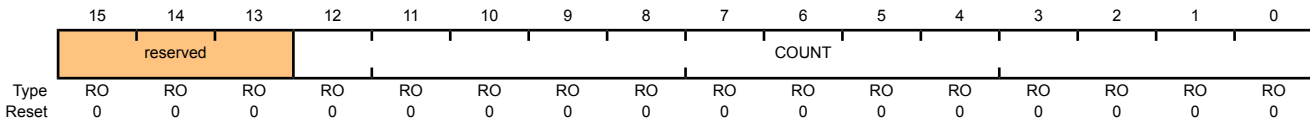
**Note:** The value returned changes as the FIFO is unloaded and is only valid while the `RXRDY` bit in the `USBRXCSSLn` register is set.

**Device**

**USBRXCount1** is a 16-bit read-only register that holds the number of data bytes in the packet currently in line to be read from the receive FIFO. If the packet is transmitted as multiple bulk packets, the number given is for the combined packet.

USB Receive Byte Count Endpoint 1 (USBRXCOUNT1)

Base 0x4005.0000  
 Offset 0x118  
 Type RO, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:13	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.
12:0	COUNT	RO	0x00	Receive Packet Count Number of bytes in the receive packet.



**Register 72: USB Host Transmit Configure Type Endpoint 1 (USBTXTYPE1), offset 0x11A**

**Register 73: USB Host Transmit Configure Type Endpoint 2 (USBTXTYPE2), offset 0x12A**

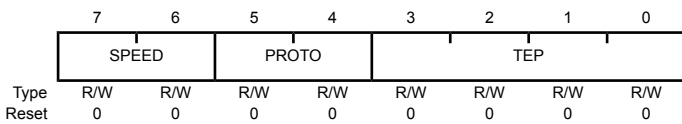
**Register 74: USB Host Transmit Configure Type Endpoint 3 (USBTXTYPE3), offset 0x13A**

**Host**

**USBTXTYPE1** is an 8-bit register that should be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently selected transmit endpoint, and its operating speed.

USB Host Transmit Configure Type Endpoint 1 (USBTXTYPE1)

Base 0x4005.0000  
 Offset 0x11A  
 Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description												
7:6	SPEED	R/W	0x00	Operating Speed  Operating speed of the target device when the core is configured with the hub option:  <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>00</td> <td>Default</td> </tr> <tr> <td></td> <td>The target is assumed to be using the same connection speed as the core.</td> </tr> <tr> <td>01</td> <td>Reserved</td> </tr> <tr> <td>10</td> <td>Full</td> </tr> <tr> <td>11</td> <td>Low</td> </tr> </tbody> </table> When the core is not configured with the hub option, these bits should not be accessed	Value	Description	00	Default		The target is assumed to be using the same connection speed as the core.	01	Reserved	10	Full	11	Low
Value	Description															
00	Default															
	The target is assumed to be using the same connection speed as the core.															
01	Reserved															
10	Full															
11	Low															
5:4	PROTO	R/W	0x00	Protocol  The CPU should set this to select the required protocol for the transmit endpoint:  <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>00</td> <td>Control</td> </tr> <tr> <td>01</td> <td>Isochronous</td> </tr> <tr> <td>10</td> <td>Bulk</td> </tr> <tr> <td>11</td> <td>Interrupt</td> </tr> </tbody> </table>	Value	Description	00	Control	01	Isochronous	10	Bulk	11	Interrupt		
Value	Description															
00	Control															
01	Isochronous															
10	Bulk															
11	Interrupt															

Bit/Field	Name	Type	Reset	Description
3:0	TEP	R/W	0x00	Target Endpoint Number The CPU should set this value to the endpoint number contained in the transmit endpoint descriptor returned to the USB controller during device enumeration.

**Register 75: USB Host Transmit Interval Endpoint 1 (USBTXINTERVAL1), offset 0x11B****Register 76: USB Host Transmit Interval Endpoint 2 (USBTXINTERVAL2), offset 0x12B****Register 77: USB Host Transmit Interval Endpoint 3 (USBTXINTERVAL3), offset 0x13B****Host**

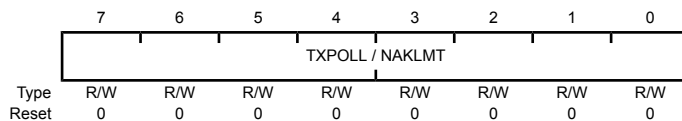
**USBTXINTERVAL<sub>n</sub>** is an 8-bit register that, for interrupt and isochronous transfers, defines the polling interval for the currently selected transmit endpoint. For bulk endpoints, this register sets the number of frames after which the endpoint should time out on receiving a stream of NAK responses.

The **USBTXINTERVAL<sub>n</sub>** register value defines a number of frames, as follows:

Transfer Type	Speed	Valid values (m)	Interpretation
Interrupt	Low-Speed or Full-Speed	1 – 255	Polling interval is $m$ frames.
Isochronous	Full-Speed	1 – 16	Polling interval is $2^{(m-1)}$ frames.
Bulk	Full-Speed	2 – 16	NAK Limit is $2^{(m-1)}$ frames. A value of 0 or 1 disables the NAK timeout function.

**USB Host Transmit Interval Endpoint 1 (USBTXINTERVAL1)**

Base 0x4005.0000  
Offset 0x11B  
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:0	TXPOLL / NAKLMT	R/W	0x00	TX Polling / NAK Limit Polling interval for interrupt/isochronous transfers; NAK limit for bulk transfers.

**Register 78: USB Host Configure Receive Type Endpoint 1 (USBRXTYPE1), offset 0x11C**

**Register 79: USB Host Configure Receive Type Endpoint 2 (USBRXTYPE2), offset 0x12C**

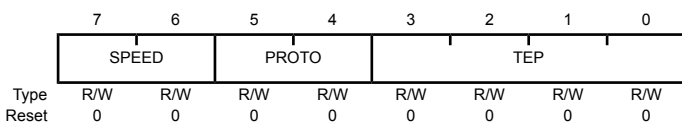
**Register 80: USB Host Configure Receive Type Endpoint 3 (USBRXTYPE3), offset 0x13C**

**Host**

**USBRXTYPE1** is an 8-bit register that should be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently selected receive endpoint, and its operating speed.

USB Host Configure Receive Type Endpoint 1 (USBRXTYPE1)

Base 0x4005.0000  
 Offset 0x11C  
 Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description										
7:6	SPEED	R/W	0x00	<p>Operating Speed</p> <p>Operating speed of the target device when the core is configured with the hub option.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>00</td> <td>Default</td> </tr> <tr> <td>01</td> <td>Reserved</td> </tr> <tr> <td>10</td> <td>Full</td> </tr> <tr> <td>11</td> <td>Low</td> </tr> </tbody> </table> <p>When the core is not configured with the hub option, these bits should not be accessed.</p>	Value	Description	00	Default	01	Reserved	10	Full	11	Low
Value	Description													
00	Default													
01	Reserved													
10	Full													
11	Low													
5:4	PROTO	R/W	0x00	<p>Protocol</p> <p>The CPU should set this to select the required protocol for the receive endpoint:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>00</td> <td>Control</td> </tr> <tr> <td>01</td> <td>Isochronous</td> </tr> <tr> <td>10</td> <td>Bulk</td> </tr> <tr> <td>11</td> <td>Interrupt</td> </tr> </tbody> </table>	Value	Description	00	Control	01	Isochronous	10	Bulk	11	Interrupt
Value	Description													
00	Control													
01	Isochronous													
10	Bulk													
11	Interrupt													

Bit/Field	Name	Type	Reset	Description
3:0	TEP	R/W	0x00	Target Endpoint Number The CPU should set this value to the endpoint number contained in the receive endpoint descriptor returned to the USB controller during device enumeration.

**Register 81: USB Host Receive Polling Interval Endpoint 1 (USBRXINTERVAL1), offset 0x11D**

**Register 82: USB Host Receive Polling Interval Endpoint 2 (USBRXINTERVAL2), offset 0x12D**

**Register 83: USB Host Receive Polling Interval Endpoint 3 (USBRXINTERVAL3), offset 0x13D**

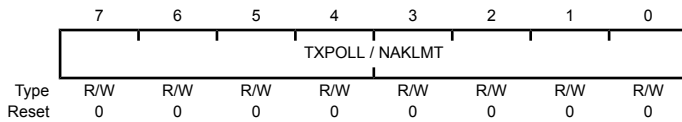
Host

**USBRXINTERVAL1** is an 8-bit register that, for interrupt and isochronous transfers, defines the polling interval for the currently selected receive endpoint. For bulk endpoints, this register sets the number of frames after which the endpoint should time out on receiving a stream of NAK responses. The value that is set defines the number of frames, as follows:

Transfer Type	Speed	Valid Values ( <i>m</i> )	Interpretation
Interrupt	Low-Speed or Full-Speed	1 – 255	Polling interval is <i>m</i> frames.
Isochronous	Full-Speed	1 – 16	Polling interval is $2^{(m-1)}$ frames.
Bulk	Full-Speed	2 – 16	NAK Limit is $2^{(m-1)}$ frames.  <b>Note:</b> A value of 0 or 1 disables the NAK timeout function.

USB Host Receive Polling Interval Endpoint 1 (USBRXINTERVAL1)

Base 0x4005.0000  
Offset 0x11D  
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:0	TXPOLL / NAKLMT	R/W	0x00	RX Polling / NAK Limit  Polling interval for interrupt/isochronous transfers; NAK limit for bulk transfers.

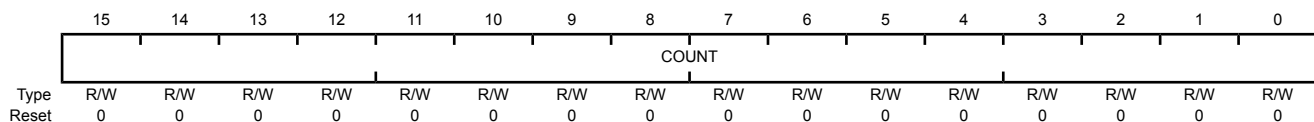
**Register 84: USB Request Packet Count in Block Transfer Endpoint 1 (USBRQPKTCOUNT1), offset 0x304****Register 85: USB Request Packet Count in Block Transfer Endpoint 2 (USBRQPKTCOUNT2), offset 0x308****Register 86: USB Request Packet Count in Block Transfer Endpoint 3 (USBRQPKTCOUNT3), offset 0x30C****Host**

This 16-bit read/write register is used in Host mode to specify the number of packets that are to be transferred in a block transfer of one or more bulk packets to receive endpoint  $n$ . The core uses the value recorded in this register to determine the number of requests to issue where the `AUTORQ` bit in the `USBRXCSRHn` register has been set. See "IN Transactions as a Host" on page 724.

**Note:** Multiple packets combined into a single bulk packet within the FIFO count as one packet.

**USB Request Packet Count in Block Transfer Endpoint 1 (USBRQPKTCOUNT1)**

Base 0x4005.0000  
Offset 0x304  
Type R/W, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:0	COUNT	R/W	0x00	Block Transfer Packet Count

Sets the number of packets of size `MaxP` that are to be transferred in a block transfer.

**Note:** This is only used in Host mode when `AUTORQ` is set. The bit has no effect in Device mode or when `AUTORQ` is not set.

### Register 87: USB Receive Double Packet Buffer Disable (USBRXDPKTBUFDIS), offset 0x340

**Host**

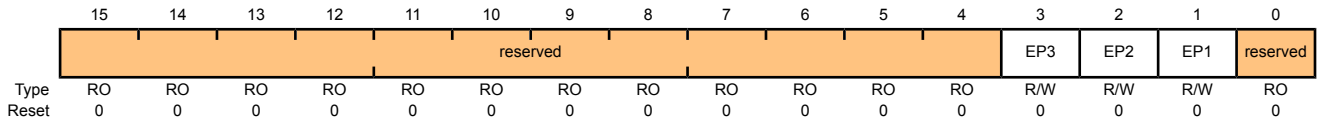
**USBRXDPKTBUFDIS** is a 16-bit register that indicates which of the receive endpoints have disabled the double-packet buffer functionality (see the section called “Double-Packet Buffering” on page 720).

**Device**

**Note:** Bits relating to endpoints that have not been configured may be asserted by writing a 1 to their respective register; however the disable bit will have no observable effect.

#### USB Receive Double Packet Buffer Disable (USBRXDPKTBUFDIS)

Base 0x4005.0000  
 Offset 0x340  
 Type R/W, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:4	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	EP3	R/W	0	EP3 RX Double-Packet Buffer Disable
2	EP2	R/W	0	EP2 RX Double-Packet Buffer Disable
1	EP1	R/W	0	EP1 RX Double-Packet Buffer Disable
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 88: USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFDIS), offset 0x342

**Host**

**USBTXDPKTBUFDIS** is a 16-bit register that indicates which of the transmit endpoints have disabled the double-packet buffer functionality (see the section called “Double-Packet Buffering” on page 719).

**Device**

**Note:** Bits relating to endpoints that have not been configured may be asserted by writing a 1 their respective register; however, the disable bit will have no observable effect.

### USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFDIS)

Base 0x4005.0000

Offset 0x342

Type R/W, reset 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												EP3	EP2	EP1	reserved
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	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
15:4	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	EP3	R/W	0	EP3 TX Double-Packet Buffer Disable
2	EP2	R/W	0	EP2 TX Double-Packet Buffer Disable
1	EP1	R/W	0	EP1 TX Double-Packet Buffer Disable
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 89: USB External Power Control (USBEPIC), offset 0x400

Host

**USBEPIC** is instantiated in a USB unit in a wrapper around the USB controller/PHY IP. This 32-bit register specifies the function of the two-pin external power interface (**USB0EPEN** and **USB0PFLT**). The assertion of the power fault input may generate an automatic action, as controlled by the hardware configuration registers. The automatic action is necessary since the fault condition may require a response faster than one provided by firmware.

Device

OTG

USB External Power Control (USBEPIC)

Base 0x4005.0000

Offset 0x400

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:10	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.
9:8	PFLTACT	R/W	0x00	Power Fault Action Specifies how the <b>USB0EPEN</b> signal is changed when detecting a USB power fault.  Value Description 0x0 Unchanged <b>USB0EPEN</b> is controlled by the combination of the <b>EPEN</b> and <b>EPENDE</b> bits. 0x1 Tristate <b>USB0EPEN</b> is undriven (tristate). 0x2 Low <b>USB0EPEN</b> driven Low. 0x3 High <b>USB0EPEN</b> driven High.
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	PFLTAEN	R/W	0	<p>Power Fault Action Enable</p> <p>Specifies whether a USB power fault triggers any automatic corrective action regarding the driven state of the <code>USB0EPEN</code> signal.</p> <p>Value Description</p> <p>0 Disabled</p> <p><code>USB0EPEN</code> is controlled by the combination of the <code>EPEN</code> and <code>EPENDE</code> bits.</p> <p>1 Enabled</p> <p>The <code>USB0EPEN</code> output is automatically changed to the state as specified in the <code>PFLTACT</code> field.</p>
5	PFLTSEN	R/W	0	<p>Power Fault Sense</p> <p>Specifies the logical sense of the <code>USB0PFLT</code> input signal that indicates an error condition.</p> <p>The complementary state is the inactive state.</p> <p>Value Description</p> <p>0 Low Fault</p> <p>If <code>USB0PFLT</code> is driven Low, the power fault is signaled internally (if enabled).</p> <p>1 High Fault</p> <p>If <code>USB0PFLT</code> is driven High, the power fault is signaled internally (if enabled).</p>
4	PFLTEN	R/W	0	<p>Power Fault Input Enable</p> <p>Specifies whether the <code>USB0PFLT</code> input signal is used in internal logic.</p> <p>Value Description</p> <p>0 Not Used</p> <p>The <code>USB0PFLT</code> signal is ignored.</p> <p>1 Used</p> <p>The <code>USB0PFLT</code> signal is used internally.</p>
3	reserved	RO	0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>

Bit/Field	Name	Type	Reset	Description
2	EPENDE	R/W	0	<p>EPEN Drive Enable</p> <p>Specifies whether the <code>USB0EPEN</code> signal is driven or undriven (tristate). When driven, the signal value is specified by the <code>EPEN</code> bit. When not driven, the <code>EPEN</code> bit is ignored and the <code>USB0EPEN</code> signal is placed in a high-impedance state.</p> <p>Value Description</p> <p>0 Not Driven The <code>USB0EPEN</code> signal is high impedance.</p> <p>1 Driven The <code>USB0EPEN</code> signal is driven to the logical value specified by the <code>EPEN</code> bit value.</p> <p>The <code>USB0EPEN</code> is undriven at reset since the sense of the external power supply enable is unknown. By adding high-impedance state, system designers may bias the power supply enable to the disabled state using a large resistor (100 kΩ) and later configure and drive the output signal to enable the power supply.</p>
1:0	EPEN	R/W	0x00	<p>External Power Supply Enable Configuration</p> <p>Specifies and controls the logical value driven on the <code>USB0EPEN</code> signal.</p> <p>Value Description</p> <p>0x0 Power Enable Active Low The <code>USB0EPEN</code> signal is driven Low if <code>EPENDE</code> is 1.</p> <p>0x1 Power Enable Active High The <code>USB0EPEN</code> signal is driven High if <code>EPENDE</code> is 1.</p> <p>0x2 Power Enable High if VBUS Low The <code>USB0EPEN</code> signal is driven High when the A device is not recognized.</p> <p>0x3 Power Enable High if VBUS High The <code>USB0EPEN</code> signal is driven High when the A device is recognized.</p>

## Register 90: USB External Power Control Raw Interrupt Status (USBEPCRIS), offset 0x404

**Host**

**USBEPCRIS** is instantiated in a USB unit in a wrapper around the USB controller/PHY IP. This 32-bit register specifies the unmasked interrupt status of the two-pin external power interface.

**Device**

USB External Power Control Raw Interrupt Status (USBEPCRIS)

Base 0x4005.0000

Offset 0x404

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															PF
Type	RO	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	PF	RO	0	<p>USB Power Fault Interrupt Status</p> <p>Specifies the unmasked state of the power fault status. This bit is cleared by writing a 1 to the <code>PF</code> bit in the <b>USBEPICISC</b> register.</p> <p>Value Description</p> <table border="0"> <tr> <td>0</td> <td>The hardware has not detected a power fault.</td> </tr> <tr> <td>1</td> <td>The hardware has detected a power fault.</td> </tr> </table>	0	The hardware has not detected a power fault.	1	The hardware has detected a power fault.
0	The hardware has not detected a power fault.							
1	The hardware has detected a power fault.							

### Register 91: USB External Power Control Interrupt Mask (USBEPICM), offset 0x408

**Host**

USBEPICM is instantiated in a USB unit in a wrapper around the USB controller/PHY IP. This 32-bit register specifies the interrupt mask of the two-pin external power interface.

**Device**

USB External Power Control Interrupt Mask (USBEPICM)

Base 0x4005.0000

Offset 0x408

Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															PF
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	PF	R/W	0	USB Power Fault Interrupt Mask Specifies whether a detected power fault generates an interrupt.

Value	Description
0	No Interrupt The hardware does not generate an interrupt on detected power fault.
1	Interrupt The hardware generates an interrupt on detected power fault.

## Register 92: USB External Power Control Interrupt Status and Clear (USBEPICISC), offset 0x40C

Host

**USBEPICISC** is instantiated in a USB unit in a wrapper around the USB controller/PHY IP. This 32-bit register specifies the masked interrupt status of the two-pin external power interface. It also provides a method to clear the interrupt state.

Device

USB External Power Control Interrupt Status and Clear (USBEPICISC)

Base 0x4005.0000

Offset 0x40C

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															PF
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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	PF	R/W1C	0	USB Power Fault Interrupt Status and Clear Specifies whether a detected power fault has generated an interrupt.
---	----	-------	---	--

Value	Description
0	No Interrupt The hardware has not generated an interrupt for a detected power fault condition.
1	Interrupt The hardware has generated an interrupt for a detected power fault condition.

Writing a 1 to this bit clears it and the **USBEPICRIS** PF bit. This bit is set if the **USBEPICRIS** PF bit is set (by hardware) and the **USBEPICIM** PF bit is set.

**Register 93: USB Device Resume Raw Interrupt Status (USBDRRIS), offset 0x410**

Host

The **USBDRRIS** 32-bit 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.

Device

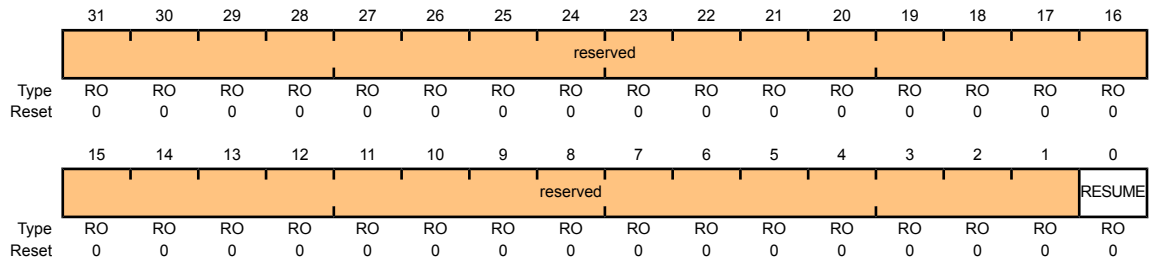
USB Device Resume Raw Interrupt Status (USBDRRIS)

Base 0x4005.0000

Offset 0x410

Type RO, reset 0x0000.0000

OTG



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	RESUME	RO	0	<p>Resume Interrupt Status</p> <p>Specifies the unmasked state of the resume status. This bit is cleared by writing a 1 to the <code>RESUME</code> bit in the <b>USBDRISC</b> register.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The hardware has not detected a Resume.</td> </tr> <tr> <td>1</td> <td>The hardware has detected a Resume.</td> </tr> </tbody> </table>	Value	Description	0	The hardware has not detected a Resume.	1	The hardware has detected a Resume.
Value	Description									
0	The hardware has not detected a Resume.									
1	The hardware has detected a Resume.									



**Register 94: USB Device Resume Interrupt Mask (USBDRIM), offset 0x414****Host**

The **USBDRIM** 32-bit 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.

**Device**

USB Device Resume Interrupt Mask (USBDRIM)

Base 0x4005.0000

Offset 0x414

Type R/W, reset 0x0000.0000

**OTG**

	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															RESUME
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	RESUME	R/W	0	Resume Interrupt Mask Specifies whether a detected Resume generates an interrupt.  Value Description 0 No Interrupt The hardware does not generate an interrupt on detected Resume. 1 Interrupt The hardware generates an interrupt on detected Resume. This should only be enabled when a suspend has been detected (Suspend bit in <b>USBIS</b> register).

### Register 95: USB Device Resume Interrupt Status and Clear (USBDRISC), offset 0x418

**Host**

The **USBDRISC** 32-bit register is the interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

**Device**

USB Device Resume Interrupt Status and Clear (USBDRISC)

Base 0x4005.0000

Offset 0x418

Type W1C, reset 0x0000.0000

**OTG**

	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															RESUME
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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	RESUME	R/W1C	0	<p>Resume Interrupt Status and Clear</p> <p>Specifies whether a detected Resume has generated an interrupt.</p> <p>Value Description</p> <table border="0"> <tr> <td>0</td> <td>No Interrupt</td> </tr> <tr> <td></td> <td>The hardware has not generated an interrupt for a detected Resume.</td> </tr> <tr> <td>1</td> <td>Interrupt</td> </tr> <tr> <td></td> <td>The hardware has generated an interrupt for a detected Resume.</td> </tr> </table> <p>Writing a 1 to this bit clears it and the <b>USBDRRIS</b> RESUME bit. This bit is set if the <b>USBDRRIS</b> RESUME bit is set (by hardware) and the <b>USBEDRIM</b> RESUME bit is set.</p>	0	No Interrupt		The hardware has not generated an interrupt for a detected Resume.	1	Interrupt		The hardware has generated an interrupt for a detected Resume.
0	No Interrupt											
	The hardware has not generated an interrupt for a detected Resume.											
1	Interrupt											
	The hardware has generated an interrupt for a detected Resume.											

**Register 96: USB VBUS Droop Control (USBVDC), offset 0x430****OTG**

This 32-bit register enables a controlled masking of VBUS to compensate for any in-rush current by a device that is connected to the host controller. The in-rush current can cause VBUS to droop, causing the USB controller's behavior to be unexpected. The USB host controller allows VBUS to fall lower than the VBusValid level (4.5 V) but not below AValid (2.0 V) for 65 microseconds without signaling a `VBUSERR` interrupt in the controller. Without this, any glitch on VBUS would force the USB host controller to remove power from VBUS and then re-enumerate the device.

**Host**

## USB VBUS Droop Control (USBVDC)

Base 0x4005.0000  
 Offset 0x430  
 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															VBDEN
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	VBDEN	R/W	0x0	VBUS Droop Enable  When enabled, the VBUS indicator to the controller will mask any changes from VBUSVALID when VBUS goes below 4.5 V but not lower than 2.0 V for 65 microseconds. During this time, the VBUS state will indicate VBUSVALID.

### Register 97: USB VBUS Droop Control Raw Interrupt Status (USBVDCRIS), offset 0x434

**OTG**

This 32-bit register specifies the unmasked interrupt status of the VBUS droop limit of 65 microseconds.

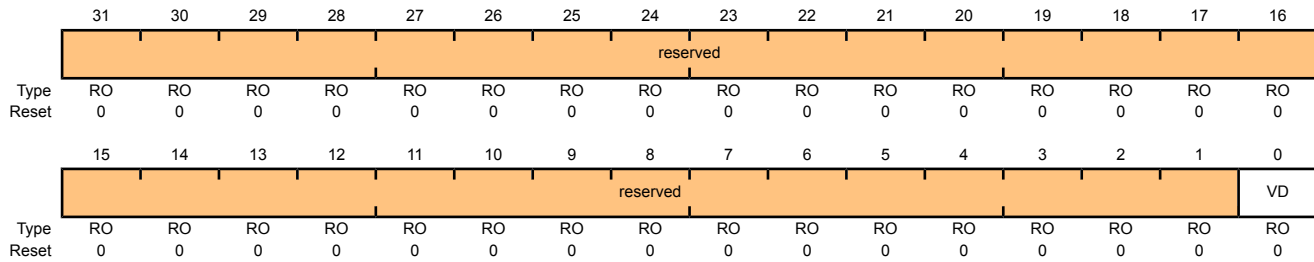
**Host**

USB VBUS Droop Control Raw Interrupt Status (USBVDCRIS)

Base 0x4005.0000

Offset 0x434

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	VD	RO	0	<p>VBUS Droop Raw Interrupt Status</p> <p>Specifies the unmasked state of the VBUS droop status. This bit is cleared by writing a 1 to the <b>VD</b> bit in the <b>USBVDCISC</b> register.</p> <p>Value Description</p> <p>0 The hardware has not detected a VBUS droop for 65 microseconds.</p> <p>1 The hardware has detected a VBUS droop for 65 microseconds.</p>

## Register 98: USB VBUS Droop Control Interrupt Mask (USBVDCIM), offset 0x438

OTG

This 32-bit register specifies the interrupt mask of the VBUS droop.

Host

USB VBUS Droop Control Interrupt Mask (USBVDCIM)

Base 0x4005.0000  
Offset 0x438  
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															VD
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	VD	R/W	0	VBUS Droop Interrupt Mask Specifies whether a detected VBUS droop generates an interrupt.

#### Value Description

0 No Interrupt

The hardware does not generate an interrupt on detected VBUS droop.

1 Interrupt

The hardware generates an interrupt on detected VBUS droop.

### Register 99: USB VBUS Droop Control Interrupt Status and Clear (USBVDCISC), offset 0x43C

**OTG**

This 32-bit register specifies the masked interrupt status of the VBUS droop. It also provides a method to clear the interrupt state.

**Host**

USB VBUS Droop Control Interrupt Status and Clear (USBVDCISC)

Base 0x4005.0000

Offset 0x43C

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															VD
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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	VD	R/W1C	0	VBUS Droop Interrupt Status and Clear Specifies whether a detected VBUS droop has generated an interrupt.

Value	Description
0	No Interrupt The hardware has not generated an interrupt for a detected VBUS droop condition.
1	Interrupt The hardware has generated an interrupt for a detected VBUS droop condition.

Writing a 1 to this bit clears it and the **USBVDCRIS** VD bit. This bit is set if the **USBVDCRIS** VD bit is set (by hardware) and the **USBVDCIM** VD bit is set.

## Register 100: USB ID Valid Detect Raw Interrupt Status (USBIDVRIS), offset 0x444

**OTG**

This 32-bit register specifies whether the unmasked interrupt status of the ID value is valid.

### USB ID Valid Detect Raw Interrupt Status (USBIDVRIS)

Base 0x4005.0000  
 Offset 0x444  
 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															ID
Type	RO	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	ID	RO	0	ID Valid Detect Raw Interrupt Status Specifies the unmasked state of the ID valid detect. This bit is cleared by writing a 1 to the ID bit in the <b>USBIDVISC</b> register.

#### Value Description

Value	Description
0	The hardware has not detected an ID value .
1	The hardware has detected an ID value.

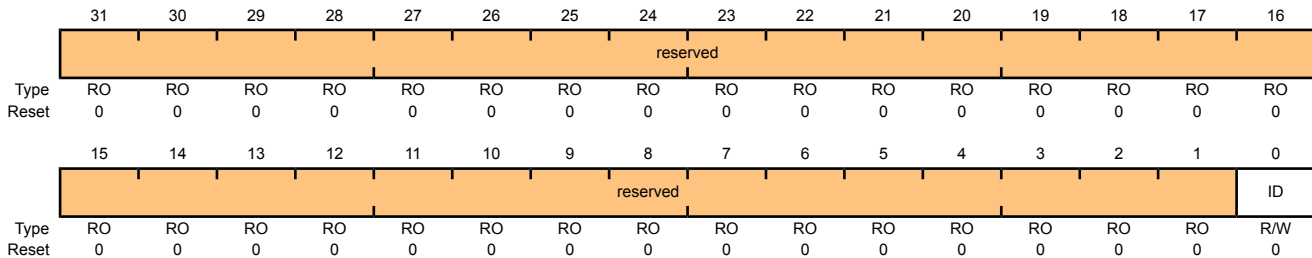
### Register 101: USB ID Valid Detect Interrupt Mask (USBIDVIM), offset 0x448

OTG

This 32-bit register specifies the interrupt mask of the ID valid detection.

USB ID Valid Detect Interrupt Mask (USBIDVIM)

Base 0x4005.0000  
 Offset 0x448  
 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	ID	R/W	0	ID Valid Detect Interrupt Mask

Specifies whether a detected ID valid detect generates an interrupt.

Value	Description
0	No Interrupt The hardware does not generate an interrupt on detected ID valid.
1	Interrupt The hardware generates an interrupt on detected ID valid.



## Register 102: USB ID Valid Detect Interrupt Status and Clear (USBIDVISC), offset 0x44C

OTG

This 32-bit register specifies the masked interrupt status of the ID valid detect. It also provides a method to clear the interrupt state.

### USB ID Valid Detect Interrupt Status and Clear (USBIDVISC)

Base 0x4005.0000

Offset 0x44C

Type R/W1C, reset 0x0000.0000

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

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	ID	R/W1C	0	ID Valid Detect Interrupt Status and Clear Specifies whether a detected ID Valid has generated an interrupt.

#### Value Description

0 No Interrupt

The hardware has not generated an interrupt for a detected ID Valid condition.

1 Interrupt

The hardware has generated an interrupt for a detected ID Valid condition.

Writing a 1 to this bit clears it and the **USBIDVRIS** ID bit. This bit is set if the **USBIDVRIS** ID bit is set (by hardware) and the **USBIDVIM** ID bit is set.

### Register 103: USB End-Point Select (USBEPS), offset 0x450

Host

This 32-bit register specifies which endpoints are mapped to the 6 allocated  $\mu$ DMA channels, see Table 8-1 on page 221 for more information on channel assignments.

Device

USB End-Point Select (USBEPS)

Base 0x4005.0000

Offset 0x450

Type R/W, reset 0x0000.0321

OTG

	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				DMAC				DMAB				DMAA			
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	1	1	0	0	1	0	0	0	0	1

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.																																		
11:8	DMAC	R/W	0x3	<p>DMA C Select</p> <p>Specifies the RX and TX mapping of the third USB endpoint on <math>\mu</math>DMA channels 4 and 5 (primary assignment).</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr><td>0x0</td><td>Endpoint 0 RX/TX</td></tr> <tr><td>0x1</td><td>Endpoint 1 RX/TX</td></tr> <tr><td>0x2</td><td>Endpoint 2 RX/TX</td></tr> <tr><td>0x3</td><td>Endpoint 3 RX/TX</td></tr> <tr><td>0x4</td><td>Endpoint 4 RX/TX</td></tr> <tr><td>0x5</td><td>Endpoint 5 RX/TX</td></tr> <tr><td>0x6</td><td>Endpoint 6 RX/TX</td></tr> <tr><td>0x7</td><td>Endpoint 7 RX/TX</td></tr> <tr><td>0x8</td><td>Endpoint 8 RX/TX</td></tr> <tr><td>0x9</td><td>Endpoint 9 RX/TX</td></tr> <tr><td>0x10</td><td>Endpoint 10 RX/TX</td></tr> <tr><td>0x11</td><td>Endpoint 11 RX/TX</td></tr> <tr><td>0x12</td><td>Endpoint 12 RX/TX</td></tr> <tr><td>0x13</td><td>Endpoint 13 RX/TX</td></tr> <tr><td>0x14</td><td>Endpoint 14 RX/TX</td></tr> <tr><td>0x15</td><td>Endpoint 15 RX/TX</td></tr> </tbody> </table>	Value	Description	0x0	Endpoint 0 RX/TX	0x1	Endpoint 1 RX/TX	0x2	Endpoint 2 RX/TX	0x3	Endpoint 3 RX/TX	0x4	Endpoint 4 RX/TX	0x5	Endpoint 5 RX/TX	0x6	Endpoint 6 RX/TX	0x7	Endpoint 7 RX/TX	0x8	Endpoint 8 RX/TX	0x9	Endpoint 9 RX/TX	0x10	Endpoint 10 RX/TX	0x11	Endpoint 11 RX/TX	0x12	Endpoint 12 RX/TX	0x13	Endpoint 13 RX/TX	0x14	Endpoint 14 RX/TX	0x15	Endpoint 15 RX/TX
Value	Description																																					
0x0	Endpoint 0 RX/TX																																					
0x1	Endpoint 1 RX/TX																																					
0x2	Endpoint 2 RX/TX																																					
0x3	Endpoint 3 RX/TX																																					
0x4	Endpoint 4 RX/TX																																					
0x5	Endpoint 5 RX/TX																																					
0x6	Endpoint 6 RX/TX																																					
0x7	Endpoint 7 RX/TX																																					
0x8	Endpoint 8 RX/TX																																					
0x9	Endpoint 9 RX/TX																																					
0x10	Endpoint 10 RX/TX																																					
0x11	Endpoint 11 RX/TX																																					
0x12	Endpoint 12 RX/TX																																					
0x13	Endpoint 13 RX/TX																																					
0x14	Endpoint 14 RX/TX																																					
0x15	Endpoint 15 RX/TX																																					
7:4	DMAB	R/W	0x2	<p>DMA B Select</p> <p>Specifies the RX and TX mapping of the second USB endpoint on <math>\mu</math>DMA channels 2 and 3 (primary assignment).</p> <p>Same bit definitions as the DMAC field.</p>																																		

Bit/Field	Name	Type	Reset	Description
3:0	DMAA	R/W	0x1	DMA A Select Specifies the RX and TX mapping of the first USB endpoint on $\mu$ DMA channels 0 and 1 (primary assignment). Same bit definitions as the $DMAC$ field.

## 20 Analog Comparators

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

**Note:** Not all comparators have the option to drive an output pin.

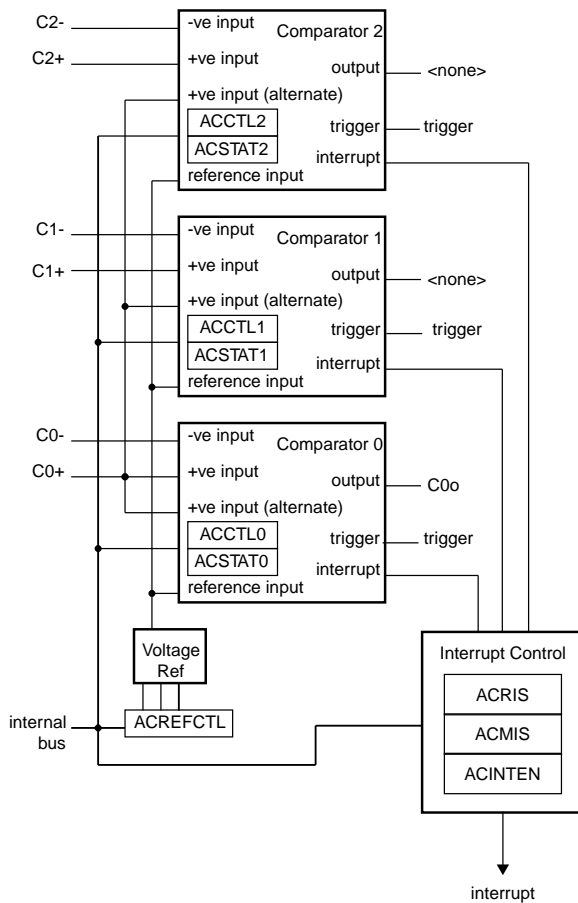
The comparator can provide its output to a device pin, acting as a replacement for an analog comparator on the board, or it can be used to signal the application via interrupts or triggers to the ADC to cause it to start capturing a sample sequence. The interrupt generation and ADC triggering logic is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

The Stellaris<sup>®</sup> Analog Comparators module has the following features:

- Compare external pin input to external pin input or to internal programmable voltage reference
- Compare a test voltage against any one of these voltages
  - An individual external reference voltage
  - A shared single external reference voltage
  - A shared internal reference voltage

## 20.1 Block Diagram

Figure 20-1. Analog Comparator Module Block Diagram



## 20.2 Functional Description

**Important:** It is recommended that the Digital-Input enable (the `GPIOEN` bit in the GPIO module) for the analog input pin be disabled to prevent excessive current draw from the I/O pads.

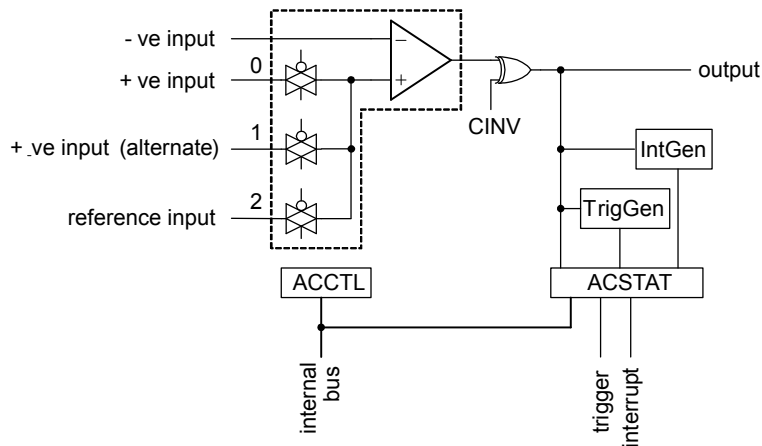
The comparator compares the  $V_{IN-}$  and  $V_{IN+}$  inputs to produce an output,  $V_{OUT}$ .

$$V_{IN-} < V_{IN+}, V_{OUT} = 1$$

$$V_{IN-} > V_{IN+}, V_{OUT} = 0$$

As shown in Figure 20-2 on page 818, the input source for  $V_{IN-}$  is an external input. In addition to an external input, input sources for  $V_{IN+}$  can be the +ve input of comparator 0 or an internal reference.

Figure 20-2. 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**).

Typically, the comparator output is used internally to generate controller interrupts. It may also be used to drive an external pin or generate an analog-to-digital converter (ADC) trigger.

**Important:** The **ASRCP** bits in the **ACCTLn** register must be set before using the analog comparators.

### 20.2.1 Internal Reference Programming

The structure of the internal reference is shown in Figure 20-3 on page 818. This is controlled by a single configuration register (**ACREFCTL**). Table 20-1 on page 818 shows the programming options to develop specific internal reference values, to compare an external voltage against a particular voltage generated internally ( $V_{IREF}$ ).

Figure 20-3. Comparator Internal Reference Structure

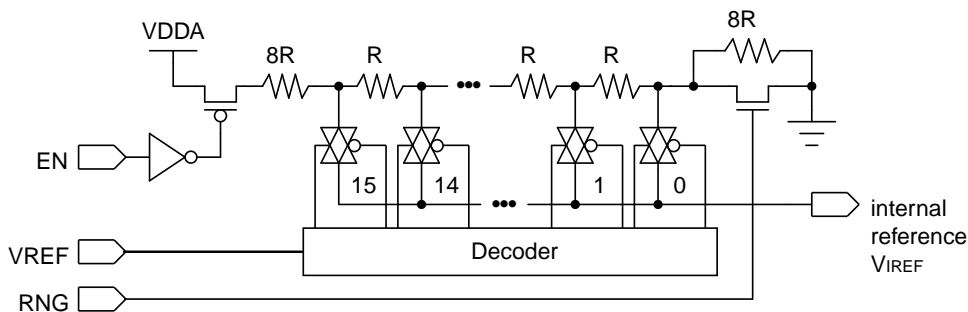


Table 20-1. Internal Reference Voltage and **ACREFCTL** Field Values

<b>ACREFCTL</b> Register		Output Reference Voltage Based on <b>VREF</b> Field Value
<b>EN</b> Bit Value	<b>RNG</b> Bit Value	
EN=0	RNG=X	0 V (GND) for any value of <b>VREF</b> ; however, it is recommended that RNG=1 and <b>VREF</b> =0 for the least noisy ground reference.

ACREFCTL Register		Output Reference Voltage Based on VREF Field Value
EN Bit Value	RNG Bit Value	
EN=1	RNG=0	<p>Total resistance in ladder is 31 R.</p> $V_{IREF} = V_{DDA} \times \frac{R_{VREF}}{R_T}$ $V_{IREF} = V_{DDA} \times \frac{(VREF + 8)}{31}$ $V_{IREF} = 0.85 + 0.106 \times VREF$ <p>The range of internal reference in this mode is 0.85-2.448 V.</p>
	RNG=1	<p>Total resistance in ladder is 23 R.</p> $V_{IREF} = V_{DDA} \times \frac{R_{VREF}}{R_T}$ $V_{IREF} = V_{DDA} \times \frac{VREF}{23}$ $V_{IREF} = 0.143 \times VREF$ <p>The range of internal reference for this mode is 0-2.152 V.</p>

## 20.3 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. See page 163.
2. In the GPIO module, enable the GPIO port/pin associated with C0- as a GPIO input.
3. Configure the internal voltage reference to 1.65 V by writing the **ACREFCTL** register with the value 0x0000.030C.
4. Configure comparator 0 to use the internal voltage reference and to *not* invert the output by writing the **ACCTL0** register with the value 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.

## 20.4 Register Map

Table 20-2 on page 820 lists the comparator registers. The offset listed is a hexadecimal increment to the register's address, relative to the Analog Comparator base address of 0x4003.C000. Note that the analog comparator clock must be enabled before the registers can be programmed (see page 163).

**Table 20-2. Analog Comparators Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	ACMIS	R/W1C	0x0000.0000	Analog Comparator Masked Interrupt Status	821
0x004	ACRIS	RO	0x0000.0000	Analog Comparator Raw Interrupt Status	822
0x008	ACINTEN	R/W	0x0000.0000	Analog Comparator Interrupt Enable	823
0x010	ACREFCTL	R/W	0x0000.0000	Analog Comparator Reference Voltage Control	824
0x020	ACSTAT0	RO	0x0000.0000	Analog Comparator Status 0	825
0x024	ACCTL0	R/W	0x0000.0000	Analog Comparator Control 0	826
0x040	ACSTAT1	RO	0x0000.0000	Analog Comparator Status 1	825
0x044	ACCTL1	R/W	0x0000.0000	Analog Comparator Control 1	826
0x060	ACSTAT2	RO	0x0000.0000	Analog Comparator Status 2	825
0x064	ACCTL2	R/W	0x0000.0000	Analog Comparator Control 2	826

## 20.5 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 0x000**

This register provides a summary of the interrupt status (masked) of the comparator.

**Analog Comparator Masked Interrupt Status (ACMIS)**

Base 0x4003.C000

Offset 0x000

Type R/W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	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													IN2	IN1	IN0	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	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	IN2	R/W1C	0	Comparator 2 Masked Interrupt Status Gives the masked interrupt state of this interrupt. Write 1 to this bit to clear the pending interrupt.
1	IN1	R/W1C	0	Comparator 1 Masked Interrupt Status Gives the masked interrupt state of this interrupt. Write 1 to this bit to clear the pending interrupt.
0	IN0	R/W1C	0	Comparator 0 Masked Interrupt Status Gives the masked interrupt state of this interrupt. Write 1 to this bit to clear the pending interrupt.

## Register 2: Analog Comparator Raw Interrupt Status (ACRIS), offset 0x004

This register provides a summary of the interrupt status (raw) of the comparator.

### Analog Comparator Raw Interrupt Status (ACRIS)

Base 0x4003.C000

Offset 0x004

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													IN2	IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	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	IN2	RO	0	Comparator 2 Interrupt Status When set, indicates that an interrupt has been generated by comparator 2.
1	IN1	RO	0	Comparator 1 Interrupt Status When set, indicates that an interrupt has been generated by comparator 1.
0	IN0	RO	0	Comparator 0 Interrupt Status When set, indicates that an interrupt has been generated by comparator 0.

### Register 3: Analog Comparator Interrupt Enable (ACINTEN), offset 0x008

This register provides the interrupt enable for the comparator.

#### Analog Comparator Interrupt Enable (ACINTEN)

Base 0x4003.C000

Offset 0x008

Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	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													IN2	IN1	IN0	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	IN2	R/W	0	Comparator 2 Interrupt Enable When set, enables the controller interrupt from the comparator 2 output
1	IN1	R/W	0	Comparator 1 Interrupt Enable When set, enables the controller interrupt from the comparator 1 output.
0	IN0	R/W	0	Comparator 0 Interrupt Enable When set, enables the controller interrupt from the comparator 0 output.

## Register 4: Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x010

This register specifies whether the resistor ladder is powered on as well as the range and tap.

### Analog Comparator Reference Voltage Control (ACREFCTL)

Base 0x4003.C000

Offset 0x010

Type R/W, reset 0x0000.0000

	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						EN	RNG	reserved				VREF			
Type	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	EN	R/W	0	Resistor Ladder Enable  The <b>EN</b> bit specifies whether the resistor ladder is powered on. If 0, the resistor ladder is unpowered. If 1, the resistor ladder is connected to the analog $V_{DD}$ .  This bit is reset to 0 so that the internal reference consumes the least amount of power if not used and programmed.
8	RNG	R/W	0	Resistor Ladder Range  The <b>RNG</b> bit specifies the range of the resistor ladder. If 0, the resistor ladder has a total resistance of 31 R. If 1, the resistor ladder has a total resistance of 23 R.
7:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	VREF	R/W	0x00	Resistor Ladder Voltage Ref  The <b>VREF</b> bit field specifies the resistor ladder tap that is passed through an analog multiplexer. The voltage corresponding to the tap position is the internal reference voltage available for comparison. See Table 20-1 on page 818 for some output reference voltage examples.

**Register 5: Analog Comparator Status 0 (ACSTAT0), offset 0x020****Register 6: Analog Comparator Status 1 (ACSTAT1), offset 0x040****Register 7: Analog Comparator Status 2 (ACSTAT2), offset 0x060**

These registers specify the current output value of the comparator.

**Analog Comparator Status 0 (ACSTAT0)**

Base 0x4003.C000

Offset 0x020

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x00	Software should not rely on the value of 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 8: Analog Comparator Control 0 (ACCTL0), offset 0x024**

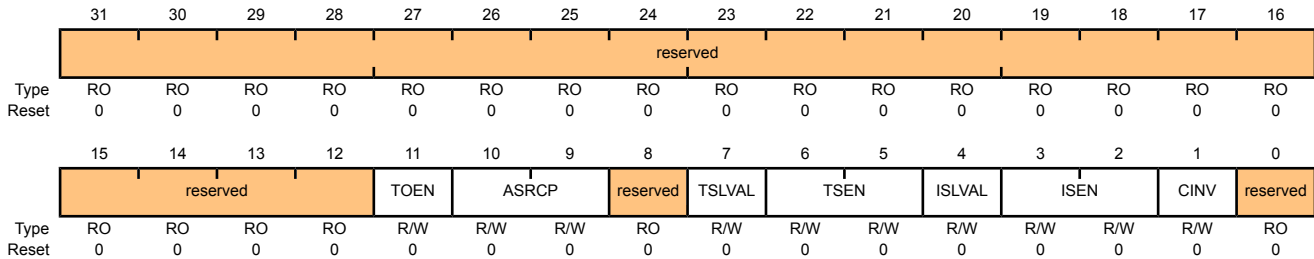
**Register 9: Analog Comparator Control 1 (ACCTL1), offset 0x044**

**Register 10: Analog Comparator Control 2 (ACCTL2), offset 0x064**

These registers configure the comparator's input and output.

Analog Comparator Control 0 (ACCTL0)

Base 0x4003.C000  
 Offset 0x024  
 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	TOEN	R/W	0	<p>Trigger Output Enable</p> <p>The <b>TOEN</b> 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.</p>										
10:9	ASRCP	R/W	0x00	<p>Analog Source Positive</p> <p>The <b>ASRCP</b> field specifies the source of input voltage to the VIN+ terminal of the comparator. The encodings for this field are as follows:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Function</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Pin value</td> </tr> <tr> <td>0x1</td> <td>Pin value of C0+</td> </tr> <tr> <td>0x2</td> <td>Internal voltage reference</td> </tr> <tr> <td>0x3</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Function	0x0	Pin value	0x1	Pin value of C0+	0x2	Internal voltage reference	0x3	Reserved
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	<p>Trigger Sense Level Value</p> <p>The <b>TSLVAL</b> 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.</p>										

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 border="1"> <thead> <tr> <th>Value</th> <th>Function</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Level sense, see TSLVAL</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 TSLVAL	0x1	Falling edge	0x2	Rising edge	0x3	Either edge
Value	Function													
0x0	Level sense, see TSLVAL													
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 border="1"> <thead> <tr> <th>Value</th> <th>Function</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Level sense, see ISLVAL</td> </tr> <tr> <td>0x1</td> <td>Falling edge</td> </tr> <tr> <td>0x2</td> <td>Rising edge</td> </tr> <tr> <td>0x3</td> <td>Either edge</td> </tr> </tbody> </table>	Value	Function	0x0	Level sense, see ISLVAL	0x1	Falling edge	0x2	Rising edge	0x3	Either edge
Value	Function													
0x0	Level sense, see ISLVAL													
0x1	Falling edge													
0x2	Rising edge													
0x3	Either edge													
1	CINV	R/W	0	<p>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	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>										

## 21 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<sup>®</sup> PWM module consists of four PWM generator blocks and a control block. The control block determines the polarity of the PWM signals, and which signals are passed through to the pins.

Each PWM generator block produces two PWM signals that can either be independent signals (other than being based on the same timer and therefore having the same frequency) or a single pair of complementary signals with dead-band delays inserted. The output of the PWM generation blocks are managed by the output control block before being passed to the device pins.

The Stellaris<sup>®</sup> PWM module provides a great deal of flexibility. It can generate simple PWM signals, such as those required by a simple charge pump. It can also generate paired PWM signals with dead-band delays, such as those required by a half-H bridge driver. Three generator blocks can also generate the full six channels of gate controls required by a 3-phase inverter bridge.

Each Stellaris<sup>®</sup> PWM module has the following features:

- 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 signal 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
- Can initiate an ADC sample sequence



The control block determines the polarity of the PWM signals and which signals are passed through to the pins. The output of the PWM generation blocks are managed by the output control block before being passed to the device pins. The PWM control block has the following options:

- 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
- Extended fault capabilities with multiple fault signals, programmable polarities, and filtering
- PWM generators can be operated independently or synchronized with other generators

## 21.1 Block Diagram

Figure 21-1 on page 829 provides the Stellaris<sup>®</sup> PWM module unit diagram and Figure 21-2 on page 830 provides a more detailed diagram of a Stellaris<sup>®</sup> PWM generator. The LM3S5791 controller contains four generator blocks (PWM0, PWM1, PWM2, and PWM3) and generates eight independent PWM signals or four paired PWM signals with dead-band delays inserted.

**Figure 21-1. PWM Unit Diagram**

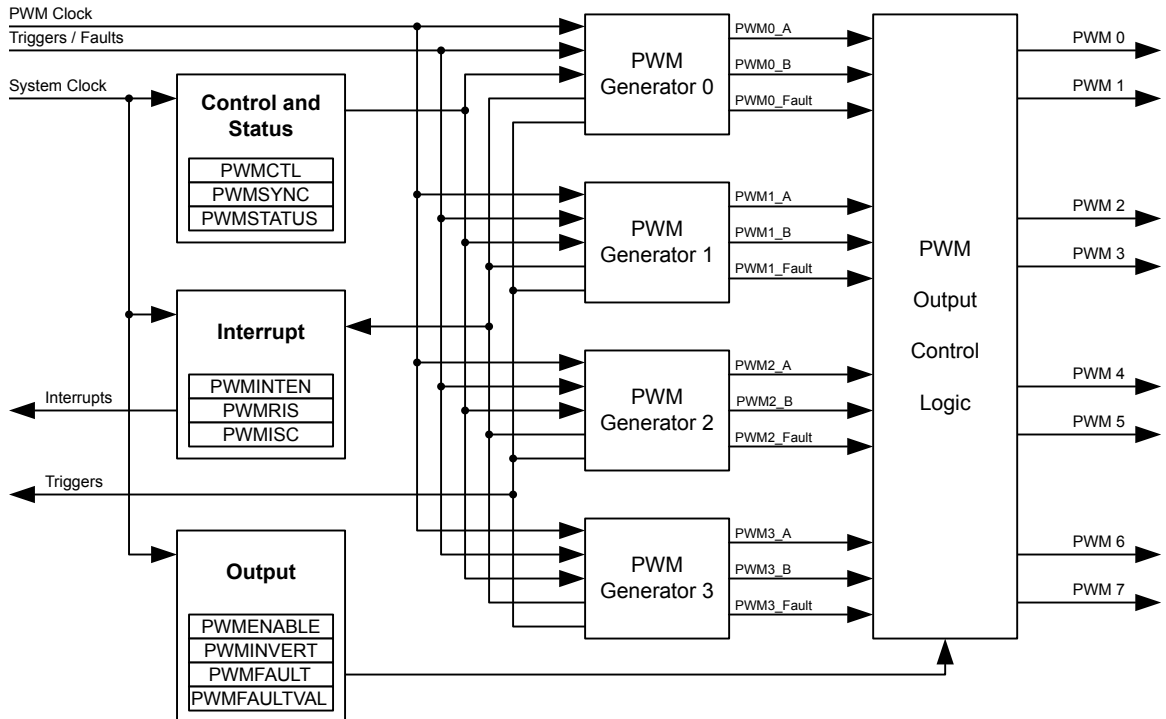
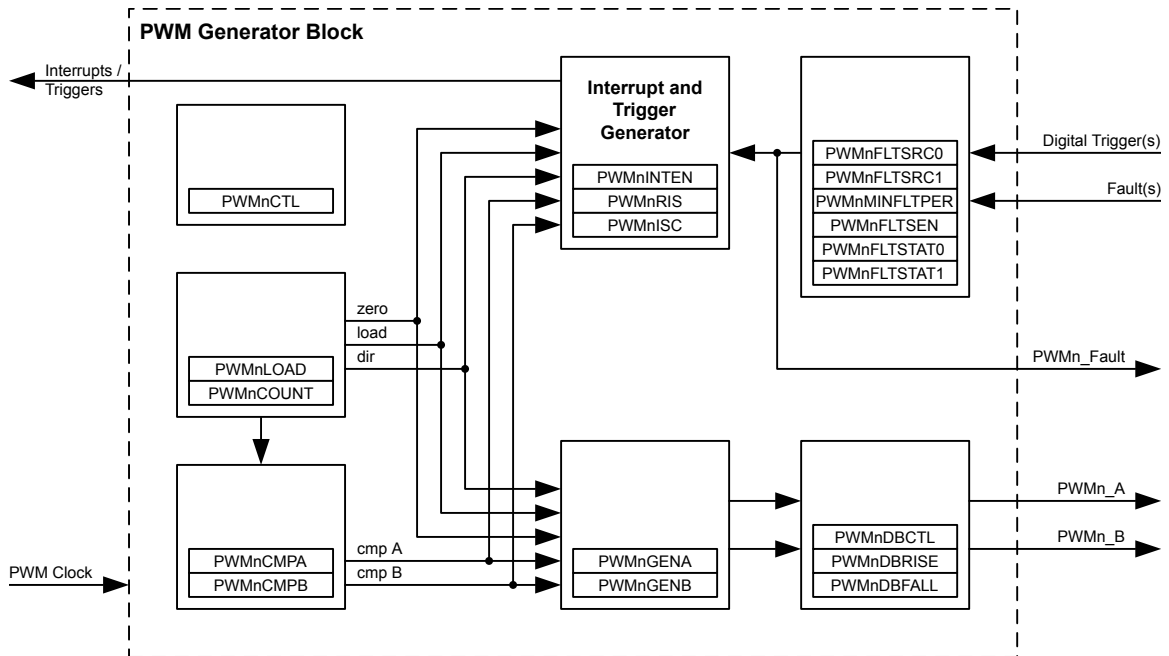


Figure 21-2. PWM Module Block Diagram



## 21.2 Functional Description

### 21.2.1 PWM Timer

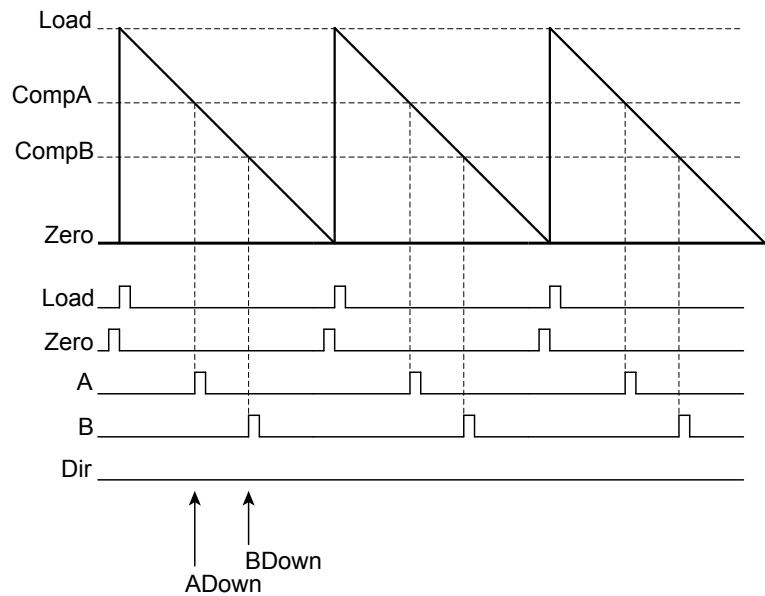
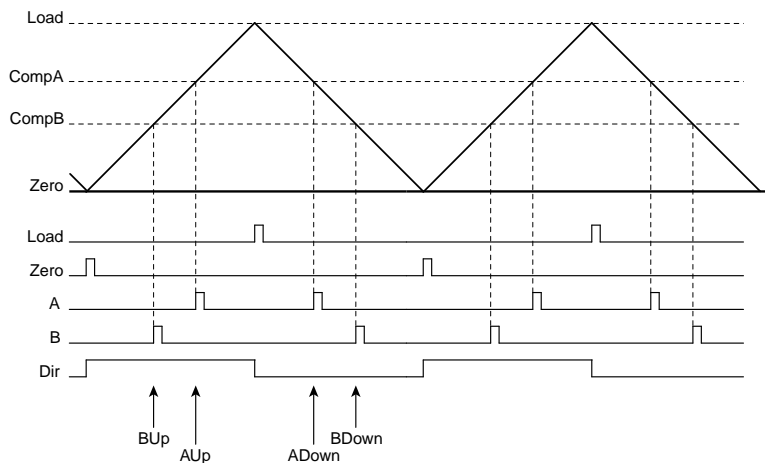
The timer in each PWM generator runs in one of two modes: Count-Down mode or Count-Up/Down mode. In Count-Down mode, the timer counts from the load value to zero, goes back to the load value, and continues counting down. In Count-Up/Down mode, the timer counts from zero up to the load value, back down 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.

### 21.2.2 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.

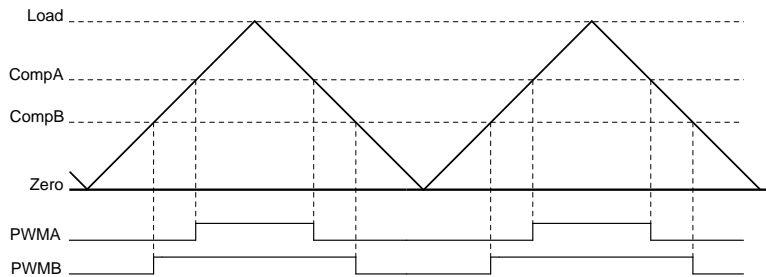
Figure 21-3 on page 831 shows the behavior of the counter and the relationship of these pulses when the counter is in Count-Down mode. Figure 21-4 on page 831 shows the behavior of the counter and the relationship of these pulses when the counter is in Count-Up/Down mode.

**Figure 21-3. PWM Count-Down Mode****Figure 21-4. PWM Count-Up/Down Mode**

### 21.2.3 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 21-5 on page 832 shows the use of Count-Up/Down mode to generate a pair of center-aligned, overlapped PWM signals that have different duty cycles.

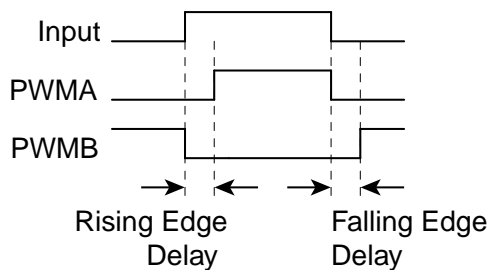
**Figure 21-5. PWM Generation Example In Count-Up/Down Mode**

In this example, the first generator is set to drive High on match A up, drive Low on match A down, and ignore the other four events. The second generator is set to drive High on match B up, drive Low on match B down, and ignore the other four events. Changing the value of comparator A changes the duty cycle of the `PWMA` signal, and changing the value of comparator B changes the duty cycle of the `PWMB` signal.

#### 21.2.4 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 21-6 on page 832 shows the effect of the dead-band generator on an input PWM signal.

**Figure 21-6. PWM Dead-Band Generator**

#### 21.2.5 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.

## 21.2.6 Synchronization Methods

The PWM unit provides four PWM generators providing eight PWM outputs that may be used in a wide variety of applications. Generally speaking, this falls into combinations of two categories of operation:

- **Unsynchronized.** The PWM generator and its two output signals are used by itself, independent of other PWM generators.
- **Synchronized.** The PWM generator and its two outputs signals are used in conjunction with other PWM generators using a common, unified time base.

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. Other states in the unit provide mechanisms to maintain the common time base and mutual synchronization.

The counter in a PWM unit generator can be reset to zero by writing the **PWM Time Base Sync (PWMSYNC)** register and setting the `Sync` bit associated with the generator. Multiple PWM generators can be synchronized together by setting all necessary `Sync` bits in one access. For example, setting the `Sync0` and `Sync1` bits in the **PWMSYNC** register causes the counters in PWM generators 0 and 1 to reset together.

Additionally, the state of a PWM unit is affected by writing to the registers of the PWM unit and the PWM units' generators, which has an effect on the synchronization between multiple PWM generators. Depending on the register accessed, the register state is updated in one of the following three ways:

- **Immediately.** The write value has immediate effect, and the hardware reacts immediately.
- **Locally Synchronized.** The write value does not affect the logic until the counter reaches the value zero. In this case, the effect of the write is deferred until the end of the PWM cycle (when 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.
- **Globally Synchronized.** The write value does not affect the logic until two sequential events have occurred: (1) the global synchronization bit applicable to the generator is set, and (2) the counter reaches zero. In this case, the effect of the write is deferred until the end of the PWM cycle (when the counter reaches zero) following the end of all updates. This 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, although this is not required in order for this mechanism to function properly.

The following registers provide either local or global synchronization based on the state of the **PWMnCTL** register `Update` bit value:

- Generator Registers: **PWMnLOAD**, **PWMnCMPA**, and **PWMnCMPB**

The following registers are provided with the optional functionality of synchronously updating rather than having all updates take immediate effect. The default update mode is immediate.

- Module-Level Register: **PWMENABLE**
- Generator Register: **PWMnGENA**, **PWMnGENB**, **PWMnDBCTL**, **PWMnDBRISE**, and **PWMnDBFALL**.

All other registers are considered statically provisioned for the execution of an application or are used dynamically for purposes unrelated to maintaining synchronization, and therefore, do not need synchronous update functionality.

### 21.2.7 Fault Conditions

A fault condition is one in which the controller must be signaled to stop normal PWM function and then sets the outputs to a safe state. There are two basic situations where this becomes necessary:

- The controller is stalled and cannot perform the necessary computation in the time required for motion control
- An external error or event is detected, such as an error

The PWM unit can use the following inputs to generate a fault condition, including:

- **FAULT<sub>n</sub>** pin assertion
- A stall of the controller generated by the debugger
- The trigger of an ADC digital comparator

Fault conditions are calculated on a per-PWM generator basis. Each PWM generator configures the necessary conditions to indicate a fault condition exists. This method allows the development of applications with dependent and independent control.

Each PWM generator's mode control, including fault condition handling, is provided in the **PWMnCTL** register. This register determines whether a single **FAULT0** input is used (as previous Stellaris<sup>®</sup> products support) or whether all **FAULT<sub>n</sub>** input signals may be used to generate a fault condition. This register allows the fault condition duration to last as long as the external condition lasts, or it may specify that the external condition be latched and the fault condition (and its effects) last until cleared by software. Finally, this register also enables a counter that may be used to extend the period of a fault condition for external events to assure that the duration is a minimum length. The minimum fault period count is specified in the **PWMnMINFLTPER** register.

These PWM generator registers provide status, control, and configure the fault condition in each PWM generator: **PWMnFLTSRC0**, **PWMnFLTSRC1**, **PWMnFLTSTAT0**, **PWMnFLTSTAT1**, and **PWMnFLTSEN**.

There are up to four **FAULT** input pins (**FAULT0-FAULT3**). These pins may be used with circuits that generate an active High or active Low signal to indicate an error condition. Each of the **FAULT<sub>n</sub>** pins may be individually programmed for this logic sense using the **PWMnFLTSEN** register.

The **PWMnFLTSRC0** and **PWMnFLTSRC1** registers define the contribution of the external fault sources. Using these registers, individual or groups of **FAULT<sub>n</sub>** signals are ORed together to specify the external fault generating conditions.

Status regarding the specific fault cause is provided in **PWMnFLTSTAT0** and **PWMnFLTSTAT1**.

PWM generator fault conditions may be promoted to a controller interrupt using the **PWMINTEN** register.

During fault conditions, the PWM output signals usually require being driven to safe values so that external equipment may be safely controlled. To facilitate this, the **PWMFAULT** register is used to determine if the generated signal continues to be passed driven, or a specific fault condition encoding is driven on the PWM output, as specified in the **PWMFAULTVAL** register.

### 21.2.8 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.

## 21.3 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. See page 154.
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module. See page 175.
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.

## 21.4 Register Map

Table 21-1 on page 836 lists the PWM registers. The offset listed is a hexadecimal increment to the register's address, relative to the PWM base address of 0x4002.8000. Note that the PWM module clock must be enabled before the registers can be programmed (see page 154).

**Table 21-1. PWM Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	PWM0CTL	R/W	0x0000.0000	PWM Master Control	839
0x004	PWMSYNC	R/W	0x0000.0000	PWM Time Base Sync	840
0x008	PWMENABLE	R/W	0x0000.0000	PWM Output Enable	841
0x00C	PWMINVERT	R/W	0x0000.0000	PWM Output Inversion	843
0x010	PWMFAULT	R/W	0x0000.0000	PWM Output Fault	844
0x014	PWMINTEN	R/W	0x0000.0000	PWM Interrupt Enable	846
0x018	PWMRIS	RO	0x0000.0000	PWM Raw Interrupt Status	848
0x01C	PWMISC	R/W1C	0x0000.0000	PWM Interrupt Status and Clear	850
0x020	PWMSTATUS	RO	0x0000.0000	PWM Status	852
0x024	PWMFAULTVAL	R/W	0x0000.0000	PWM Fault Condition Value	853
0x040	PWM0CTL	R/W	0x0000.0000	PWM0 Control	855
0x044	PWM0INTEN	R/W	0x0000.0000	PWM0 Interrupt and Trigger Enable	860
0x048	PWM0RIS	RO	0x0000.0000	PWM0 Raw Interrupt Status	862
0x04C	PWM0ISC	R/W1C	0x0000.0000	PWM0 Interrupt Status and Clear	863
0x050	PWM0LOAD	R/W	0x0000.0000	PWM0 Load	864
0x054	PWM0COUNT	RO	0x0000.0000	PWM0 Counter	865
0x058	PWM0CMPA	R/W	0x0000.0000	PWM0 Compare A	866
0x05C	PWM0CMPB	R/W	0x0000.0000	PWM0 Compare B	867
0x060	PWM0GENA	R/W	0x0000.0000	PWM0 Generator A Control	868
0x064	PWM0GENB	R/W	0x0000.0000	PWM0 Generator B Control	871
0x068	PWM0DBCTL	R/W	0x0000.0000	PWM0 Dead-Band Control	874
0x06C	PWM0DBRISE	R/W	0x0000.0000	PWM0 Dead-Band Rising-Edge Delay	875
0x070	PWM0DBFALL	R/W	0x0000.0000	PWM0 Dead-Band Falling-Edge-Delay	876



Offset	Name	Type	Reset	Description	See page
0x074	PWM0FLTSRC0	R/W	0x0000.0000	PWM0 Fault Source 0	877
0x078	PWM0FLTSRC1	R/W	0x0000.0000	PWM0 Fault Source 1	879
0x07C	PWM0MINFLTPER	R/W	0x0000.0000	PWM0 Minimum Fault Period	881
0x080	PWM1CTL	R/W	0x0000.0000	PWM1 Control	855
0x084	PWM1INTEN	R/W	0x0000.0000	PWM1 Interrupt and Trigger Enable	860
0x088	PWM1RIS	RO	0x0000.0000	PWM1 Raw Interrupt Status	862
0x08C	PWM1ISC	R/W1C	0x0000.0000	PWM1 Interrupt Status and Clear	863
0x090	PWM1LOAD	R/W	0x0000.0000	PWM1 Load	864
0x094	PWM1COUNT	RO	0x0000.0000	PWM1 Counter	865
0x098	PWM1CMPA	R/W	0x0000.0000	PWM1 Compare A	866
0x09C	PWM1CMPB	R/W	0x0000.0000	PWM1 Compare B	867
0x0A0	PWM1GENA	R/W	0x0000.0000	PWM1 Generator A Control	868
0x0A4	PWM1GENB	R/W	0x0000.0000	PWM1 Generator B Control	871
0x0A8	PWM1DBCTL	R/W	0x0000.0000	PWM1 Dead-Band Control	874
0x0AC	PWM1DBRISE	R/W	0x0000.0000	PWM1 Dead-Band Rising-Edge Delay	875
0x0B0	PWM1DBFALL	R/W	0x0000.0000	PWM1 Dead-Band Falling-Edge-Delay	876
0x0B4	PWM1FLTSRC0	R/W	0x0000.0000	PWM1 Fault Source 0	877
0x0B8	PWM1FLTSRC1	R/W	0x0000.0000	PWM1 Fault Source 1	879
0x0BC	PWM1MINFLTPER	R/W	0x0000.0000	PWM1 Minimum Fault Period	881
0x0C0	PWM2CTL	R/W	0x0000.0000	PWM2 Control	855
0x0C4	PWM2INTEN	R/W	0x0000.0000	PWM2 Interrupt and Trigger Enable	860
0x0C8	PWM2RIS	RO	0x0000.0000	PWM2 Raw Interrupt Status	862
0x0CC	PWM2ISC	R/W1C	0x0000.0000	PWM2 Interrupt Status and Clear	863
0x0D0	PWM2LOAD	R/W	0x0000.0000	PWM2 Load	864
0x0D4	PWM2COUNT	RO	0x0000.0000	PWM2 Counter	865
0x0D8	PWM2CMPA	R/W	0x0000.0000	PWM2 Compare A	866
0x0DC	PWM2CMPB	R/W	0x0000.0000	PWM2 Compare B	867
0x0E0	PWM2GENA	R/W	0x0000.0000	PWM2 Generator A Control	868
0x0E4	PWM2GENB	R/W	0x0000.0000	PWM2 Generator B Control	871
0x0E8	PWM2DBCTL	R/W	0x0000.0000	PWM2 Dead-Band Control	874
0x0EC	PWM2DBRISE	R/W	0x0000.0000	PWM2 Dead-Band Rising-Edge Delay	875
0x0F0	PWM2DBFALL	R/W	0x0000.0000	PWM2 Dead-Band Falling-Edge-Delay	876
0x0F4	PWM2FLTSRC0	R/W	0x0000.0000	PWM2 Fault Source 0	877

Offset	Name	Type	Reset	Description	See page
0x0F8	PWM2FLTSRC1	R/W	0x0000.0000	PWM2 Fault Source 1	879
0x0FC	PWM2MINFLTPER	R/W	0x0000.0000	PWM2 Minimum Fault Period	881
0x100	PWM3CTL	R/W	0x0000.0000	PWM3 Control	855
0x104	PWM3INTEN	R/W	0x0000.0000	PWM3 Interrupt and Trigger Enable	860
0x108	PWM3RIS	RO	0x0000.0000	PWM3 Raw Interrupt Status	862
0x10C	PWM3ISC	R/W1C	0x0000.0000	PWM3 Interrupt Status and Clear	863
0x110	PWM3LOAD	R/W	0x0000.0000	PWM3 Load	864
0x114	PWM3COUNT	RO	0x0000.0000	PWM3 Counter	865
0x118	PWM3CMPA	R/W	0x0000.0000	PWM3 Compare A	866
0x11C	PWM3CMPB	R/W	0x0000.0000	PWM3 Compare B	867
0x120	PWM3GENA	R/W	0x0000.0000	PWM3 Generator A Control	868
0x124	PWM3GENB	R/W	0x0000.0000	PWM3 Generator B Control	871
0x128	PWM3DBCTL	R/W	0x0000.0000	PWM3 Dead-Band Control	874
0x12C	PWM3DBRISE	R/W	0x0000.0000	PWM3 Dead-Band Rising-Edge Delay	875
0x130	PWM3DBFALL	R/W	0x0000.0000	PWM3 Dead-Band Falling-Edge-Delay	876
0x134	PWM3FLTSRC0	R/W	0x0000.0000	PWM3 Fault Source 0	877
0x138	PWM3FLTSRC1	R/W	0x0000.0000	PWM3 Fault Source 1	879
0x13C	PWM3MINFLTPER	R/W	0x0000.0000	PWM3 Minimum Fault Period	881
0x800	PWM0FLTSEN	R/W	0x0000.0000	PWM0 Fault Pin Logic Sense	882
0x804	PWM0FLTSTAT0	-	0x0000.0000	PWM0 Fault Status 0	883
0x808	PWM0FLTSTAT1	-	0x0000.0000	PWM0 Fault Status 1	885
0x880	PWM1FLTSEN	R/W	0x0000.0000	PWM1 Fault Pin Logic Sense	882
0x884	PWM1FLTSTAT0	-	0x0000.0000	PWM1 Fault Status 0	883
0x888	PWM1FLTSTAT1	-	0x0000.0000	PWM1 Fault Status 1	885
0x900	PWM2FLTSEN	R/W	0x0000.0000	PWM2 Fault Pin Logic Sense	882
0x904	PWM2FLTSTAT0	-	0x0000.0000	PWM2 Fault Status 0	883
0x908	PWM2FLTSTAT1	-	0x0000.0000	PWM2 Fault Status 1	885
0x980	PWM3FLTSEN	R/W	0x0000.0000	PWM3 Fault Pin Logic Sense	882
0x984	PWM3FLTSTAT0	-	0x0000.0000	PWM3 Fault Status 0	883
0x988	PWM3FLTSTAT1	-	0x0000.0000	PWM3 Fault Status 1	885

## 21.5 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

	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												GlobalSync3	GlobalSync2	GlobalSync1	GlobalSync0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	GlobalSync3	R/W	0	Update PWM Generator 3 Same as GlobalSync0 but for PWM generator 3.
2	GlobalSync2	R/W	0	Update PWM Generator 2 Same as GlobalSync0 but for PWM generator 2.
1	GlobalSync1	R/W	0	Update PWM Generator 1 Same as GlobalSync0 but for PWM generator 1.
0	GlobalSync0	R/W	0	Update PWM Generator 0 Setting this bit causes any queued update to a load or comparator register in PWM generator 0 to be applied the next time the corresponding counter becomes zero. This bit automatically clears when the updates have completed; it cannot be cleared by software.

## Register 2: PWM Time Base Sync (PWMSYNC), offset 0x004

This register provides a method to perform synchronization of the counters in the PWM generation blocks. Writing a bit in this register to 1 causes the specified counter to reset back to 0; writing multiple bits resets multiple counters simultaneously. The bits auto-clear after the reset has occurred; reading them back as zero indicates that the synchronization has completed.

### PWM Time Base Sync (PWMSYNC)

Base 0x4002.8000  
 Offset 0x004  
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												Sync3	Sync2	Sync1	Sync0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	Sync3	R/W	0	Reset Generator 3 Counter Performs a reset of the PWM generator 3 counter.
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 (PWMENABLE), 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 (PWMENABLE)

Base 0x4002.8000  
Offset 0x008  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	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								PWM7En	PWM6En	PWM5En	PWM4En	PWM3En	PWM2En	PWM1En	PWM0En
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	PWM7En	R/W	0	PWM7 Output Enable  When set, allows the generated <code>PWM7</code> signal to be passed to the device pin.
6	PWM6En	R/W	0	PWM6 Output Enable  When set, allows the generated <code>PWM6</code> signal to be passed to the device pin.
5	PWM5En	R/W	0	PWM5 Output Enable  When set, allows the generated <code>PWM5</code> signal to be passed to the device pin.
4	PWM4En	R/W	0	PWM4 Output Enable  When set, allows the generated <code>PWM4</code> signal to be passed to the device pin.
3	PWM3En	R/W	0	PWM3 Output Enable  When set, allows the generated <code>PWM3</code> signal to be passed to the device pin.
2	PWM2En	R/W	0	PWM2 Output Enable  When set, allows the generated <code>PWM2</code> signal to be passed to the device pin.

Bit/Field	Name	Type	Reset	Description
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 (PWMINVERT), 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 (PWMINVERT)

Base 0x4002.8000  
Offset 0x00C  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PWM7Inv	PWM6Inv	PWM5Inv	PWM4Inv	PWM3Inv	PWM2Inv	PWM1Inv	PWM0Inv
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	PWM7Inv	R/W	0	Invert PWM7 Signal When set, the generated PWM7 signal is inverted.
6	PWM6Inv	R/W	0	Invert PWM6 Signal When set, the generated PWM6 signal is inverted.
5	PWM5Inv	R/W	0	Invert PWM5 Signal When set, the generated PWM5 signal is inverted.
4	PWM4Inv	R/W	0	Invert PWM4 Signal When set, the generated PWM4 signal is inverted.
3	PWM3Inv	R/W	0	Invert PWM3 Signal When set, the generated PWM3 signal is inverted.
2	PWM2Inv	R/W	0	Invert PWM2 Signal When set, the generated PWM2 signal is inverted.
1	PWM1Inv	R/W	0	Invert PWM1 Signal When set, the generated PWM1 signal is inverted.
0	PWM0Inv	R/W	0	Invert PWM0 Signal When set, the generated PWM0 signal is inverted.

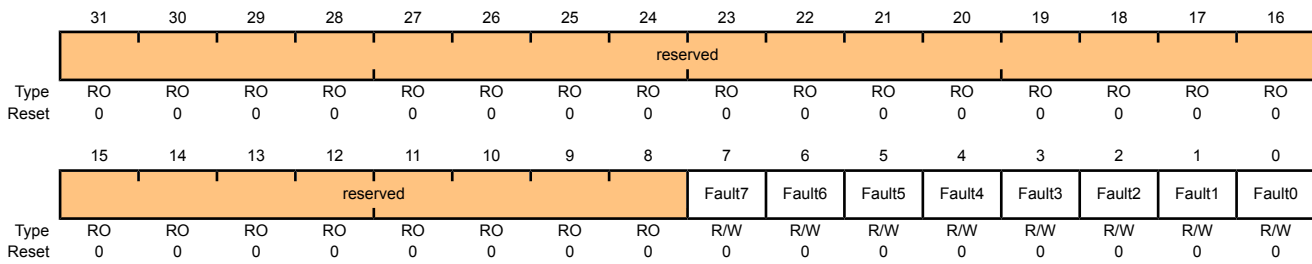
### Register 5: PWM Output Fault (PWMFAULT), offset 0x010

This register controls the behavior of the PWM outputs in the presence of fault conditions. Both the fault inputs and debug events are considered fault conditions. On a fault condition, each PWM signal can be passed through unmodified or driven to a specified value. For outputs that are configured for pass-through, the debug event handling on the corresponding PWM generator also determines if the PWM signal continues to be generated.

Fault condition control occurs before the output inverter, so PWM signals driven to a specified value on fault are inverted if the channel is configured for inversion (therefore, the pin is driven to the logical complement of the specified value on a fault condition).

#### PWM Output Fault (PWMFAULT)

Base 0x4002.8000  
 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	Fault7	R/W	0	PWM7 Fault  When set, the PWM7 output signal is driven to a specified value on a fault condition.
6	Fault6	R/W	0	PWM6 Fault  When set, the PWM6 output signal is driven to a specified value on a fault condition.
5	Fault5	R/W	0	PWM5 Fault  When set, the PWM5 output signal is driven to a specified value on a fault condition.
4	Fault4	R/W	0	PWM4 Fault  When set, the PWM4 output signal is driven to a specified value on a fault condition.
3	Fault3	R/W	0	PWM3 Fault  When set, the PWM3 output signal is driven to a specified value on a fault condition.
2	Fault2	R/W	0	PWM2 Fault  When set, the PWM2 output signal is driven to a specified value on a fault condition.



Bit/Field	Name	Type	Reset	Description
1	Fault1	R/W	0	PWM1 Fault When set, the $P_{WM1}$ output signal is driven to a specified value on a fault condition.
0	Fault0	R/W	0	PWM0 Fault When set, the $P_{WM0}$ output signal is driven to a specified value 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												IntFault3	IntFault2	IntFault1	IntFault0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												IntPWM3	IntPWM2	IntPWM1	IntPWM0
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
31:20	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	IntFault3	R/W	0	Interrupt Fault 3  When set, an interrupt occurs when the fault condition for PWM generator 3 is asserted.
18	IntFault2	R/W	0	Interrupt Fault 2  When set, an interrupt occurs when the fault condition for PWM generator 2 is asserted.
17	IntFault1	R/W	0	Interrupt Fault 1  When set, an interrupt occurs when the fault condition for PWM generator 1 is asserted.
16	IntFault0	R/W	0	Interrupt Fault 0  When set, an interrupt occurs when the <code>FAULT0</code> input is asserted or the fault condition for PWM generator 0 is asserted.
15:4	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	IntPWM3	R/W	0	PWM3 Interrupt Enable  When set, an interrupt occurs when the PWM generator 3 block asserts an interrupt.
2	IntPWM2	R/W	0	PWM2 Interrupt Enable  When set, an interrupt occurs when the PWM generator 2 block asserts an interrupt.
1	IntPWM1	R/W	0	PWM1 Interrupt Enable  When set, an interrupt occurs when the PWM generator 1 block asserts an interrupt.

Bit/Field	Name	Type	Reset	Description
0	IntPWM0	R/W	0	PWM0 Interrupt Enable When set, an interrupt occurs when the PWM generator 0 block asserts an interrupt.

### Register 7: PWM Raw Interrupt Status (PWMRIS), offset 0x018

This register provides the current set of interrupt sources that are asserted, regardless of whether they cause an interrupt to be asserted to the controller. The fault interrupt is latched on detection; it must be cleared through the **PWM Interrupt Status and Clear (PWMISC)** register (see page 850). The PWM generator interrupts simply reflect the status of the PWM generators; they are cleared via the interrupt status register in the PWM generator blocks. Bits set to 1 indicate the events that are active; zero bits indicate that the event in question is not active.

#### PWM Raw Interrupt Status (PWMRIS)

Base 0x4002.8000  
 Offset 0x018  
 Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved												IntFault3	IntFault2	IntFault1	IntFault0
Type	RO	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												IntPWM3	IntPWM2	IntPWM1	IntPWM0
Type	RO	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
31:20	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	IntFault3	RO	0	Interrupt Fault PWM 3 Indicates that the fault condition for PWM generator 3 is asserting.
18	IntFault2	RO	0	Interrupt Fault PWM 2 Indicates that the fault condition for PWM generator 2 is asserting.
17	IntFault1	RO	0	Interrupt Fault PWM 1 Indicates that the fault condition for PWM generator 1 is asserting.
16	IntFault0	RO	0	Interrupt Fault PWM 0 Indicates that the <code>FAULT0</code> input is asserting or the fault condition for PWM generator 0 is asserting.
15:4	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	IntPWM3	RO	0	PWM3 Interrupt Asserted Indicates that the PWM generator 3 block is asserting its interrupt.
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.

Bit/Field	Name	Type	Reset	Description
0	IntPWM0	RO	0	PWM0 Interrupt Asserted Indicates that the PWM generator 0 block is asserting its interrupt.

### Register 8: PWM Interrupt Status and Clear (PWMISC), offset 0x01C

This register provides a summary of the interrupt status of the individual PWM generator blocks. A bit set to 1 indicates that the corresponding generator block is asserting an interrupt. The individual interrupt status registers in each block must be consulted to determine the reason for the interrupt, and used to clear the interrupt. For the fault interrupt, a write of 1 to that bit position clears the latched interrupt status.

#### PWM Interrupt Status and Clear (PWMISC)

Base 0x4002.8000  
 Offset 0x01C  
 Type R/W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved												IntFault3	IntFault2	IntFault1	IntFault0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												IntPWM3	IntPWM2	IntPWM1	IntPWM0
Type	RO	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
31:20	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	IntFault3	R/W1C	0	<b>FAULT3 Interrupt Asserted</b>  Indicates that the FAULT3 input is asserting or the FAULT3 latch has captured an assertion.
18	IntFault2	R/W1C	0	<b>FAULT2 Interrupt Asserted</b>  Indicates that the FAULT2 input is asserting or the FAULT2 latch has captured an assertion.
17	IntFault1	R/W1C	0	<b>FAULT1 Interrupt Asserted</b>  Indicates that the FAULT1 input is asserting or the FAULT1 latch has captured an assertion.
16	IntFault0	R/W1C	0	<b>FAULT0 Interrupt Asserted</b>  Indicates that the FAULT0 input is asserting or the fault condition for generator 0 is asserting a fault.
15:4	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	IntPWM3	RO	0	<b>PWM3 Interrupt Status</b>  Indicates if the PWM generator 3 block is asserting an interrupt.
2	IntPWM2	RO	0	<b>PWM2 Interrupt Status</b>  Indicates if the PWM generator 2 block is asserting an interrupt.
1	IntPWM1	RO	0	<b>PWM1 Interrupt Status</b>  Indicates if the PWM generator 1 block is asserting an interrupt.

Bit/Field	Name	Type	Reset	Description
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 signals.

#### PWM Status (PWMSTATUS)

Base 0x4002.8000  
 Offset 0x020  
 Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												Fault3	Fault2	Fault1	Fault0
Type	RO	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	0x00	Software should not rely on the value of 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	Fault3	RO	0	Fault3 Interrupt Status When set, indicates the fault condition for PWM generator 3 is asserted.
2	Fault2	RO	0	Fault2 Interrupt Status When set, indicates the fault condition for PWM generator 2 is asserted.
1	Fault1	RO	0	Fault1 Interrupt Status When set, indicates the fault condition for PWM generator 1 is asserted.
0	Fault0	RO	0	Fault0 Interrupt Status When set, indicates the FAULT0 input is asserted, or that the fault condition for PWM generator 0 is asserted.



**Register 10: PWM Fault Condition Value (PWMFAULTVAL), offset 0x024**

This register specifies the output value driven on the PWM signals during a fault condition if the corresponding bit in the **PWMFAULT** register is indicating that the PWM signal drives a value.

## PWM Fault Condition Value (PWMFAULTVAL)

Base 0x4002.8000

Offset 0x024

Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PWM7	PWM6	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	PWM7	R/W	0	<p>PWM7 Fault Value</p> <p>The PWM7 output signal is driven to the value specified in this bit during fault conditions if the <code>Fault7</code> bit in the <b>PWMFAULT</b> register is set.</p>
6	PWM6	R/W	0	<p>PWM6 Fault Value</p> <p>The PWM6 output signal is driven to the value specified in this bit during fault conditions if the <code>Fault6</code> bit in the <b>PWMFAULT</b> register is set.</p>
5	PWM5	R/W	0	<p>PWM5 Fault Value</p> <p>The PWM5 output signal is driven to the value specified in this bit during fault conditions if the <code>Fault5</code> bit in the <b>PWMFAULT</b> register is set.</p>
4	PWM4	R/W	0	<p>PWM4 Fault Value</p> <p>The PWM4 output signal is driven to the value specified in this bit during fault conditions if the <code>Fault4</code> bit in the <b>PWMFAULT</b> register is set.</p>
3	PWM3	R/W	0	<p>PWM3 Fault Value</p> <p>The PWM3 output signal is driven to the value specified in this bit during fault conditions if the <code>Fault3</code> bit in the <b>PWMFAULT</b> register is set.</p>
2	PWM2	R/W	0	<p>PWM2 Fault Value</p> <p>The PWM2 output signal is driven to the value specified in this bit during fault conditions if the <code>Fault2</code> bit in the <b>PWMFAULT</b> register is set.</p>
1	PWM1	R/W	0	<p>PWM1 Fault Value</p> <p>The PWM1 output signal is driven to the value specified in this bit during fault conditions if the <code>Fault1</code> bit in the <b>PWMFAULT</b> register is set.</p>

Bit/Field	Name	Type	Reset	Description
0	PWM0	R/W	0	PWM0 Fault Value The PWM0 output signal is driven to the value specified in this bit during fault conditions if the Fault0 bit in the <b>PWMFAULT</b> register is set.

**Register 11: PWM0 Control (PWM0CTL), offset 0x040****Register 12: PWM1 Control (PWM1CTL), offset 0x080****Register 13: PWM2 Control (PWM2CTL), offset 0x0C0****Register 14: PWM3 Control (PWM3CTL), offset 0x100**

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, the PWM2 block produces the  $PWM4$  and  $PWM5$  outputs, and the PWM3 block produces the  $PWM6$  and  $PWM7$  outputs.

## PWM0 Control (PWM0CTL)

Base 0x4002.8000  
Offset 0x040  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved													LATCH	MINFLTPER	FLTSRC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DBFallUpd		DBRiseUpd		DBCtlUpd		GenBUpd		GenAUpd		CmpBUpd	CmpAUpd	LoadUpd	Debug	Mode	Enable
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:19	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
18	LATCH	R/W	0	<p>Latch Fault Input</p> <p>This bit controls the behavior of the fault condition in a PWM generator.</p> <p>The fault condition may be latched and internally asserted because the fault condition logic includes the generator's <code>IntFaultn</code> bit (of the <b>PWMISC</b> register) enabled by the <code>LATCH</code> bit.</p> <p>Therefore, if the <b>PWMINTEN</b> <code>IntFaultn</code> bit is set, a fault condition sets the <b>PWMISC</b> <code>IntFaultn</code> bit (generating an interrupt) and the fault condition is extended in the generator logic until software clears the <b>PWMISC</b> <code>IntFaultn</code> bit.</p> <p>Value Description</p> <p>0 Fault Condition Not Latched</p> <p>A fault condition is in effect for as long as the generating source is asserting.</p> <p>1 Fault Condition Latched</p> <p>A fault condition is set as the result of the assertion of the faulting source and is held (latched) while the <b>PWMISC</b> <code>IntFaultn</code> bit is set. Clearing the <code>IntFaultn</code> bit clears the fault condition.</p>
17	MINFLTPER	R/W	0	<p>Minimum Fault Period</p> <p>This bit specifies that the PWM generator enables a one-shot counter to provide a minimum fault condition period.</p> <p>The timer begins counting on the rising edge of the fault condition to extend the condition for a minimum duration of the count value. The timer ignores the state of the fault condition while counting.</p> <p>The minimum fault delay is in effect only when the <code>MINFLTPER</code> bit is set. If a detected fault is in the process of being extended when the <code>MINFLTPER</code> bit is cleared, the fault condition extension is aborted.</p> <p>The delay time is specified by the <b>PWMnMINFLTPER</b> register <code>MFP</code> field value. The effect of this is to pulse stretch the fault condition input.</p> <p>The delay value is defined by the PWM clock period. Because the fault input is not synchronized to the PWM clock, the period of the time is <math>PWMClock * (MFP \text{ value} + 1)</math> or <math>PWMClock * (MFP \text{ value} + 2)</math>.</p> <p>The delay function makes sense only if the fault source is unlatched. A latched fault source makes the fault condition appear asserted until cleared by software and negates the utility of the extend feature. It applies to all fault condition sources as specified in the <code>FLTSRC</code> field.</p> <p>Value Description</p> <p>0 Fault Condition Period Not Extended</p> <p>The <code>FAULT</code> input deassertion is unaffected.</p> <p>1 Fault Condition Period Extended</p> <p>The <b>PWMnMINFLTPER</b> one-shot counter is active and extends the period of the fault condition to a minimum period.</p>

Bit/Field	Name	Type	Reset	Description																
16	FLTSRC	R/W	0	<p>Fault Condition Source</p> <p>This bit specifies the fault condition source.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Fault0</td> </tr> <tr> <td></td> <td>The Fault condition is determined by the Fault0 input.</td> </tr> <tr> <td>1</td> <td>Register-Defined</td> </tr> <tr> <td></td> <td>The Fault condition is determined by the configuration of the <b>PWMnFLTSRC0</b> and <b>PWMnFLTSRC1</b> registers.</td> </tr> </tbody> </table>	Value	Description	0	Fault0		The Fault condition is determined by the Fault0 input.	1	Register-Defined		The Fault condition is determined by the configuration of the <b>PWMnFLTSRC0</b> and <b>PWMnFLTSRC1</b> registers.						
Value	Description																			
0	Fault0																			
	The Fault condition is determined by the Fault0 input.																			
1	Register-Defined																			
	The Fault condition is determined by the configuration of the <b>PWMnFLTSRC0</b> and <b>PWMnFLTSRC1</b> registers.																			
15:14	DBFallUpd	R/W	0	<p><b>PWMnDBFALL</b> Update Mode</p> <p>Specifies the update mode for the <b>PWMnDBFALL</b> register.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Immediate</td> </tr> <tr> <td></td> <td>The <b>PWMnDBFALL</b> register value is immediately updated on a write.</td> </tr> <tr> <td>1</td> <td>Reserved</td> </tr> <tr> <td>2</td> <td>Locally Synchronized</td> </tr> <tr> <td></td> <td>Updates to the register are reflected to the generator the next time the counter is 0.</td> </tr> <tr> <td>3</td> <td>Globally Synchronized</td> </tr> <tr> <td></td> <td>Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (<b>PWMCTL</b>) register.</td> </tr> </tbody> </table>	Value	Description	0	Immediate		The <b>PWMnDBFALL</b> register value is immediately updated on a write.	1	Reserved	2	Locally Synchronized		Updates to the register are reflected to the generator the next time the counter is 0.	3	Globally Synchronized		Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control ( <b>PWMCTL</b> ) register.
Value	Description																			
0	Immediate																			
	The <b>PWMnDBFALL</b> register value is immediately updated on a write.																			
1	Reserved																			
2	Locally Synchronized																			
	Updates to the register are reflected to the generator the next time the counter is 0.																			
3	Globally Synchronized																			
	Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control ( <b>PWMCTL</b> ) register.																			
13:12	DBRiseUpd	R/W	0	<p><b>PWMnDBRISE</b> Update Mode</p> <p>Specifies the update mode for the <b>PWMnDBRISE</b> register.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Immediate</td> </tr> <tr> <td></td> <td>The <b>PWMnDBRISE</b> register value is immediately updated on a write.</td> </tr> <tr> <td>1</td> <td>Reserved</td> </tr> <tr> <td>2</td> <td>Locally Synchronized</td> </tr> <tr> <td></td> <td>Updates to the register are reflected to the generator the next time the counter is 0.</td> </tr> <tr> <td>3</td> <td>Globally Synchronized</td> </tr> <tr> <td></td> <td>Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (<b>PWMCTL</b>) register.</td> </tr> </tbody> </table>	Value	Description	0	Immediate		The <b>PWMnDBRISE</b> register value is immediately updated on a write.	1	Reserved	2	Locally Synchronized		Updates to the register are reflected to the generator the next time the counter is 0.	3	Globally Synchronized		Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control ( <b>PWMCTL</b> ) register.
Value	Description																			
0	Immediate																			
	The <b>PWMnDBRISE</b> register value is immediately updated on a write.																			
1	Reserved																			
2	Locally Synchronized																			
	Updates to the register are reflected to the generator the next time the counter is 0.																			
3	Globally Synchronized																			
	Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control ( <b>PWMCTL</b> ) register.																			

Bit/Field	Name	Type	Reset	Description								
11:10	DBCtlUpd	R/W	0	<p><b>PWMnDBCTL</b> Update Mode</p> <p>Specifies the update mode for the <b>PWMnDBCTL</b> register.</p> <p>Value Description</p> <table border="1"><tr><td>0</td><td>Immediate</td></tr><tr><td>1</td><td>Reserved</td></tr><tr><td>2</td><td>Locally Synchronized</td></tr><tr><td>3</td><td>Globally Synchronized</td></tr></table>	0	Immediate	1	Reserved	2	Locally Synchronized	3	Globally Synchronized
0	Immediate											
1	Reserved											
2	Locally Synchronized											
3	Globally Synchronized											
9:8	GenBUpd	R/W	0	<p><b>PWMnGENB</b> Update Mode</p> <p>Specifies the update mode for the <b>PWMnGENB</b> register.</p> <p>Value Description</p> <table border="1"><tr><td>0</td><td>Immediate</td></tr><tr><td>1</td><td>Reserved</td></tr><tr><td>2</td><td>Locally Synchronized</td></tr><tr><td>3</td><td>Globally Synchronized</td></tr></table>	0	Immediate	1	Reserved	2	Locally Synchronized	3	Globally Synchronized
0	Immediate											
1	Reserved											
2	Locally Synchronized											
3	Globally Synchronized											

Bit/Field	Name	Type	Reset	Description										
7:6	GenAUpd	R/W	0	<p><b>PWMnGENA</b> Update Mode</p> <p>Specifies the update mode for the <b>PWMnGENA</b> register.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td> <p>Immediate</p> <p>The <b>PWMnGENA</b> register value is immediately updated on a write.</p> </td> </tr> <tr> <td>1</td> <td>Reserved</td> </tr> <tr> <td>2</td> <td> <p>Locally Synchronized</p> <p>Updates to the register are reflected to the generator the next time the counter is 0.</p> </td> </tr> <tr> <td>3</td> <td> <p>Globally Synchronized</p> <p>Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (<b>PWMCTL</b>) register.</p> </td> </tr> </tbody> </table>	Value	Description	0	<p>Immediate</p> <p>The <b>PWMnGENA</b> register value is immediately updated on a write.</p>	1	Reserved	2	<p>Locally Synchronized</p> <p>Updates to the register are reflected to the generator the next time the counter is 0.</p>	3	<p>Globally Synchronized</p> <p>Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (<b>PWMCTL</b>) register.</p>
Value	Description													
0	<p>Immediate</p> <p>The <b>PWMnGENA</b> register value is immediately updated on a write.</p>													
1	Reserved													
2	<p>Locally Synchronized</p> <p>Updates to the register are reflected to the generator the next time the counter is 0.</p>													
3	<p>Globally Synchronized</p> <p>Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (<b>PWMCTL</b>) register.</p>													
5	CmpBUpd	R/W	0	<p>Comparator B Update Mode</p> <p>Same as <code>CmpAUpd</code> but for the comparator B register.</p>										
4	CmpAUpd	R/W	0	<p>Comparator A Update Mode</p> <p>The Update mode for the comparator A register. When not set, updates to the register are reflected to the comparator the next time the counter is 0. When set, updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the <b>PWM Master Control (PWMCTL)</b> register (see page 839).</p>										
3	LoadUpd	R/W	0	<p>Load Register Update Mode</p> <p>The Update mode for the load register. When not set, updates to the register are reflected to the counter the next time the counter is 0. When set, updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the <b>PWM Master Control (PWMCTL)</b> register.</p>										
2	Debug	R/W	0	<p>Debug Mode</p> <p>The behavior of the counter in Debug mode. When not set, the counter stops running when it next reaches 0, and continues running again when no longer in Debug mode. When set, the counter always runs.</p>										
1	Mode	R/W	0	<p>Counter Mode</p> <p>The mode for the counter. When not set, the counter counts down from the load value to 0 and then wraps back to the load value (Count-Down mode). When set, the counter counts up from 0 to the load value, back down to 0, and then repeats (Count-Up/Down mode).</p>										
0	Enable	R/W	0	<p>PWM Block Enable</p> <p>Master enable for the PWM generation block. When not set, the entire block is disabled and not clocked. When set, the block is enabled and produces PWM signals.</p>										

**Register 15: PWM0 Interrupt and Trigger Enable (PWM0INTEN), offset 0x044**

**Register 16: PWM1 Interrupt and Trigger Enable (PWM1INTEN), offset 0x084**

**Register 17: PWM2 Interrupt and Trigger Enable (PWM2INTEN), offset 0x0C4**

**Register 18: PWM3 Interrupt and Trigger Enable (PWM3INTEN), offset 0x104**

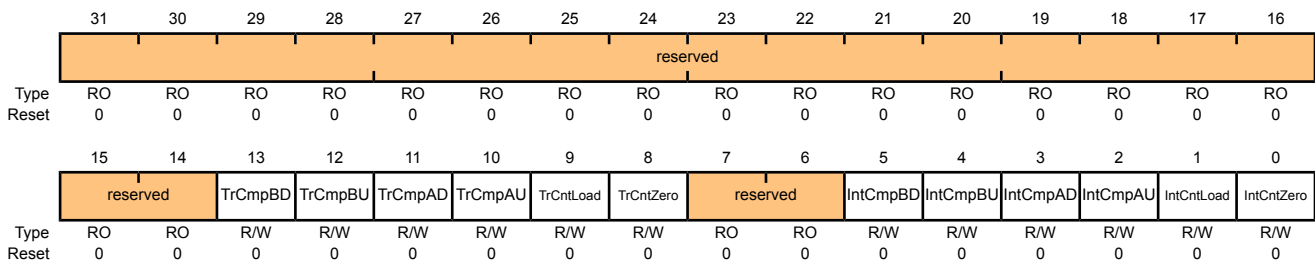
These registers control the interrupt and ADC trigger generation capabilities of the PWM generators (**PWM0INTEN** controls the PWM generator 0 block, and so on). The events that can cause an interrupt or an ADC trigger are:

- 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



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.



Bit/Field	Name	Type	Reset	Description
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.
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 19: PWM0 Raw Interrupt Status (PWM0RIS), offset 0x048**

**Register 20: PWM1 Raw Interrupt Status (PWM1RIS), offset 0x088**

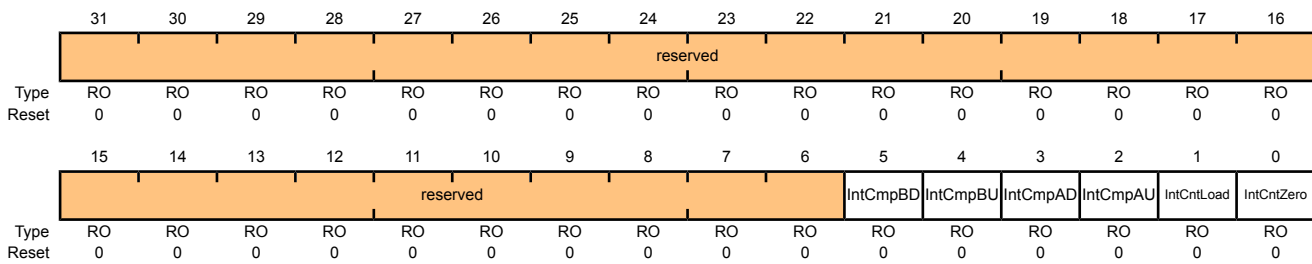
**Register 21: PWM2 Raw Interrupt Status (PWM2RIS), offset 0x0C8**

**Register 22: PWM3 Raw Interrupt Status (PWM3RIS), offset 0x108**

These registers provide the current set of interrupt sources that are asserted, regardless of whether they cause an interrupt to be asserted to the controller (**PWM0RIS** controls the PWM generator 0 block, and so on). Bits set to 1 indicate the latched events that have occurred; bits set to 0 indicate that the event in question has not occurred.

PWM0 Raw Interrupt Status (PWM0RIS)

Base 0x4002.8000  
 Offset 0x048  
 Type RO, reset 0x0000.0000



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 23: PWM0 Interrupt Status and Clear (PWM0ISC), offset 0x04C****Register 24: PWM1 Interrupt Status and Clear (PWM1ISC), offset 0x08C****Register 25: PWM2 Interrupt Status and Clear (PWM2ISC), offset 0x0CC****Register 26: PWM3 Interrupt Status and Clear (PWM3ISC), offset 0x10C**

These registers provide the current set of interrupt sources that are asserted to the controller (**PWM0ISC** controls the PWM generator 0 block, and so on). Bits set to 1 indicate the latched events that have occurred; bits set to 0 indicate that the event in question has not occurred. These are R/W1C registers; writing a 1 to a bit position clears the corresponding interrupt reason.

## PWM0 Interrupt Status and Clear (PWM0ISC)

Base 0x4002.8000

Offset 0x04C

Type R/W1C, reset 0x0000.0000

	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											IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bit/Field	Name	Type	Reset	Description
31:6	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 27: PWM0 Load (PWM0LOAD), offset 0x050**

**Register 28: PWM1 Load (PWM1LOAD), offset 0x090**

**Register 29: PWM2 Load (PWM2LOAD), offset 0x0D0**

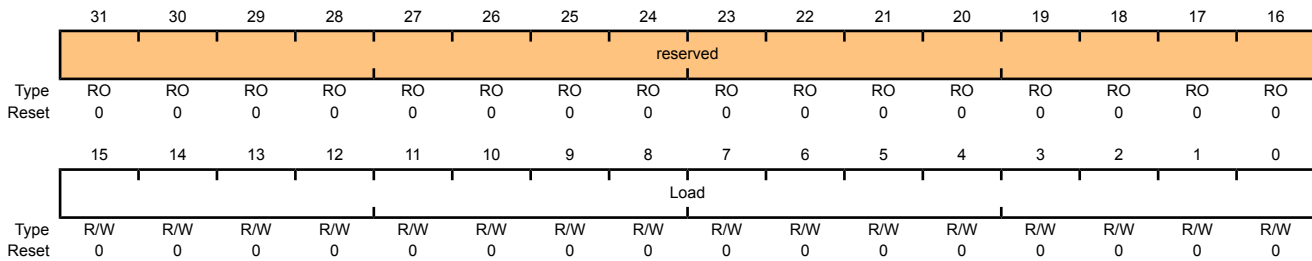
**Register 30: PWM3 Load (PWM3LOAD), offset 0x110**

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 839). 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 31: PWM0 Counter (PWM0COUNT), offset 0x054****Register 32: PWM1 Counter (PWM1COUNT), offset 0x094****Register 33: PWM2 Counter (PWM2COUNT), offset 0x0D4****Register 34: PWM3 Counter (PWM3COUNT), offset 0x114**

These registers contain the current value of the PWM counter. 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 868 and page 871) or drive an interrupt or ADC trigger (via the **PWMnINTEN** register, see page 860). 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

	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
	Count															
Type	RO	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
31:16	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 35: PWM0 Compare A (PWM0CMPA), offset 0x058**

**Register 36: PWM1 Compare A (PWM1CMPA), offset 0x098**

**Register 37: PWM2 Compare A (PWM2CMPA), offset 0x0D8**

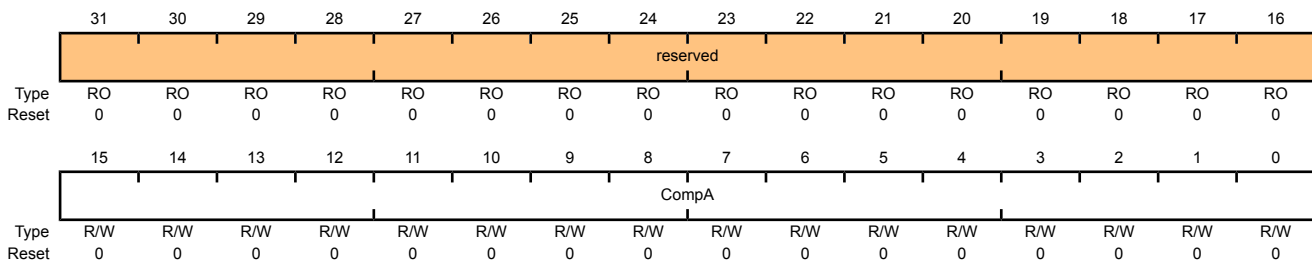
**Register 38: PWM3 Compare A (PWM3CMPA), offset 0x118**

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 864), then no pulse is ever output.

If the comparator A update mode is immediate (based on the **CompAUpd** bit in the **PWMnCTL** register), this 16-bit **CompA** value is used the next time the counter reaches zero. If the update mode is synchronous, it is used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 839). 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



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 39: PWM0 Compare B (PWM0CMPB), offset 0x05C****Register 40: PWM1 Compare B (PWM1CMPB), offset 0x09C****Register 41: PWM2 Compare B (PWM2CMPB), offset 0x0DC****Register 42: PWM3 Compare B (PWM3CMPB), offset 0x11C**

These registers contain a value to be compared against the counter (**PWM0CMPB** controls the PWM generator 0 block, and so on). When this value matches the counter, a pulse is output; this can drive the generation of a PWM signal (via the **PWMnGENA/PWMnGENB** registers) or drive an interrupt or ADC trigger (via the **PWMnINTEN** register). If the value of this register is greater than the **PWMnLOAD** register, no pulse is ever output.

If the comparator B update mode is immediate (based on the **CompBUpd** bit in the **PWMnCTL** register), this 16-bit **CompB** value is used the next time the counter reaches zero. If the update mode is synchronous, it is used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 839). 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

	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
	CompB															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	CompB	R/W	0x00	Comparator B Value The value to be compared against the counter.

**Register 43: PWM0 Generator A Control (PWM0GENA), offset 0x060**

**Register 44: PWM1 Generator A Control (PWM1GENA), offset 0x0A0**

**Register 45: PWM2 Generator A Control (PWM2GENA), offset 0x0E0**

**Register 46: PWM3 Generator A Control (PWM3GENA), offset 0x120**

These registers control the generation of the  $PWM_nA$  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; **PWM2GENA**, the  $PWM2A$  signal; and **PWM3GENA**, the  $PWM3A$  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.

If the Generator A update mode is immediate (based on the  $GenAUpd$  field encoding in the **PWMnCTL** register), this 16-bit  $GenAUpd$  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 839). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWM0 Generator A Control (PWM0GENA)

Base 0x4002.8000  
Offset 0x060  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	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				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	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.



Bit/Field	Name	Type	Reset	Description										
11:10	ActCmpBD	R/W	0x0	<p>Action for Comparator B Down</p> <p>The action to be taken when the counter matches comparator B while counting down.</p> <p>The table below defines the effect of the event on the output signal.</p> <table border="1"> <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.													
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 <code>Mode</code> bit in the <b>PWMnCTL</b> register (see page 855) is set to 1.</p> <p>The table below defines the effect of the event on the output signal.</p> <table border="1"> <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 border="1"> <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 <code>Mode</code> 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 border="1"> <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
3:2	ActLoad	R/W	0x0	Action for Counter=Load The action to be taken when the counter matches the load value. 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.
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 47: PWM0 Generator B Control (PWM0GENB), offset 0x064**

**Register 48: PWM1 Generator B Control (PWM1GENB), offset 0x0A4**

**Register 49: PWM2 Generator B Control (PWM2GENB), offset 0x0E4**

**Register 50: PWM3 Generator B Control (PWM3GENB), offset 0x124**

These registers control the generation of the  $PWM_nB$  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; **PWM2GENB**, the  $PWM2B$  signal; and **PWM3GENB**, the  $PWM3B$  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.

If the Generator B update mode is immediate (based on the  $GenBU_{pd}$  field encoding in the **PWMnCTL** register), this 16-bit  $GenBU_{pd}$  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 839). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

#### PWM0 Generator B Control (PWM0GENB)

Base 0x4002.8000  
Offset 0x064  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	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				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	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.

Bit/Field	Name	Type	Reset	Description										
11:10	ActCmpBD	R/W	0x0	<p>Action for Comparator B Down</p> <p>The action to be taken when the counter matches comparator B while counting down.</p> <p>The table below defines the effect of the event on the output signal.</p> <table border="1"> <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.													
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 <code>Mode</code> 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 border="1"> <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 border="1"> <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 <code>Mode</code> 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 border="1"> <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										
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 border="1"> <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.													
1:0	ActZero	R/W	0x0	<p>Action for Counter=0</p> <p>The action to be taken when the counter is 0.</p> <p>The table below defines the effect of the event on the output signal.</p> <table border="1"> <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.													

**Register 51: PWM0 Dead-Band Control (PWM0DBCTL), offset 0x068**

**Register 52: PWM1 Dead-Band Control (PWM1DBCTL), offset 0x0A8**

**Register 53: PWM2 Dead-Band Control (PWM2DBCTL), offset 0x0E8**

**Register 54: PWM3 Dead-Band Control (PWM3DBCTL), offset 0x128**

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 875), 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 876). In a similar manner, **PWM2** and **PWM3** are produced from the **PWM1A** and **PWM1B** signals, **PWM4** and **PWM5** are produced from the **PWM2A** and **PWM2B** signals, and **PWM6** and **PWM7** are produced from the **PWM3A** and **PWM3B** signals.

If the Dead-Band Control mode is immediate (based on the **DBCtlUpd** field encoding in the **PWMnCTL** register), this 16-bit **DBCtlUpd** 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 839). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

**PWM0 Dead-Band Control (PWM0DBCTL)**

Base 0x4002.8000  
 Offset 0x068  
 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															Enable
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	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 55: PWM0 Dead-Band Rising-Edge Delay (PWM0DBRISE), offset 0x06C**

**Register 56: PWM1 Dead-Band Rising-Edge Delay (PWM1DBRISE), offset 0x0AC**

**Register 57: PWM2 Dead-Band Rising-Edge Delay (PWM2DBRISE), offset 0x0EC**

**Register 58: PWM3 Dead-Band Rising-Edge Delay (PWM3DBRISE), offset 0x12C**

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; **PWM4** is produced from **PWM2A** with its rising edge delayed; and **PWM6** is produced from **PWM3A** with its rising edge delayed.

If the Dead-Band Rising-Edge Delay mode is immediate (based on the **DBRiseUpd** field encoding in the **PWMnCTL** register), this 16-bit **DBRiseUpd** 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 839). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

#### PWM0 Dead-Band Rising-Edge Delay (PWM0DBRISE)

Base 0x4002.8000  
Offset 0x06C  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				RiseDelay											
Type	RO	RO	RO	RO	R/W	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	RiseDelay	R/W	0	Dead-Band Rise Delay The number of clock ticks to delay the rising edge.

**Register 59: PWM0 Dead-Band Falling-Edge-Delay (PWM0DBFALL), offset 0x070**

**Register 60: PWM1 Dead-Band Falling-Edge-Delay (PWM1DBFALL), offset 0x0B0**

**Register 61: PWM2 Dead-Band Falling-Edge-Delay (PWM2DBFALL), offset 0x0F0**

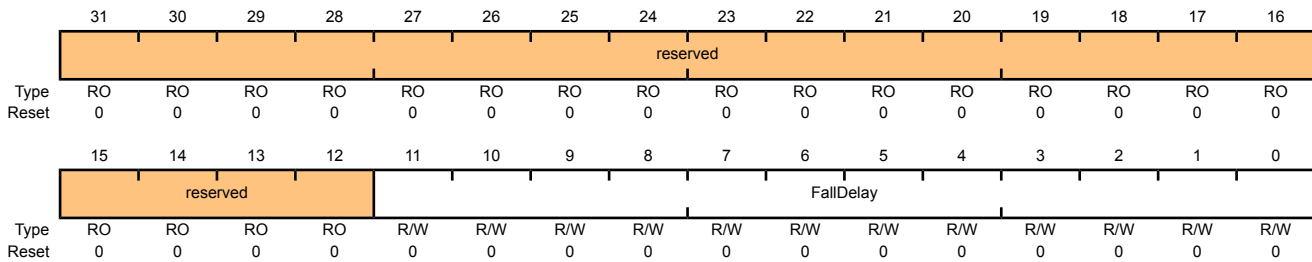
**Register 62: PWM3 Dead-Band Falling-Edge-Delay (PWM3DBFALL), offset 0x130**

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, **PWM5** is produced from **PWM2A** with its falling edge delayed, and **PWM7** is produced from **PWM3A** with its falling edge delayed.

If the Dead-Band Falling-Edge-Delay mode is immediate (based on the **DBFallUp** field encoding in the **PWMnCTL** register), this 16-bit **DBFallUp** 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 839). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

**PWM0 Dead-Band Falling-Edge-Delay (PWM0DBFALL)**

Base 0x4002.8000  
 Offset 0x070  
 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	FallDelay	R/W	0x00	Dead-Band Fall Delay The number of clock ticks to delay the falling edge.



**Register 63: PWM0 Fault Source 0 (PWM0FLTSRC0), offset 0x074**

**Register 64: PWM1 Fault Source 0 (PWM1FLTSRC0), offset 0x0B4**

**Register 65: PWM2 Fault Source 0 (PWM2FLTSRC0), offset 0x0F4**

**Register 66: PWM3 Fault Source 0 (PWM3FLTSRC0), offset 0x134**

This register specifies which fault pin inputs are used to indicate a fault condition. Each bit in the following register indicates whether the corresponding fault pin is included in the fault condition. All enabled fault pins are ORed together to form the **PWMnFLTSRC0** portion of the fault condition. The **PWMnFLTSRC0** fault condition is then ORed with the **PWMnFLTSRC1** fault condition to generate the final fault condition for the PWM generator.

If the **FLTSRC** bit in the **PWMnCTL** register (see page 855) is clear, only the PWM **Fault0** pin affects the fault condition generated. Otherwise, sources defined in **PWMnFLTSRC0** and **PWMnFLTSRC1** affect the fault condition generated.

PWM0 Fault Source 0 (PWM0FLTSRC0)

Base 0x4002.8000  
Offset 0x074  
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												FAULT3	FAULT2	FAULT1	FAULT0	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of 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	FAULT3	R/W	0	<p><b>Fault3</b></p> <p>The same function as <b>Fault0</b>, except applied for the <b>FAULT3</b> input.</p> <p><b>Note:</b> The <b>FLTSRC</b> bit in the <b>PWMnCTL</b> register must be set for this bit to affect fault condition generation.</p>
2	FAULT2	R/W	0	<p><b>Fault2</b></p> <p>The same function as <b>Fault0</b>, except applied for the <b>FAULT2</b> input.</p> <p><b>Note:</b> The <b>FLTSRC</b> bit in the <b>PWMnCTL</b> register must be set for this bit to affect fault condition generation.</p>
1	FAULT1	R/W	0	<p><b>Fault1</b></p> <p>The same function as <b>Fault0</b>, except applied for the <b>FAULT1</b> input.</p> <p><b>Note:</b> The <b>FLTSRC</b> bit in the <b>PWMnCTL</b> register must be set for this bit to affect fault condition generation.</p>

Bit/Field	Name	Type	Reset	Description				
0	FAULT0	R/W	0	<p>FAULT0</p> <p>Specifies the contribution of the FAULT0 input to the generation of a fault condition.</p> <p>Value Description</p> <table><tbody><tr><td>0</td><td>Suppressed</td></tr><tr><td>1</td><td>Generated</td></tr></tbody></table> <p>The FAULT0 signal is suppressed and cannot generate a fault condition.</p> <p>The FAULT0 signal value is ORed with all other fault condition generation inputs (Fault signals).</p>	0	Suppressed	1	Generated
0	Suppressed							
1	Generated							

**Register 67: PWM0 Fault Source 1 (PWM0FLTSRC1), offset 0x078**

**Register 68: PWM1 Fault Source 1 (PWM1FLTSRC1), offset 0x0B8**

**Register 69: PWM2 Fault Source 1 (PWM2FLTSRC1), offset 0x0F8**

**Register 70: PWM3 Fault Source 1 (PWM3FLTSRC1), offset 0x138**

This register specifies which digital comparator triggers from the ADC are used to indicate a fault condition. Each bit in the following register indicates whether the corresponding digital comparator trigger is included in the fault condition. All enabled digital comparator triggers are ORed together to form the **PWMnFLTSRC1** portion of the fault condition. The **PWMnFLTSRC1** fault condition is then ORed with the **PWMnFLTSRC0** fault condition to generate the final fault condition for the PWM generator.

If the **FLTSRC** bit in the **PWMnCTL** register (see page 855) is clear, only the PWM **Fault0** pin affects the fault condition generated. Otherwise, sources defined in **PWMnFLTSRC0** and **PWMnFLTSRC1** affect the fault condition generated.

PWM0 Fault Source 1 (PWM0FLTSRC1)

Base 0x4002.8000  
 Offset 0x078  
 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								DCMP7	DCMP6	DCMP5	DCMP4	DCMP3	DCMP2	DCMP1	DCMP0
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	DCMP7	R/W	0	Digital Comparator 7  The same function as Digital Comparator 0, except applied for Comparator 7.
6	DCMP6	R/W	0	Digital Comparator 6  The same function as Digital Comparator 0, except applied for Comparator 6.
5	DCMP5	R/W	0	Digital Comparator 5  The same function as Digital Comparator 0, except applied for Comparator 5.
4	DCMP4	R/W	0	Digital Comparator 4  The same function as Digital Comparator 0, except applied for Comparator 4.

Bit/Field	Name	Type	Reset	Description
3	DCMP3	R/W	0	Digital Comparator 3 The same function as Digital Comparator 0, except applied for Comparator 3.
2	DCMP2	R/W	0	Digital Comparator 2 The same function as Digital Comparator 0, except applied for Comparator 2.
1	DCMP1	R/W	0	Digital Comparator 1 The same function as Digital Comparator 0, except applied for Comparator 1.
0	DCMP0	R/W	0	Digital Comparator 0 Specifies the contribution of Digital Comparator 0 to the generation of a fault condition.  Value Description 0 Suppressed The comparator trigger output signal is suppressed and cannot generate a fault condition. 1 Triggers Fault The comparator trigger output signal value is ORed with all other enabled trigger outputs.

**Register 71: PWM0 Minimum Fault Period (PWM0MINFLTPER), offset 0x07C**

**Register 72: PWM1 Minimum Fault Period (PWM1MINFLTPER), offset 0x0BC**

**Register 73: PWM2 Minimum Fault Period (PWM2MINFLTPER), offset 0x0FC**

**Register 74: PWM3 Minimum Fault Period (PWM3MINFLTPER), offset 0x13C**

If the `MINFLTPER` bit in the `PWMnCTL` register is set, this register specifies the 16-bit time-extension value to be used in extending the fault condition. The value is loaded into a 16-bit down counter, and the counter value is used to extend the fault condition. The fault condition is released in the clock immediately after the counter value reaches 0. The fault condition is asynchronous to the PWM clock; and the delay value is the product of the PWM clock period and the (MFP field value + 1) or (MFP field value + 2) depending on when the fault condition asserts with respect to the PWM clock. The counter decrements at the PWM clock rate, without pause or condition.

#### PWM0 Minimum Fault Period (PWM0MINFLTPER)

Base 0x4002.8000  
Offset 0x07C  
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	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
	MFP															
Type	RO	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
31:16	reserved	R/W	0	Software should not rely on the value of 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	MFP	RO	0	Minimum Fault Period  The number of PWM clocks by which a fault condition is extended when the delay is enabled by <code>PWMnCTL</code> <code>MINFLTPER</code> .

**Register 75: PWM0 Fault Pin Logic Sense (PWM0FLTSEN), offset 0x800**

**Register 76: PWM1 Fault Pin Logic Sense (PWM1FLTSEN), offset 0x880**

**Register 77: PWM2 Fault Pin Logic Sense (PWM2FLTSEN), offset 0x900**

**Register 78: PWM3 Fault Pin Logic Sense (PWM3FLTSEN), offset 0x980**

This register defines the PWM fault pin logic sense.

PWM0 Fault Pin Logic Sense (PWM0FLTSEN)

Base 0x4002.8000  
 Offset 0x800  
 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												FAULT3	FAULT2	FAULT1	FAULT0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of 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	FAULT3	R/W	0	Fault3 Sense  The same function as FLT0SEN, except applied for the FAULT3 input.
2	FAULT2	R/W	0	Fault2 Sense  The same function as FLT0SEN, except applied for the FAULT2 input.
1	FAULT1	R/W	0	Fault1 Sense  The same function as FLT0SEN, except applied for the FAULT1 input.
0	FAULT0	R/W	0	Fault0 Sense  This bit specifies the sense of the FAULT0 input pin, and it determines what sense is considered asserted, that is, the sense of the input (High or Low) that indicates error.  Value Description 0 High 1 Low  The fault sense is used to translate the incoming FAULT0 pin signal sense to an internal positive signal.

**Register 79: PWM0 Fault Status 0 (PWM0FLTSTAT0), offset 0x804**

**Register 80: PWM1 Fault Status 0 (PWM1FLTSTAT0), offset 0x884**

**Register 81: PWM2 Fault Status 0 (PWM2FLTSTAT0), offset 0x904**

**Register 82: PWM3 Fault Status 0 (PWM3FLTSTAT0), offset 0x984**

Along with the **PWMnFLTSTAT1** register, this register provides status regarding the fault condition inputs.

If the **LATCH** bit in the **PWMnCTL** register is clear, the contents of the **PWMnFLTSTAT0** register are read-only (RO) and provide the current state of the **FAULTn** inputs.

If the **LATCH** bit in the **PWMnCTL** register is set, the contents of the **PWMnFLTSTAT0** register are read / write 1 to clear (R/W1C) and provide a latched version of the **FAULTn** inputs. In this mode, the register bits are cleared by writing a 1 to a set bit. The **FAULTn** inputs are recorded after their sense is adjusted in the generator.

The contents of this register can only be written if the fault source extensions are enabled (the **FLTSRC** bit in the **PWMnCTL** register is set).

#### PWM0 Fault Status 0 (PWM0FLTSTAT0)

Base 0x4002.8000

Offset 0x804

Type -, 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												FAULT3	FAULT2	FAULT1	FAULT0
Type	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	FAULT3	-	0	Fault Input 3 The same function as <b>FAULT0</b> , except applied for the <b>FAULT3</b> input.
2	FAULT2	-	0	Fault Input 2 The same function as <b>FAULT0</b> , except applied for the <b>FAULT2</b> input.
1	FAULT1	-	0	Fault Input 1 The same function as <b>FAULT0</b> , except applied for the <b>FAULT1</b> input.

Bit/Field	Name	Type	Reset	Description
0	FAULT0	-	0	<p>Fault Input 0</p> <p>If the <b>PWMnCTL</b> register <b>LATCH</b> bit is clear, this bit is RO and represents the current state of the <b>FAULT0</b> input signal after the logic sense adjustment.</p> <p>If the <b>PWMnCTL</b> register <b>LATCH</b> bit is set, this bit is R/W1C and represents a sticky version of the <b>FAULT0</b> input signal after the logic sense adjustment.</p> <ul style="list-style-type: none"><li>■ If <b>FAULT0</b> is set, the input transitioned to the active state previously.</li><li>■ If <b>FAULT0</b> is clear, the input has not transitioned to the active state since the last time it was cleared.</li><li>■ The <b>FAULT0</b> bit is cleared by writing it with the value 1.</li></ul>



**Register 83: PWM0 Fault Status 1 (PWM0FLTSTAT1), offset 0x808****Register 84: PWM1 Fault Status 1 (PWM1FLTSTAT1), offset 0x888****Register 85: PWM2 Fault Status 1 (PWM2FLTSTAT1), offset 0x908****Register 86: PWM3 Fault Status 1 (PWM3FLTSTAT1), offset 0x988**

Along with the **PWMnFLTSTAT0** register, this register provides status regarding the fault condition inputs.

If the **LATCH** bit in the **PWMnCTL** register is clear, the contents of the **PWMnFLTSTAT1** register are read-only (RO) and provide the current state of the digital comparator triggers.

If the **LATCH** bit in the **PWMnCTL** register is set, the contents of the **PWMnFLTSTAT1** register are read / write 1 to clear (R/W1C) and provide a latched version of the digital comparator triggers. In this mode, the register bits are cleared by writing a 1 to a set bit. The contents of this register can only be written if the fault source extensions are enabled (the **FLTSRC** bit in the **PWMnCTL** register is set).

**PWM0 Fault Status 1 (PWM0FLTSTAT1)**

Base 0x4002.8000  
Offset 0x808  
Type -, 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								DCMP7	DCMP6	DCMP5	DCMP4	DCMP3	DCMP2	DCMP1	DCMP0
Type	RO	RO	RO	RO	RO	RO	RO	RO	-	-	-	-	-	-	-	-
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	DCMP7	-	0	Digital Comparator 7 Trigger The same function as <b>DCMP0</b> , except applied for the Digital Comparator 7 trigger.
6	DCMP6	-	0	Digital Comparator 6 Trigger The same function as <b>DCMP0</b> , except applied for the Digital Comparator 6 trigger.
5	DCMP5	-	0	Digital Comparator 5 Trigger The same function as <b>DCMP0</b> , except applied for the Digital Comparator 5 trigger.
4	DCMP4	-	0	Digital Comparator 4 Trigger The same function as <b>DCMP0</b> , except applied for the Digital Comparator 4 trigger.

Bit/Field	Name	Type	Reset	Description
3	DCMP3	-	0	Digital Comparator 3 Trigger The same function as DCMP0, except applied for the Digital Comparator 3 trigger.
2	DCMP2	-	0	Digital Comparator 2 Trigger The same function as DCMP0, except applied for the Digital Comparator 2 trigger.
1	DCMP1	-	0	Digital Comparator 1 Trigger The same function as DCMP0, except applied for the Digital Comparator 1 trigger.
0	DCMP0	-	0	Digital Comparator 0 Trigger If the <b>PWMnCTL</b> register <b>LATCH</b> bit is clear, this bit represents the current state of the Digital Comparator 0 trigger input. If the <b>PWMnCTL</b> register <b>LATCH</b> bit is set, this bit represents a sticky version of the trigger. If the bit is set, the trigger transitioned to the active state previously. If clear, the trigger has not transitioned to the active state since the last time it was cleared. This bit is cleared by writing it with the value 1 (R/W1C).

## 22 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 LM3S5791 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.

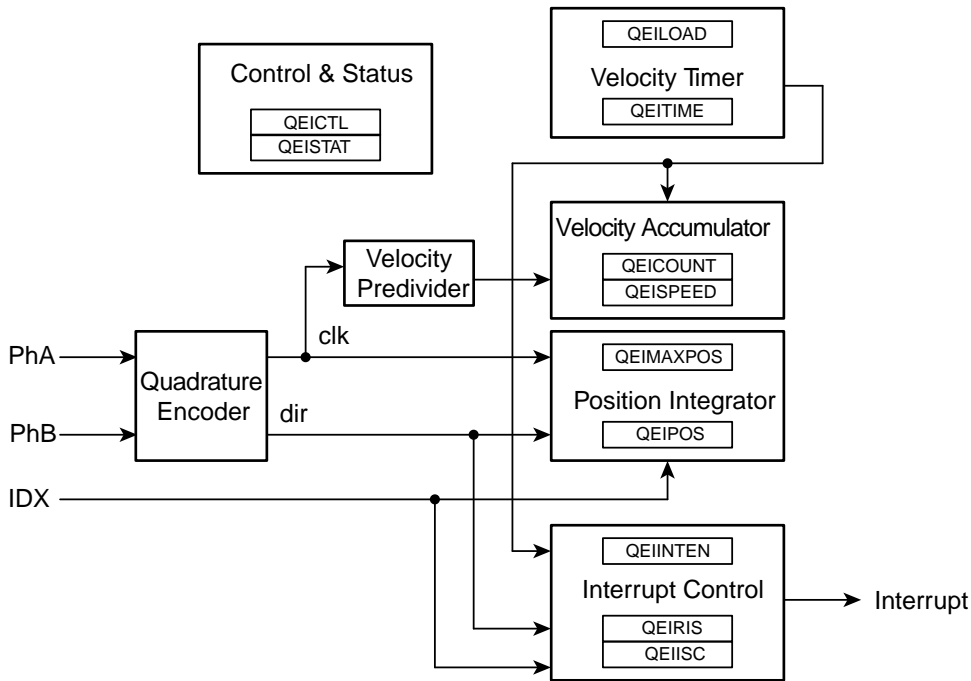
Each Stellaris<sup>®</sup> quadrature encoder has the following features:

- Position integrator that tracks the encoder position
- Programmable noise filter on the inputs
- Velocity capture using built-in timer
- Interrupt generation on:
  - Index pulse
  - Velocity-timer expiration
  - Direction change
  - Quadrature error detection

### 22.1 Block Diagram

Figure 22-1 on page 888 provides a block diagram of a Stellaris<sup>®</sup> QEI module.

Figure 22-1. QEI Block Diagram



## 22.2 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 input signals have a digital noise filter on them that can be enabled to prevent spurious operation. The noise filter requires that the inputs be stable for 3 consecutive clock cycles before updating the edge detector. The filter is enabled by the `FILTEN` bit in the **QEI Control (QEICTL)** register. The frequency of the input update is programmable using the `FILTCNT` bit field in the **QEICTL** register.

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 892).

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.

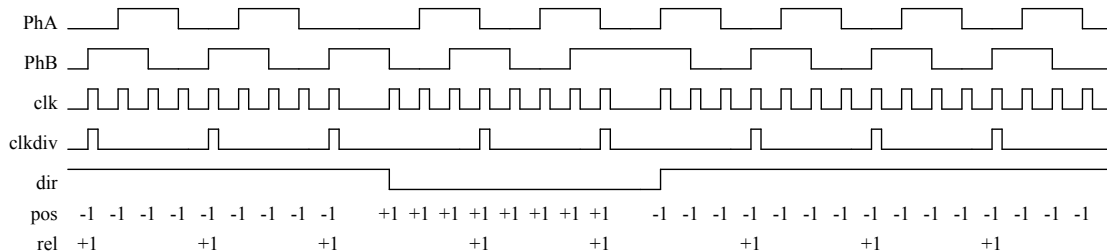
When  $ResMode$  is 0, the positional counter is reset when the index pulse is sensed. This limits the positional counter to the values  $[0:N-1]$ , where  $N$  is the number of phase edges in a full revolution of the encoder wheel. The **QEIMAXPOS** register must be programmed with  $N-1$  so that the 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 22-2 on page 889 shows how the Stellaris<sup>®</sup> 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 22-2. 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:

$$\text{rpm} = (\text{clock} * (2 \wedge \text{VelDiv}) * \text{Speed} * 60) \div (\text{Load} * \text{ppr} * \text{edges})$$

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.

## 22.3 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. See page 163.
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module. See page 175.
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.

## 22.4 Register Map

Table 22-1 on page 891 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

Note that the QEI module clock must be enabled before the registers can be programmed (see page 163).

**Table 22-1. QEI Register Map**

Offset	Name	Type	Reset	Description	See page
0x000	QEICTL	R/W	0x0000.0000	QEI Control	892
0x004	QEISTAT	RO	0x0000.0000	QEI Status	894
0x008	QEIPOS	R/W	0x0000.0000	QEI Position	895
0x00C	QEIMAXPOS	R/W	0x0000.0000	QEI Maximum Position	896
0x010	QEILOAD	R/W	0x0000.0000	QEI Timer Load	897
0x014	QEITIME	RO	0x0000.0000	QEI Timer	898
0x018	QEICOUNT	RO	0x0000.0000	QEI Velocity Counter	899
0x01C	QEISPEED	RO	0x0000.0000	QEI Velocity	900
0x020	QEIINTEN	R/W	0x0000.0000	QEI Interrupt Enable	901
0x024	QEIRIS	RO	0x0000.0000	QEI Raw Interrupt Status	902
0x028	QEIISC	R/W1C	0x0000.0000	QEI Interrupt Status and Clear	903

## 22.5 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												FILTCNT			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		FILTEN	STALLEN	INVI	INVB	INVA	VelDiv		VelEn	ResMode	CapMode	SigMode	Swap	Enable	
Type	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:20	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.
19:16	FILTCNT	R/W	0x0	Input Filter Pre-Scale Count  This field controls the frequency of the input update.
15:14	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	FILTEN	R/W	0	Enable Input Filter  When set, a digital noise filter is enabled on the input QEI signals. Inputs must be stable for 3 consecutive clock edges before the edge detector is updated.
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.



Bit/Field	Name	Type	Reset	Description																		
8:6	VelDiv	R/W	0x0	<p>Predivide Velocity</p> <p>A predivider of the input quadrature pulses before being applied to the QEICOUNT accumulator. This field can be set to the following values:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Predivider</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>÷1</td> </tr> <tr> <td>0x1</td> <td>÷2</td> </tr> <tr> <td>0x2</td> <td>÷4</td> </tr> <tr> <td>0x3</td> <td>÷8</td> </tr> <tr> <td>0x4</td> <td>÷16</td> </tr> <tr> <td>0x5</td> <td>÷32</td> </tr> <tr> <td>0x6</td> <td>÷64</td> </tr> <tr> <td>0x7</td> <td>÷128</td> </tr> </tbody> </table>	Value	Predivider	0x0	÷1	0x1	÷2	0x2	÷4	0x3	÷8	0x4	÷16	0x5	÷32	0x6	÷64	0x7	÷128
Value	Predivider																					
0x0	÷1																					
0x1	÷2																					
0x2	÷4																					
0x3	÷8																					
0x4	÷16																					
0x5	÷32																					
0x6	÷64																					
0x7	÷128																					
5	VelEn	R/W	0	<p>Capture Velocity</p> <p>When set, enables capture of the velocity of the quadrature encoder.</p>																		
4	ResMode	R/W	0	<p>Reset Mode</p> <p>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.</p>																		
3	CapMode	R/W	0	<p>Capture Mode</p> <p>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.</p>																		
2	SigMode	R/W	0	<p>Signal Mode</p> <p>When 1, the PhA and PhB signals are clock and direction; when 0, they are quadrature phase signals.</p>																		
1	Swap	R/W	0	<p>Swap Signals</p> <p>Swaps the PhA and PhB signals.</p>																		
0	Enable	R/W	0	<p>Enable QEI</p> <p>Enables the quadrature encoder module.</p>																		

## 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

	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															Direction	Error
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description						
31:2	reserved	RO	0x00	Software should not rely on the value of 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 <code>Direction</code> values are defined as follows:  <table border="0"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>Forward rotation</td> </tr> <tr> <td>1</td> <td>Reverse rotation</td> </tr> </table>	Value	Description	0	Forward rotation	1	Reverse rotation
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 (QEIP0S), 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 (QEIP0S)**

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

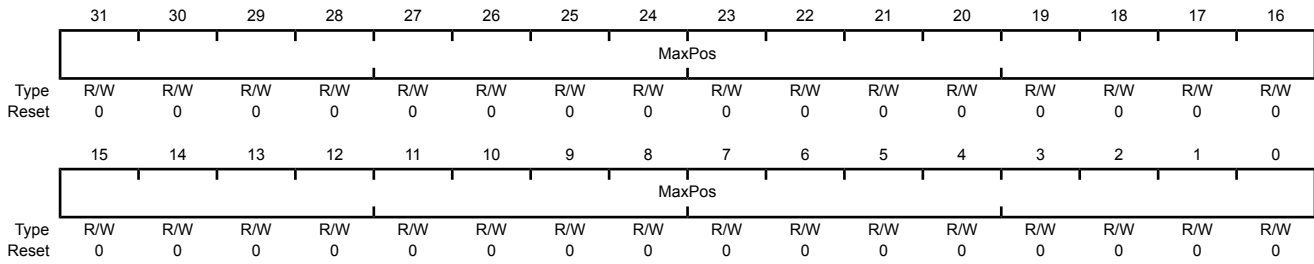
Bit/Field	Name	Type	Reset	Description
31:0	Position	R/W	0x00	Current Position Integrator Value The current value of the position integrator.

**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

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

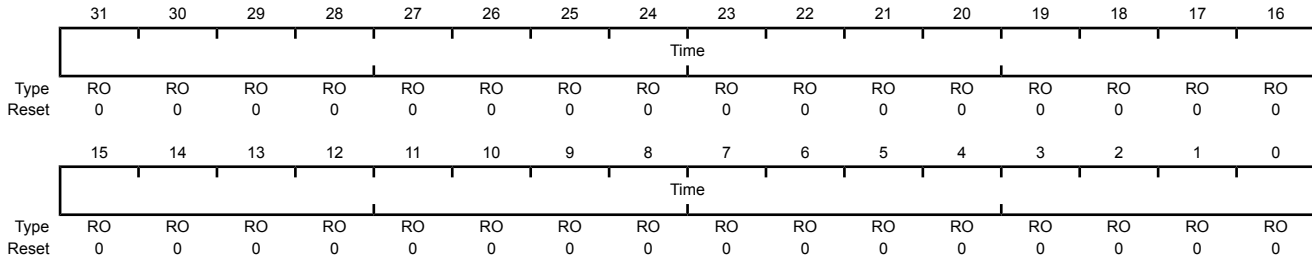
Bit/Field	Name	Type	Reset	Description
31:0	Load	R/W	0x00	Velocity Timer Load Value The load value for the velocity timer.

### 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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	Count															
Type	RO	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
	Count															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

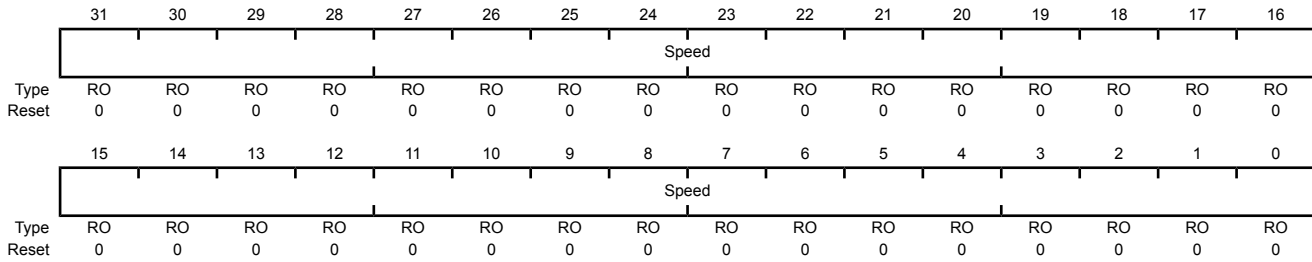
Bit/Field	Name	Type	Reset	Description
31:0	Count	RO	0x00	Velocity Pulse Count The running total of encoder pulses during this velocity timer period.

**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 **QEICTL** 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 (QEIINTEN), 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 (QEIINTEN)**

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	
	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												IntError	IntDir	IntTimer	IntIndex	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

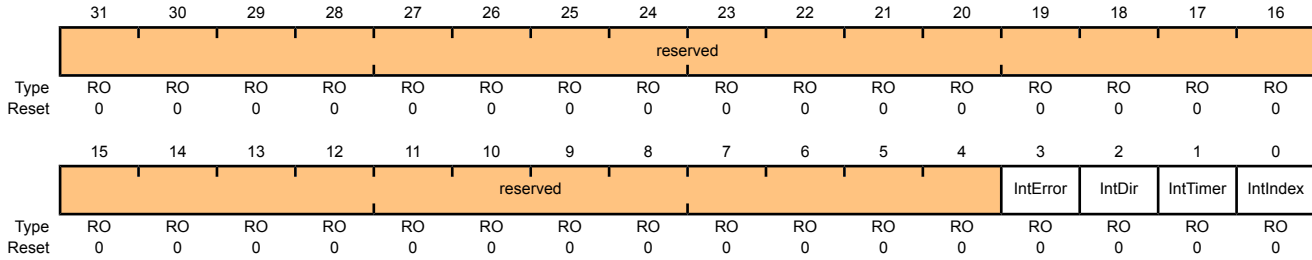
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	IntError	R/W	0	Phase Error Interrupt Enable When 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 (QEIISC), 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 (QEIISC)

QEI0 base: 0x4002.C000

QEI1 base: 0x4002.D000

Offset 0x028

Type R/W1C, reset 0x0000.0000

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

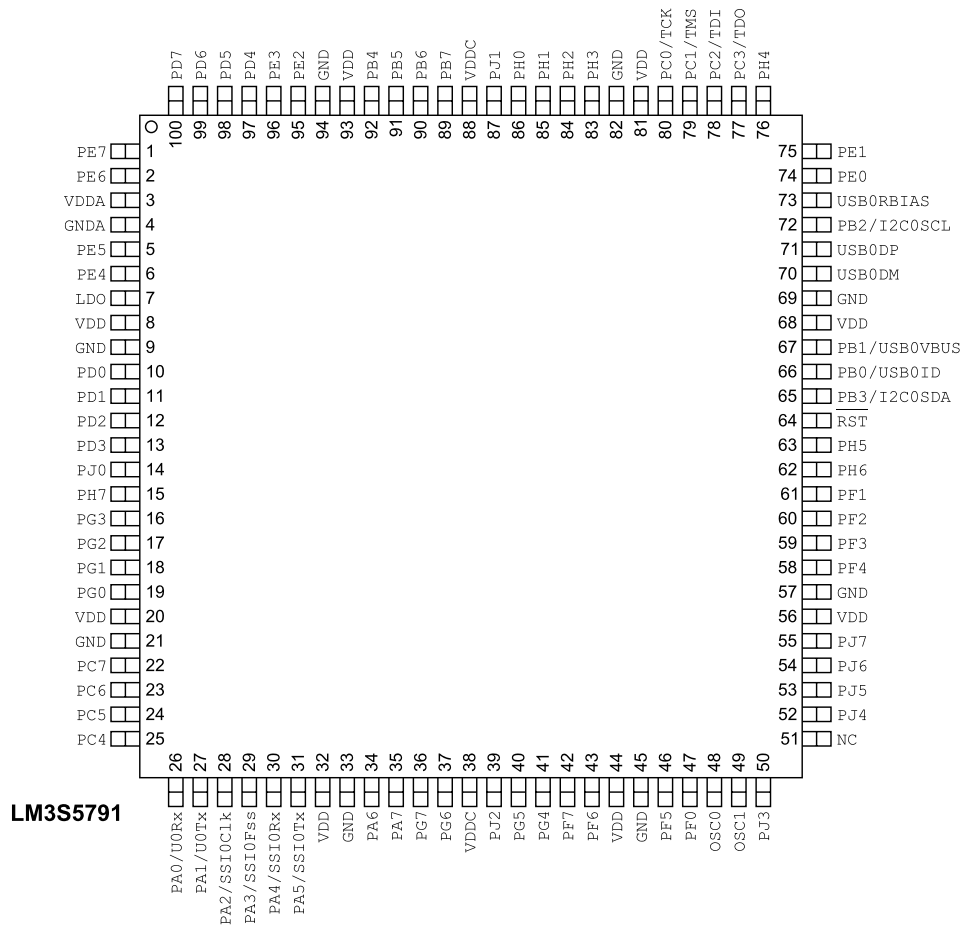
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	IntError	R/W1C	0	Phase Error Interrupt Indicates 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.

## 23 Pin Diagram

The LM3S5791 microcontroller pin diagram is shown below.

Each GPIO signal is identified by its GPIO port unless it defaults to an alternate function on reset. In this case, the GPIO port name is followed by the default alternate function. To see a complete list of possible functions for each pin, see Table 24-5 on page 937.

**Figure 23-1. 100-Pin LQFP Package Pin Diagram**



## 24 Signal Tables

The following tables list the signals available for each pin. Signals are configured as GPIOs on reset, except for those noted below. For a GPIO pin to be used for an alternate function, the corresponding bit in the **GPIOAFSEL** register (see page 304) must be set. Further pin muxing options are provided through the **PMC<sub>x</sub>** field in the **GPIOCTL** register (see page 321), which selects one of several available peripheral functions for that GPIO.

**Important:** All GPIO pins are configured as GPIOs by default with the exception of the pins shown in Table 9-1. A Power-On-Reset ( $\overline{\text{POR}}$ ) or asserting  $\overline{\text{RST}}$  puts the pins back to their default state.

**Table 24-1. GPIO Pins With Default Alternate Functions**

GPIO Pin	Default State	GPIOAFSEL Bit	GPIOCTL PMC <sub>x</sub> Bit Field
PA[1:0]	UART0	1	0x1
PA[5:2]	SSI0	1	0x1
PB[3:2]	I <sup>2</sup> C0	1	0x1
PC[3:0]	JTAG/SWD	1	0x3

Table 24-2 on page 905 shows the pin-to-signal-name mapping, including functional characteristics of the signals. Each possible alternate function is listed for each pin.

Table 24-3 on page 917 lists the signals in alphabetical order by signal name. If it is possible for a signal to be on multiple pins, each possible pin assignment is listed. The "Pin Mux" column indicates the GPIO and the encoding needed in the **PMC<sub>x</sub>** bit field in the **GPIOCTL** register.

Table 24-4 on page 928 groups the signals by functionality, except for GPIOs. If it is possible for a signal to be on multiple pins, each possible pin assignment is listed.

Table 24-5 on page 937 lists the GPIO pins and their alternate functions. The table heading "enc=" shows what the appropriate encoding for **PMC<sub>x</sub>** should be to select the function in that column (see page 321). Table entries that are shaded gray are the default values for the corresponding GPIO pin.

**Table 24-2. Signals by Pin Number**

Pin Number	Pin Name	Pin Type	Buffer Type <sup>a</sup>	Description
1	PE7	I/O	TTL	GPIO port E bit 7.
	AIN0	I	Analog	ADC 0 input.
	PWM5	O	TTL	PWM 5.
	C2o	O	TTL	Analog comparator 2 output.
	U1DCD	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
2	PE6	I/O	TTL	GPIO port E bit 6.
	AIN1	I	Analog	ADC 1 input.
	PWM4	O	TTL	PWM 4.
	C1o	O	TTL	Analog comparator 1 output.
	U1CTS	I	TTL	UART module 1 Clear To Send modem status input signal.

Pin Number	Pin Name	Pin Type	Buffer Type <sup>a</sup>	Description
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	GND A	-	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	PE5	I/O	TTL	GPIO port E bit 5.
	AIN2	I	Analog	ADC 2 input.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	I2S0TXSD	I/O	TTL	I <sup>2</sup> S module 0 transmit data.
6	PE4	I/O	TTL	GPIO port E bit 4.
	AIN3	I	Analog	ADC 3 input.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	Fault0	I	TTL	PWM Fault 0.
	U2Tx	O	TTL	UART module 2 transmit.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	I2S0TXWS	I/O	TTL	I <sup>2</sup> S module 0 transmit word select.
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 VDDC 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.
	AIN15	I	Analog	ADC 15 input.
	PWM0	O	TTL	PWM 0.
	CAN0Rx	I	TTL	CAN module 0 receive.
	IDX0	I	TTL	QEI module 0 index.
	U2Rx	I	TTL	UART module 2 receive.
	U1Rx	I	TTL	UART module 1 receive.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	I2S0RXSCK	I/O	TTL	I <sup>2</sup> S module 0 receive clock.
U1CTS	I	TTL	UART module 1 Clear To Send modem status input signal.	

Pin Number	Pin Name	Pin Type	Buffer Type <sup>a</sup>	Description
11	PD1	I/O	TTL	GPIO port D bit 1.
	AIN14	I	Analog	ADC 14 input.
	PWM1	O	TTL	PWM 1.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	PhA0	I	TTL	QEI module 0 phase A.
	U2Tx	O	TTL	UART module 2 transmit.
	U1Tx	O	TTL	UART module 1 transmit.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	I2S0RXWS	I/O	TTL	I <sup>2</sup> S module 0 receive word select.
	U1DCD	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
PhB1	I	TTL	QEI module 1 phase B.	
12	PD2	I/O	TTL	GPIO port D bit 2.
	AIN13	I	Analog	ADC 13 input.
	U1Rx	I	TTL	UART module 1 receive.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	PWM2	O	TTL	PWM 2.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	EPI0S20	I/O	TTL	EPI module 0 signal 20.
13	PD3	I/O	TTL	GPIO port D bit 3.
	AIN12	I	Analog	ADC 12 input.
	U1Tx	O	TTL	UART module 1 transmit.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	PWM3	O	TTL	PWM 3.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	EPI0S21	I/O	TTL	EPI module 0 signal 21.
14	PJ0	I/O	TTL	GPIO port J bit 0.
	EPI0S16	I/O	TTL	EPI module 0 signal 16.
	PWM0	O	TTL	PWM 0.
	I2C1SCL	I/O	OD	I <sup>2</sup> C module 1 clock.
15	PH7	I/O	TTL	GPIO port H bit 7.
	EPI0S27	I/O	TTL	EPI module 0 signal 27.
	PWM5	O	TTL	PWM 5.
	SSI1Tx	O	TTL	SSI module 1 transmit.
16	PG3	I/O	TTL	GPIO port G bit 3.
	PWM1	O	TTL	PWM 1.
	Fault2	I	TTL	PWM Fault 2.
	Fault0	I	TTL	PWM Fault 0.
	I2S0RXMCLK	I/O	TTL	I <sup>2</sup> S module 0 receive master clock.

Pin Number	Pin Name	Pin Type	Buffer Type <sup>a</sup>	Description
17	PG2	I/O	TTL	GPIO port G bit 2.
	PWM0	O	TTL	PWM 0.
	Fault0	I	TTL	PWM Fault 0.
	IDX1	I	TTL	QE1 module 1 index.
	I2S0RXSD	I/O	TTL	I <sup>2</sup> S module 0 receive data.
18	PG1	I/O	TTL	GPIO port G bit 1.
	U2Tx	O	TTL	UART module 2 transmit.
	PWM1	O	TTL	PWM 1.
	I2C1SDA	I/O	OD	I <sup>2</sup> C module 1 data.
	PWM5	O	TTL	PWM 5.
	EPI0S14	I/O	TTL	EPI module 0 signal 14.
19	PG0	I/O	TTL	GPIO port G bit 0.
	U2Rx	I	TTL	UART module 2 receive.
	PWM0	O	TTL	PWM 0.
	I2C1SCL	I/O	OD	I <sup>2</sup> C module 1 clock.
	PWM4	O	TTL	PWM 4.
	USB0EPEN	O	TTL	Used in Host mode to control an external power source to supply power to the USB bus.
	EPI0S13	I/O	TTL	EPI module 0 signal 13.
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.
	C2-	I	Analog	Analog comparator 2 negative input.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	PhB0	I	TTL	QE1 module 0 phase B.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	U1Tx	O	TTL	UART module 1 transmit.
	USB0PFLT	I	TTL	Used in Host mode by an external power source to indicate an error state by that power source.
	C1o	O	TTL	Analog comparator 1 output.
	EPI0S5	I/O	TTL	EPI module 0 signal 5.
23	PC6	I/O	TTL	GPIO port C bit 6.
	C2+	I	Analog	Analog comparator 2 positive input.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	PhB0	I	TTL	QE1 module 0 phase B.
	C2o	O	TTL	Analog comparator 2 output.
	PWM7	O	TTL	PWM 7.
	U1Rx	I	TTL	UART module 1 receive.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	USB0PFLT	I	TTL	Used in Host mode by an external power source to indicate an error state by that power source.
	EPI0S4	I/O	TTL	EPI module 0 signal 4.



Pin Number	Pin Name	Pin Type	Buffer Type <sup>a</sup>	Description
24	PC5	I/O	TTL	GPIO port C bit 5.
	C1+	I	Analog	Analog comparator 1 positive input.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	C1o	O	TTL	Analog comparator 1 output.
	C0o	O	TTL	Analog comparator 0 output.
	Fault2	I	TTL	PWM Fault 2.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	USB0EPEN	O	TTL	Used in Host mode to control an external power source to supply power to the USB bus.
	EPI0S3	I/O	TTL	EPI module 0 signal 3.
25	PC4	I/O	TTL	GPIO port C bit 4.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	PhA0	I	TTL	QEI module 0 phase A.
	PWM6	O	TTL	PWM 6.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	EPI0S2	I/O	TTL	EPI module 0 signal 2.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
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.
	I2C1SCL	I/O	OD	I <sup>2</sup> C module 1 clock.
	U1Rx	I	TTL	UART module 1 receive.
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.
	I2C1SDA	I/O	OD	I <sup>2</sup> C module 1 data.
	U1Tx	O	TTL	UART module 1 transmit.
28	PA2	I/O	TTL	GPIO port A bit 2.
	SSI0C1k	I/O	TTL	SSI module 0 clock.
	PWM4	O	TTL	PWM 4.
	I2S0RXSD	I/O	TTL	I <sup>2</sup> S module 0 receive data.
29	PA3	I/O	TTL	GPIO port A bit 3.
	SSI0Fss	I/O	TTL	SSI module 0 frame.
	PWM5	O	TTL	PWM 5.
	I2S0RXMCLK	I/O	TTL	I <sup>2</sup> S module 0 receive master clock.
30	PA4	I/O	TTL	GPIO port A bit 4.
	SSI0Rx	I	TTL	SSI module 0 receive.
	PWM6	O	TTL	PWM 6.
	CAN0Rx	I	TTL	CAN module 0 receive.
	I2S0TXSCK	I/O	TTL	I <sup>2</sup> S module 0 transmit clock.

Pin Number	Pin Name	Pin Type	Buffer Type <sup>a</sup>	Description
31	PA5	I/O	TTL	GPIO port A bit 5.
	SSI0Tx	O	TTL	SSI module 0 transmit.
	PWM7	O	TTL	PWM 7.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	I2S0TXWS	I/O	TTL	I <sup>2</sup> S module 0 transmit word select.
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.
	I2C1SCL	I/O	OD	I <sup>2</sup> C module 1 clock.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	PWM0	O	TTL	PWM 0.
	PWM4	O	TTL	PWM 4.
	CAN0Rx	I	TTL	CAN module 0 receive.
	USB0EPEN	O	TTL	Used in Host mode to control an external power source to supply power to the USB bus.
35	U1CTS	I	TTL	UART module 1 Clear To Send modem status input signal.
	PA7	I/O	TTL	GPIO port A bit 7.
	I2C1SDA	I/O	OD	I <sup>2</sup> C module 1 data.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	PWM1	O	TTL	PWM 1.
	PWM5	O	TTL	PWM 5.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	USB0PFLT	I	TTL	Used in Host mode by an external power source to indicate an error state by that power source.
36	U1DCD	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
	PG7	I/O	TTL	GPIO port G bit 7.
	PhB1	I	TTL	QEI module 1 phase B.
	PWM7	O	TTL	PWM 7.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
37	EPI0S31	I/O	TTL	EPI module 0 signal 31.
	PG6	I/O	TTL	GPIO port G bit 6.
	PhA1	I	TTL	QEI module 1 phase A.
	PWM6	O	TTL	PWM 6.
	Fault1	I	TTL	PWM Fault 1.
	I2S0RXWS	I/O	TTL	I <sup>2</sup> S module 0 receive word select.
38	U1RI	I	TTL	UART module 1 Ring Indicator modem status input signal.
	VDDC	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
39	PJ2	I/O	TTL	GPIO port J bit 2.
	EPI0S18	I/O	TTL	EPI module 0 signal 18.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	Fault0	I	TTL	PWM Fault 0.

Pin Number	Pin Name	Pin Type	Buffer Type <sup>a</sup>	Description
40	PG5	I/O	TTL	GPIO port G bit 5.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	IDX0	I	TTL	QEI module 0 index.
	Fault1	I	TTL	PWM Fault 1.
	PWM7	O	TTL	PWM 7.
	I2S0RXSCK	I/O	TTL	I <sup>2</sup> S module 0 receive clock.
	U1DTR	O	TTL	UART module 1 Data Terminal Ready modem status input signal.
41	PG4	I/O	TTL	GPIO port G bit 4.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	Fault1	I	TTL	PWM Fault 1.
	EPI0S15	I/O	TTL	EPI module 0 signal 15.
	PWM6	O	TTL	PWM 6.
	U1RI	I	TTL	UART module 1 Ring Indicator modem status input signal.
42	PF7	I/O	TTL	GPIO port F bit 7.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	PhB0	I	TTL	QEI module 0 phase B.
	EPI0S12	I/O	TTL	EPI module 0 signal 12.
	Fault1	I	TTL	PWM Fault 1.
43	PF6	I/O	TTL	GPIO port F bit 6.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	C2o	O	TTL	Analog comparator 2 output.
	PhA0	I	TTL	QEI module 0 phase A.
	I2S0TXMCLK	I/O	TTL	I <sup>2</sup> S module 0 transmit master clock.
	U1RTS	O	TTL	UART module 1 Request to Send modem output control line.
44	VDD	-	Power	Positive supply for I/O and some logic.
45	GND	-	Power	Ground reference for logic and I/O pins.
46	PF5	I/O	TTL	GPIO port F bit 5.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	C1o	O	TTL	Analog comparator 1 output.
	EPI0S15	I/O	TTL	EPI module 0 signal 15.
	SSI1Tx	O	TTL	SSI module 1 transmit.
47	PF0	I/O	TTL	GPIO port F bit 0.
	CAN1Rx	I	TTL	CAN module 1 receive.
	PhB0	I	TTL	QEI module 0 phase B.
	PWM0	O	TTL	PWM 0.
	I2S0TXSD	I/O	TTL	I <sup>2</sup> S module 0 transmit data.
	U1DSR	I	TTL	UART module 1 Data Set Ready modem output control line.
48	OSC0	I	Analog	Main oscillator crystal input or an external clock reference input.
49	OSC1	O	Analog	Main oscillator crystal output.

Pin Number	Pin Name	Pin Type	Buffer Type <sup>a</sup>	Description
50	PJ3	I/O	TTL	GPIO port J bit 3.
	EPI0S19	I/O	TTL	EPI module 0 signal 19.
	U1CTS	I	TTL	UART module 1 Clear To Send modem status input signal.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
51	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
52	PJ4	I/O	TTL	GPIO port J bit 4.
	EPI0S28	I/O	TTL	EPI module 0 signal 28.
	U1DCD	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
53	PJ5	I/O	TTL	GPIO port J bit 5.
	EPI0S29	I/O	TTL	EPI module 0 signal 29.
	U1DSR	I	TTL	UART module 1 Data Set Ready modem output control line.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
54	PJ6	I/O	TTL	GPIO port J bit 6.
	EPI0S30	I/O	TTL	EPI module 0 signal 30.
	U1RTS	O	TTL	UART module 1 Request to Send modem output control line.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
55	PJ7	I/O	TTL	GPIO port J bit 7.
	U1DTR	O	TTL	UART module 1 Data Terminal Ready modem status input signal.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
56	VDD	-	Power	Positive supply for I/O and some logic.
57	GND	-	Power	Ground reference for logic and I/O pins.
58	PF4	I/O	TTL	GPIO port F bit 4.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	C0o	O	TTL	Analog comparator 0 output.
	Fault0	I	TTL	PWM Fault 0.
	EPI0S12	I/O	TTL	EPI module 0 signal 12.
	SSI1Rx	I	TTL	SSI module 1 receive.
59	PF3	I/O	TTL	GPIO port F bit 3.
	LED0	O	TTL	MII LED 0.
	PWM5	O	TTL	PWM 5.
	PWM3	O	TTL	PWM 3.
	SSI1Fss	I/O	TTL	SSI module 1 frame.
60	PF2	I/O	TTL	GPIO port F bit 2.
	LED1	O	TTL	MII LED 1.
	PWM4	O	TTL	PWM 4.
	PWM2	O	TTL	PWM 2.
	SSI1Clk	I/O	TTL	SSI module 1 clock.

Pin Number	Pin Name	Pin Type	Buffer Type <sup>a</sup>	Description
61	PF1	I/O	TTL	GPIO port F bit 1.
	CAN1Tx	O	TTL	CAN module 1 transmit.
	IDX1	I	TTL	QEI module 1 index.
	PWM1	O	TTL	PWM 1.
	I2S0TXMCLK	I/O	TTL	I <sup>2</sup> S module 0 transmit master clock.
	U1RTS	O	TTL	UART module 1 Request to Send modem output control line.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
62	PH6	I/O	TTL	GPIO port H bit 6.
	EPI0S26	I/O	TTL	EPI module 0 signal 26.
	PWM4	O	TTL	PWM 4.
	SSI1Rx	I	TTL	SSI module 1 receive.
63	PH5	I/O	TTL	GPIO port H bit 5.
	EPI0S11	I/O	TTL	EPI module 0 signal 11.
	Fault2	I	TTL	PWM Fault 2.
	SSI1Fss	I/O	TTL	SSI module 1 frame.
64	RST	I	TTL	System reset input.
65	PB3	I/O	TTL	GPIO port B bit 3.
	I2C0SDA	I/O	OD	I <sup>2</sup> C module 0 data.
	Fault0	I	TTL	PWM Fault 0.
	Fault3	I	TTL	PWM Fault 3.
	USB0PFLT	I	TTL	Used in Host mode by an external power source to indicate an error state by that power source.
66	PB0	I/O	TTL	GPIO port B bit 0.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	PWM2	O	TTL	PWM 2.
	U1Rx	I	TTL	UART module 1 receive.
	USB0ID	I	Analog	This signal senses the state of the USB ID signal. The USB PHY enables an integrated pull-up, and an external element (USB connector) indicates the initial state of the USB controller (pulled down is an A device and not pulled down is a B device).
67	PB1	I/O	TTL	GPIO port B bit 1.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	PWM3	O	TTL	PWM 3.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	U1Tx	O	TTL	UART module 1 transmit.
	USB0VBUS	I/O	Analog	This signal is used during the session negotiation protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing.
68	VDD	-	Power	Positive supply for I/O and some logic.
69	GND	-	Power	Ground reference for logic and I/O pins.
70	USB0DM	I/O	Analog	Bidirectional differential data pin (D- per USB specification).
71	USB0DP	I/O	Analog	Bidirectional differential data pin (D+ per USB specification).

Pin Number	Pin Name	Pin Type	Buffer Type <sup>a</sup>	Description
72	PB2	I/O	TTL	GPIO port B bit 2.
	I2COSCL	I/O	OD	I <sup>2</sup> C module 0 clock.
	IDX0	I	TTL	QEI module 0 index.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	USB0EPEN	O	TTL	Used in Host mode to control an external power source to supply power to the USB bus.
73	USB0RBIAS	O	Analog	9.1-kΩ resistor (1% precision) used internally for USB analog circuitry.
74	PE0	I/O	TTL	GPIO port E bit 0.
	PWM4	O	TTL	PWM 4.
	SSI1Clk	I/O	TTL	SSI module 1 clock.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	EPI0S8	I/O	TTL	EPI module 0 signal 8.
	USB0PFLT	I	TTL	Used in Host mode by an external power source to indicate an error state by that power source.
75	PE1	I/O	TTL	GPIO port E bit 1.
	PWM5	O	TTL	PWM 5.
	SSI1Fss	I/O	TTL	SSI module 1 frame.
	Fault0	I	TTL	PWM Fault 0.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
76	EPI0S9	I/O	TTL	EPI module 0 signal 9.
	PH4	I/O	TTL	GPIO port H bit 4.
	USB0PFLT	I	TTL	Used in Host mode by an external power source to indicate an error state by that power source.
	EPI0S10	I/O	TTL	EPI module 0 signal 10.
77	SSI1Clk	I/O	TTL	SSI module 1 clock.
	PC3	I/O	TTL	GPIO port C bit 3.
	TDO	O	TTL	JTAG TDO and SWO.
78	SWO	O	TTL	JTAG TDO and SWO.
	PC2	I/O	TTL	GPIO port C bit 2.
79	TDI	I	TTL	JTAG TDI.
	PC1	I/O	TTL	GPIO port C bit 1.
	TMS	I	TTL	JTAG TMS and SWDIO.
80	SWDIO	I/O	TTL	JTAG TMS and SWDIO.
	PC0	I/O	TTL	GPIO port C bit 0.
	TCK	I	TTL	JTAG/SWD CLK.
81	SWCLK	I	TTL	JTAG/SWD CLK.
	VDD	-	Power	Positive supply for I/O and some logic.
82	GND	-	Power	Ground reference for logic and I/O pins.

Pin Number	Pin Name	Pin Type	Buffer Type <sup>a</sup>	Description
83	PH3	I/O	TTL	GPIO port H bit 3.
	PhB0	I	TTL	QEI module 0 phase B.
	Fault0	I	TTL	PWM Fault 0.
	USB0EPEN	O	TTL	Used in Host mode to control an external power source to supply power to the USB bus.
	EPI0S0	I/O	TTL	EPI module 0 signal 0.
84	PH2	I/O	TTL	GPIO port H bit 2.
	IDX1	I	TTL	QEI module 1 index.
	C1o	O	TTL	Analog comparator 1 output.
	Fault3	I	TTL	PWM Fault 3.
	EPI0S1	I/O	TTL	EPI module 0 signal 1.
85	PH1	I/O	TTL	GPIO port H bit 1.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	PWM3	O	TTL	PWM 3.
	EPI0S7	I/O	TTL	EPI module 0 signal 7.
	PWM5	O	TTL	PWM 5.
86	PH0	I/O	TTL	GPIO port H bit 0.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	PWM2	O	TTL	PWM 2.
	EPI0S6	I/O	TTL	EPI module 0 signal 6.
	PWM4	O	TTL	PWM 4.
87	PJ1	I/O	TTL	GPIO port J bit 1.
	EPI0S17	I/O	TTL	EPI module 0 signal 17.
	USB0PFLT	I	TTL	Used in Host mode by an external power source to indicate an error state by that power source.
	PWM1	O	TTL	PWM 1.
	I2C1SDA	I/O	OD	I <sup>2</sup> C module 1 data.
88	VDDC	-	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.
	NMI	I	TTL	Non-maskable interrupt.

Pin Number	Pin Name	Pin Type	Buffer Type <sup>a</sup>	Description
90	PB6	I/O	TTL	GPIO port B bit 6.
	VREFA	I	Analog	This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AIN <sub>n</sub> signal is converted to 1023. The VREFA input is limited to the range specified in Table 26-2 on page 941.
	C0+	I	Analog	Analog comparator 0 positive input.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	C0o	O	TTL	Analog comparator 0 output.
	Fault1	I	TTL	PWM Fault 1.
	IDX0	I	TTL	QEI module 0 index.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
I2S0TXSCK	I/O	TTL	I <sup>2</sup> S module 0 transmit clock.	
91	PB5	I/O	TTL	GPIO port B bit 5.
	AIN11	I	Analog	ADC 11 input.
	C1-	I	Analog	Analog comparator 1 negative input.
	C0o	O	TTL	Analog comparator 0 output.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	U1Tx	O	TTL	UART module 1 transmit.
EPI0S22	I/O	TTL	EPI module 0 signal 22.	
92	PB4	I/O	TTL	GPIO port B bit 4.
	AIN10	I	Analog	ADC 10 input.
	C0-	I	Analog	Analog comparator 0 negative input.
	U2Rx	I	TTL	UART module 2 receive.
	CAN0Rx	I	TTL	CAN module 0 receive.
	IDX0	I	TTL	QEI module 0 index.
	U1Rx	I	TTL	UART module 1 receive.
	EPI0S23	I/O	TTL	EPI module 0 signal 23.
93	VDD	-	Power	Positive supply for I/O and some logic.
94	GND	-	Power	Ground reference for logic and I/O pins.
95	PE2	I/O	TTL	GPIO port E bit 2.
	AIN9	I	Analog	ADC 9 input.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	SSI1Rx	I	TTL	SSI module 1 receive.
	PhB1	I	TTL	QEI module 1 phase B.
	PhA0	I	TTL	QEI module 0 phase A.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	EPI0S24	I/O	TTL	EPI module 0 signal 24.



Pin Number	Pin Name	Pin Type	Buffer Type <sup>a</sup>	Description
96	PE3	I/O	TTL	GPIO port E bit 3.
	AIN8	I	Analog	ADC 8 input.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	SSI1Tx	O	TTL	SSI module 1 transmit.
	PhA1	I	TTL	QEI module 1 phase A.
	PhB0	I	TTL	QEI module 0 phase B.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	EPI0S25	I/O	TTL	EPI module 0 signal 25.
97	PD4	I/O	TTL	GPIO port D bit 4.
	AIN7	I	Analog	ADC 7 input.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	I2S0RXSD	I/O	TTL	I <sup>2</sup> S module 0 receive data.
	U1RI	I	TTL	UART module 1 Ring Indicator modem status input signal.
	EPI0S19	I/O	TTL	EPI module 0 signal 19.
98	PD5	I/O	TTL	GPIO port D bit 5.
	AIN6	I	Analog	ADC 6 input.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	I2S0RXMCLK	I/O	TTL	I <sup>2</sup> S module 0 receive master clock.
	U2Rx	I	TTL	UART module 2 receive.
	EPI0S28	I/O	TTL	EPI module 0 signal 28.
99	PD6	I/O	TTL	GPIO port D bit 6.
	AIN5	I	Analog	ADC 5 input.
	Fault0	I	TTL	PWM Fault 0.
	I2S0TXSCK	I/O	TTL	I <sup>2</sup> S module 0 transmit clock.
	U2Tx	O	TTL	UART module 2 transmit.
	EPI0S29	I/O	TTL	EPI module 0 signal 29.
100	PD7	I/O	TTL	GPIO port D bit 7.
	AIN4	I	Analog	ADC 4 input.
	IDX0	I	TTL	QEI module 0 index.
	COo	O	TTL	Analog comparator 0 output.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	I2S0TXWS	I/O	TTL	I <sup>2</sup> S module 0 transmit word select.
	U1DTR	O	TTL	UART module 1 Data Terminal Ready modem status input signal.
	EPI0S30	I/O	TTL	EPI module 0 signal 30.

a. The TTL designation indicates the pin is TTL-compatible.

**Table 24-3. Signals by Signal Name**

Pin Name	Pin Number	Pin Mux	Pin Type	Buffer Type <sup>a</sup>	Description
AIN0	1	-	I	Analog	ADC 0 input.
AIN1	2	-	I	Analog	ADC 1 input.

Pin Name	Pin Number	Pin Mux	Pin Type	Buffer Type <sup>a</sup>	Description
AIN2	5	-	I	Analog	ADC 2 input.
AIN3	6	-	I	Analog	ADC 3 input.
AIN4	100	-	I	Analog	ADC 4 input.
AIN5	99	-	I	Analog	ADC 5 input.
AIN6	98	-	I	Analog	ADC 6 input.
AIN7	97	-	I	Analog	ADC 7 input.
AIN8	96	-	I	Analog	ADC 8 input.
AIN9	95	-	I	Analog	ADC 9 input.
AIN10	92	-	I	Analog	ADC 10 input.
AIN11	91	-	I	Analog	ADC 11 input.
AIN12	13	-	I	Analog	ADC 12 input.
AIN13	12	-	I	Analog	ADC 13 input.
AIN14	11	-	I	Analog	ADC 14 input.
AIN15	10	-	I	Analog	ADC 15 input.
C0+	90	-	I	Analog	Analog comparator 0 positive input.
C0-	92	-	I	Analog	Analog comparator 0 negative input.
C0o	24 58 90 91 100	PC5 (3) PF4 (2) PB6 (3) PB5 (1) PD7 (2)	O	TTL	Analog comparator 0 output.
C1+	24	-	I	Analog	Analog comparator 1 positive input.
C1-	91	-	I	Analog	Analog comparator 1 negative input.
C1o	2 22 24 46 84	PE6 (2) PC7 (7) PC5 (2) PF5 (2) PH2 (2)	O	TTL	Analog comparator 1 output.
C2+	23	-	I	Analog	Analog comparator 2 positive input.
C2-	22	-	I	Analog	Analog comparator 2 negative input.
C2o	1 23 43	PE7 (2) PC6 (3) PF6 (2)	O	TTL	Analog comparator 2 output.
CAN0Rx	10 30 34 92	PD0 (2) PA4 (5) PA6 (6) PB4 (5)	I	TTL	CAN module 0 receive.
CAN0Tx	11 31 35 91	PD1 (2) PA5 (5) PA7 (6) PB5 (5)	O	TTL	CAN module 0 transmit.
CAN1Rx	47	PF0 (1)	I	TTL	CAN module 1 receive.
CAN1Tx	61	PF1 (1)	O	TTL	CAN module 1 transmit.

Pin Name	Pin Number	Pin Mux	Pin Type	Buffer Type <sup>a</sup>	Description
CCP0	13 22 23 39 55 58 66 72 91 97	PD3 (4) PC7 (4) PC6 (6) PJ2 (9) PJ7 (10) PF4 (1) PB0 (1) PB2 (5) PB5 (4) PD4 (1)	I/O	TTL	Capture/Compare/PWM 0.
CCP1	24 25 34 43 54 67 90 96 100	PC5 (1) PC4 (9) PA6 (2) PF6 (1) PJ6 (10) PB1 (4) PB6 (1) PE3 (1) PD7 (3)	I/O	TTL	Capture/Compare/PWM 1.
CCP2	6 11 25 46 53 67 75 91 95 98	PE4 (6) PD1 (10) PC4 (5) PF5 (1) PJ5 (10) PB1 (1) PE1 (4) PB5 (6) PE2 (5) PD5 (1)	I/O	TTL	Capture/Compare/PWM 2.
CCP3	6 23 24 35 41 61 72 74 97	PE4 (1) PC6 (1) PC5 (5) PA7 (7) PG4 (1) PF1 (10) PB2 (4) PE0 (3) PD4 (2)	I/O	TTL	Capture/Compare/PWM 3.
CCP4	22 25 35 42 52 95 98	PC7 (1) PC4 (6) PA7 (2) PF7 (1) PJ4 (10) PE2 (1) PD5 (2)	I/O	TTL	Capture/Compare/PWM 4.
CCP5	5 12 25 36 40 90 91	PE5 (1) PD2 (4) PC4 (1) PG7 (8) PG5 (1) PB6 (6) PB5 (2)	I/O	TTL	Capture/Compare/PWM 5.
CCP6	10 12 50 75 86 91	PD0 (6) PD2 (2) PJ3 (10) PE1 (5) PH0 (1) PB5 (3)	I/O	TTL	Capture/Compare/PWM 6.

Pin Name	Pin Number	Pin Mux	Pin Type	Buffer Type <sup>a</sup>	Description
CCP7	11 13 85 90 96	PD1 (6) PD3 (2) PH1 (1) PB6 (2) PE3 (5)	I/O	TTL	Capture/Compare/PWM 7.
EPI0S0	83	PH3 (8)	I/O	TTL	EPI module 0 signal 0.
EPI0S1	84	PH2 (8)	I/O	TTL	EPI module 0 signal 1.
EPI0S2	25	PC4 (8)	I/O	TTL	EPI module 0 signal 2.
EPI0S3	24	PC5 (8)	I/O	TTL	EPI module 0 signal 3.
EPI0S4	23	PC6 (8)	I/O	TTL	EPI module 0 signal 4.
EPI0S5	22	PC7 (8)	I/O	TTL	EPI module 0 signal 5.
EPI0S6	86	PH0 (8)	I/O	TTL	EPI module 0 signal 6.
EPI0S7	85	PH1 (8)	I/O	TTL	EPI module 0 signal 7.
EPI0S8	74	PE0 (8)	I/O	TTL	EPI module 0 signal 8.
EPI0S9	75	PE1 (8)	I/O	TTL	EPI module 0 signal 9.
EPI0S10	76	PH4 (8)	I/O	TTL	EPI module 0 signal 10.
EPI0S11	63	PH5 (8)	I/O	TTL	EPI module 0 signal 11.
EPI0S12	42 58	PF7 (8) PF4 (8)	I/O	TTL	EPI module 0 signal 12.
EPI0S13	19	PG0 (8)	I/O	TTL	EPI module 0 signal 13.
EPI0S14	18	PG1 (8)	I/O	TTL	EPI module 0 signal 14.
EPI0S15	41 46	PG4 (8) PF5 (8)	I/O	TTL	EPI module 0 signal 15.
EPI0S16	14	PJ0 (8)	I/O	TTL	EPI module 0 signal 16.
EPI0S17	87	PJ1 (8)	I/O	TTL	EPI module 0 signal 17.
EPI0S18	39	PJ2 (8)	I/O	TTL	EPI module 0 signal 18.
EPI0S19	50 97	PJ3 (8) PD4 (10)	I/O	TTL	EPI module 0 signal 19.
EPI0S20	12	PD2 (8)	I/O	TTL	EPI module 0 signal 20.
EPI0S21	13	PD3 (8)	I/O	TTL	EPI module 0 signal 21.
EPI0S22	91	PB5 (8)	I/O	TTL	EPI module 0 signal 22.
EPI0S23	92	PB4 (8)	I/O	TTL	EPI module 0 signal 23.
EPI0S24	95	PE2 (8)	I/O	TTL	EPI module 0 signal 24.
EPI0S25	96	PE3 (8)	I/O	TTL	EPI module 0 signal 25.
EPI0S26	62	PH6 (8)	I/O	TTL	EPI module 0 signal 26.
EPI0S27	15	PH7 (8)	I/O	TTL	EPI module 0 signal 27.
EPI0S28	52 98	PJ4 (8) PD5 (10)	I/O	TTL	EPI module 0 signal 28.
EPI0S29	53 99	PJ5 (8) PD6 (10)	I/O	TTL	EPI module 0 signal 29.
EPI0S30	54 100	PJ6 (8) PD7 (10)	I/O	TTL	EPI module 0 signal 30.
EPI0S31	36	PG7 (9)	I/O	TTL	EPI module 0 signal 31.

Pin Name	Pin Number	Pin Mux	Pin Type	Buffer Type <sup>a</sup>	Description
Fault0	6 16 17 39 58 65 75 83 99	PE4 (4) PG3 (8) PG2 (4) PJ2 (10) PF4 (4) PB3 (2) PE1 (3) PH3 (2) PD6 (1)	I	TTL	PWM Fault 0.
Fault1	37 40 41 42 90	PG6 (8) PG5 (5) PG4 (4) PF7 (9) PB6 (4)	I	TTL	PWM Fault 1.
Fault2	16 24 63	PG3 (4) PC5 (4) PH5 (10)	I	TTL	PWM Fault 2.
Fault3	65 84	PB3 (4) PH2 (4)	I	TTL	PWM Fault 3.
GND	9 21 33 45 57 69 82 94	-	-	Power	Ground reference for logic and I/O pins.
GND <sub>A</sub>	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 V <sub>DD</sub> from affecting the analog functions.
I2C0SCL	72	PB2 (1)	I/O	OD	I <sup>2</sup> C module 0 clock.
I2C0SDA	65	PB3 (1)	I/O	OD	I <sup>2</sup> C module 0 data.
I2C1SCL	14 19 26 34	PJ0 (11) PG0 (3) PA0 (8) PA6 (1)	I/O	OD	I <sup>2</sup> C module 1 clock.
I2C1SDA	18 27 35 87	PG1 (3) PA1 (8) PA7 (1) PJ1 (11)	I/O	OD	I <sup>2</sup> C module 1 data.
I2S0RXMCLK	16 29 98	PG3 (9) PA3 (9) PD5 (8)	I/O	TTL	I <sup>2</sup> S module 0 receive master clock.
I2S0RXSCK	10 40	PD0 (8) PG5 (9)	I/O	TTL	I <sup>2</sup> S module 0 receive clock.
I2S0RXSD	17 28 97	PG2 (9) PA2 (9) PD4 (8)	I/O	TTL	I <sup>2</sup> S module 0 receive data.
I2S0RXWS	11 37	PD1 (8) PG6 (9)	I/O	TTL	I <sup>2</sup> S module 0 receive word select.
I2S0TXMCLK	43 61	PF6 (9) PF1 (8)	I/O	TTL	I <sup>2</sup> S module 0 transmit master clock.

Pin Name	Pin Number	Pin Mux	Pin Type	Buffer Type <sup>a</sup>	Description
I2S0TXSCK	30 90 99	PA4 (9) PB6 (9) PD6 (8)	I/O	TTL	I <sup>2</sup> S module 0 transmit clock.
I2S0TXSD	5 47	PE5 (9) PF0 (8)	I/O	TTL	I <sup>2</sup> S module 0 transmit data.
I2S0TXWS	6 31 100	PE4 (9) PA5 (9) PD7 (8)	I/O	TTL	I <sup>2</sup> S module 0 transmit word select.
IDX0	10 40 72 90 92 100	PD0 (3) PG5 (4) PB2 (2) PB6 (5) PB4 (6) PD7 (1)	I	TTL	QEI module 0 index.
IDX1	17 61 84	PG2 (8) PF1 (2) PH2 (1)	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 VDDC pins at the board level in addition to the decoupling capacitor(s).
LED0	59	PF3 (1)	O	TTL	MII LED 0.
LED1	60	PF2 (1)	O	TTL	MII LED 1.
NC	51	-	-	-	No connect. Leave the pin electrically unconnected/isolated.
NMI	89	PB7 (4)	I	TTL	Non-maskable interrupt.
OSC0	48	-	I	Analog	Main oscillator crystal input or an external clock reference input.
OSC1	49	-	O	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.
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	72	-	I/O	TTL	GPIO port B bit 2.
PB3	65	-	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.

Pin Name	Pin Number	Pin Mux	Pin Type	Buffer Type <sup>a</sup>	Description
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	97	-	I/O	TTL	GPIO port D bit 4.
PD5	98	-	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	74	-	I/O	TTL	GPIO port E bit 0.
PE1	75	-	I/O	TTL	GPIO port E bit 1.
PE2	95	-	I/O	TTL	GPIO port E bit 2.
PE3	96	-	I/O	TTL	GPIO port E bit 3.
PE4	6	-	I/O	TTL	GPIO port E bit 4.
PE5	5	-	I/O	TTL	GPIO port E bit 5.
PE6	2	-	I/O	TTL	GPIO port E bit 6.
PE7	1	-	I/O	TTL	GPIO port E bit 7.
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.
PF4	58	-	I/O	TTL	GPIO port F bit 4.
PF5	46	-	I/O	TTL	GPIO port F bit 5.
PF6	43	-	I/O	TTL	GPIO port F bit 6.
PF7	42	-	I/O	TTL	GPIO port F bit 7.
PG0	19	-	I/O	TTL	GPIO port G bit 0.
PG1	18	-	I/O	TTL	GPIO port G bit 1.
PG2	17	-	I/O	TTL	GPIO port G bit 2.
PG3	16	-	I/O	TTL	GPIO port G bit 3.
PG4	41	-	I/O	TTL	GPIO port G bit 4.
PG5	40	-	I/O	TTL	GPIO port G bit 5.
PG6	37	-	I/O	TTL	GPIO port G bit 6.
PG7	36	-	I/O	TTL	GPIO port G bit 7.
PH0	86	-	I/O	TTL	GPIO port H bit 0.
PH1	85	-	I/O	TTL	GPIO port H bit 1.
PH2	84	-	I/O	TTL	GPIO port H bit 2.

Pin Name	Pin Number	Pin Mux	Pin Type	Buffer Type <sup>a</sup>	Description
PH3	83	-	I/O	TTL	GPIO port H bit 3.
PH4	76	-	I/O	TTL	GPIO port H bit 4.
PH5	63	-	I/O	TTL	GPIO port H bit 5.
PH6	62	-	I/O	TTL	GPIO port H bit 6.
PH7	15	-	I/O	TTL	GPIO port H bit 7.
PJ0	14	-	I/O	TTL	GPIO port J bit 0.
PJ1	87	-	I/O	TTL	GPIO port J bit 1.
PJ2	39	-	I/O	TTL	GPIO port J bit 2.
PJ3	50	-	I/O	TTL	GPIO port J bit 3.
PJ4	52	-	I/O	TTL	GPIO port J bit 4.
PJ5	53	-	I/O	TTL	GPIO port J bit 5.
PJ6	54	-	I/O	TTL	GPIO port J bit 6.
PJ7	55	-	I/O	TTL	GPIO port J bit 7.
PWM0	10 14 17 19 34 47	PD0 (1) PJ0 (10) PG2 (1) PG0 (2) PA6 (4) PF0 (3)	O	TTL	PWM 0.
PWM1	11 16 18 35 61 87	PD1 (1) PG3 (1) PG1 (2) PA7 (4) PF1 (3) PJ1 (10)	O	TTL	PWM 1.
PWM2	12 60 66 86	PD2 (3) PF2 (4) PB0 (2) PH0 (2)	O	TTL	PWM 2.
PWM3	13 59 67 85	PD3 (3) PF3 (4) PB1 (2) PH1 (2)	O	TTL	PWM 3.
PWM4	2 19 28 34 60 62 74 86	PE6 (1) PG0 (4) PA2 (4) PA6 (5) PF2 (2) PH6 (10) PE0 (1) PH0 (9)	O	TTL	PWM 4.
PWM5	1 15 18 29 35 59 75 85	PE7 (1) PH7 (10) PG1 (4) PA3 (4) PA7 (5) PF3 (2) PE1 (1) PH1 (9)	O	TTL	PWM 5.
PWM6	25 30 37 41	PC4 (4) PA4 (4) PG6 (4) PG4 (9)	O	TTL	PWM 6.



Pin Name	Pin Number	Pin Mux	Pin Type	Buffer Type <sup>a</sup>	Description
PWM7	23 31 36 40	PC6 (4) PA5 (4) PG7 (4) PG5 (8)	O	TTL	PWM 7.
PhA0	11 25 43 95	PD1 (3) PC4 (2) PF6 (4) PE2 (4)	I	TTL	QEI module 0 phase A.
PhA1	37 96	PG6 (1) PE3 (3)	I	TTL	QEI module 1 phase A.
PhB0	22 23 42 47 83 96	PC7 (2) PC6 (2) PF7 (4) PF0 (2) PH3 (1) PE3 (4)	I	TTL	QEI module 0 phase B.
PhB1	11 36 95	PD1 (11) PG7 (1) PE2 (3)	I	TTL	QEI module 1 phase B.
$\overline{\text{RST}}$	64	-	I	TTL	System reset input.
SSI0Clk	28	PA2 (1)	I/O	TTL	SSI module 0 clock.
SSI0Fss	29	PA3 (1)	I/O	TTL	SSI module 0 frame.
SSI0Rx	30	PA4 (1)	I	TTL	SSI module 0 receive.
SSI0Tx	31	PA5 (1)	O	TTL	SSI module 0 transmit.
SSI1Clk	60 74 76	PF2 (9) PE0 (2) PH4 (11)	I/O	TTL	SSI module 1 clock.
SSI1Fss	59 63 75	PF3 (9) PH5 (11) PE1 (2)	I/O	TTL	SSI module 1 frame.
SSI1Rx	58 62 95	PF4 (9) PH6 (11) PE2 (2)	I	TTL	SSI module 1 receive.
SSI1Tx	15 46 96	PH7 (11) PF5 (9) PE3 (2)	O	TTL	SSI module 1 transmit.
SWCLK	80	PC0 (3)	I	TTL	JTAG/SWD CLK.
SWDIO	79	PC1 (3)	I/O	TTL	JTAG TMS and SWDIO.
SWO	77	PC3 (3)	O	TTL	JTAG TDO and SWO.
TCK	80	PC0 (3)	I	TTL	JTAG/SWD CLK.
TDI	78	PC2 (3)	I	TTL	JTAG TDI.
TDO	77	PC3 (3)	O	TTL	JTAG TDO and SWO.
TMS	79	PC1 (3)	I	TTL	JTAG TMS and SWDIO.
U0Rx	26	PA0 (1)	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
U0Tx	27	PA1 (1)	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
U1CTS	2 10 34 50	PE6 (9) PD0 (9) PA6 (9) PJ3 (9)	I	TTL	UART module 1 Clear To Send modem status input signal.

Pin Name	Pin Number	Pin Mux	Pin Type	Buffer Type <sup>a</sup>	Description
U1DCD	1 11 35 52	PE7 (9) PD1 (9) PA7 (9) PJ4 (9)	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
U1DSR	47 53	PF0 (9) PJ5 (9)	I	TTL	UART module 1 Data Set Ready modem output control line.
U1DTR	40 55 100	PG5 (10) PJ7 (9) PD7 (9)	O	TTL	UART module 1 Data Terminal Ready modem status input signal.
U1RI	37 41 97	PG6 (10) PG4 (10) PD4 (9)	I	TTL	UART module 1 Ring Indicator modem status input signal.
U1RTS	43 54 61	PF6 (10) PJ6 (9) PF1 (9)	O	TTL	UART module 1 Request to Send modem output control line.
U1Rx	10 12 23 26 66 92	PD0 (5) PD2 (1) PC6 (5) PA0 (9) PB0 (5) PB4 (7)	I	TTL	UART module 1 receive.
U1Tx	11 13 22 27 67 91	PD1 (5) PD3 (1) PC7 (5) PA1 (9) PB1 (5) PB5 (7)	O	TTL	UART module 1 transmit.
U2Rx	10 19 92 98	PD0 (4) PG0 (1) PB4 (4) PD5 (9)	I	TTL	UART module 2 receive.
U2Tx	6 11 18 99	PE4 (5) PD1 (4) PG1 (1) PD6 (9)	O	TTL	UART module 2 transmit.
USB0DM	70	-	I/O	Analog	Bidirectional differential data pin (D- per USB specification).
USB0DP	71	-	I/O	Analog	Bidirectional differential data pin (D+ per USB specification).
USB0EPEN	19 24 34 72 83	PG0 (7) PC5 (6) PA6 (8) PB2 (8) PH3 (4)	O	TTL	Used in Host mode to control an external power source to supply power to the USB bus.
USB0ID	66	-	I	Analog	This signal senses the state of the USB ID signal. The USB PHY enables an integrated pull-up, and an external element (USB connector) indicates the initial state of the USB controller (pulled down is an A device and not pulled down is a B device).

Pin Name	Pin Number	Pin Mux	Pin Type	Buffer Type <sup>a</sup>	Description
USB0PFLT	22 23 35 65 74 76 87	PC7 (6) PC6 (7) PA7 (8) PB3 (8) PE0 (9) PH4 (4) PJ1 (9)	I	TTL	Used in Host mode by an external power source to indicate an error state by that power source.
USB0RBIAS	73	-	O	Analog	9.1-kΩ resistor (1% precision) used internally for USB analog circuitry.
USB0VBUS	67	-	I/O	Analog	This signal is used during the session negotiation protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing.
VDD	8 20 32 44 56 68 81 93	-	-	Power	Positive supply for I/O and some logic.
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.
VDDC	38 88	-	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VREFA	90	-	I	Analog	This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AIN <sub>n</sub> signal is converted to 1023. The VREFA input is limited to the range specified in Table 26-2 on page 941.

a. The TTL designation indicates the pin is TTL-compatible.

Table 24-4. Signals by Function, Except for GPIO

Function	Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
ADC	AIN0	1	I	Analog	ADC 0 input.
	AIN1	2	I	Analog	ADC 1 input.
	AIN2	5	I	Analog	ADC 2 input.
	AIN3	6	I	Analog	ADC 3 input.
	AIN4	100	I	Analog	ADC 4 input.
	AIN5	99	I	Analog	ADC 5 input.
	AIN6	98	I	Analog	ADC 6 input.
	AIN7	97	I	Analog	ADC 7 input.
	AIN8	96	I	Analog	ADC 8 input.
	AIN9	95	I	Analog	ADC 9 input.
	AIN10	92	I	Analog	ADC 10 input.
	AIN11	91	I	Analog	ADC 11 input.
	AIN12	13	I	Analog	ADC 12 input.
	AIN13	12	I	Analog	ADC 13 input.
	AIN14	11	I	Analog	ADC 14 input.
	AIN15	10	I	Analog	ADC 15 input.
	VREFA	90	I	Analog	This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AIN <sub>n</sub> signal is converted to 1023. The VREFA input is limited to the range specified in Table 26-2 on page 941.
Analog Comparators	C0+	90	I	Analog	Analog comparator 0 positive input.
	C0-	92	I	Analog	Analog comparator 0 negative input.
	C0o	24 58 90 91 100	O	TTL	Analog comparator 0 output.
	C1+	24	I	Analog	Analog comparator 1 positive input.
	C1-	91	I	Analog	Analog comparator 1 negative input.
	C1o	2 22 24 46 84	O	TTL	Analog comparator 1 output.
	C2+	23	I	Analog	Analog comparator 2 positive input.
	C2-	22	I	Analog	Analog comparator 2 negative input.
	C2o	1 23 43	O	TTL	Analog comparator 2 output.

Function	Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
Controller Area Network	CAN0Rx	10 30 34 92	I	TTL	CAN module 0 receive.
	CAN0Tx	11 31 35 91	O	TTL	CAN module 0 transmit.
	CAN1Rx	47	I	TTL	CAN module 1 receive.
	CAN1Tx	61	O	TTL	CAN module 1 transmit.
Ethernet PHY	LED0	59	O	TTL	MII LED 0.
	LED1	60	O	TTL	MII LED 1.

Function	Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
External Peripheral Interface	EPI0S0	83	I/O	TTL	EPI module 0 signal 0.
	EPI0S1	84	I/O	TTL	EPI module 0 signal 1.
	EPI0S2	25	I/O	TTL	EPI module 0 signal 2.
	EPI0S3	24	I/O	TTL	EPI module 0 signal 3.
	EPI0S4	23	I/O	TTL	EPI module 0 signal 4.
	EPI0S5	22	I/O	TTL	EPI module 0 signal 5.
	EPI0S6	86	I/O	TTL	EPI module 0 signal 6.
	EPI0S7	85	I/O	TTL	EPI module 0 signal 7.
	EPI0S8	74	I/O	TTL	EPI module 0 signal 8.
	EPI0S9	75	I/O	TTL	EPI module 0 signal 9.
	EPI0S10	76	I/O	TTL	EPI module 0 signal 10.
	EPI0S11	63	I/O	TTL	EPI module 0 signal 11.
	EPI0S12	42 58	I/O	TTL	EPI module 0 signal 12.
	EPI0S13	19	I/O	TTL	EPI module 0 signal 13.
	EPI0S14	18	I/O	TTL	EPI module 0 signal 14.
	EPI0S15	41 46	I/O	TTL	EPI module 0 signal 15.
	EPI0S16	14	I/O	TTL	EPI module 0 signal 16.
	EPI0S17	87	I/O	TTL	EPI module 0 signal 17.
	EPI0S18	39	I/O	TTL	EPI module 0 signal 18.
	EPI0S19	50 97	I/O	TTL	EPI module 0 signal 19.
	EPI0S20	12	I/O	TTL	EPI module 0 signal 20.
	EPI0S21	13	I/O	TTL	EPI module 0 signal 21.
	EPI0S22	91	I/O	TTL	EPI module 0 signal 22.
	EPI0S23	92	I/O	TTL	EPI module 0 signal 23.
	EPI0S24	95	I/O	TTL	EPI module 0 signal 24.
	EPI0S25	96	I/O	TTL	EPI module 0 signal 25.
	EPI0S26	62	I/O	TTL	EPI module 0 signal 26.
	EPI0S27	15	I/O	TTL	EPI module 0 signal 27.
	EPI0S28	52 98	I/O	TTL	EPI module 0 signal 28.
	EPI0S29	53 99	I/O	TTL	EPI module 0 signal 29.
	EPI0S30	54 100	I/O	TTL	EPI module 0 signal 30.
EPI0S31	36	I/O	TTL	EPI module 0 signal 31.	

Function	Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
General-Purpose Timers	CCP0	13 22 23 39 55 58 66 72 91 97	I/O	TTL	Capture/Compare/PWM 0.
	CCP1	24 25 34 43 54 67 90 96 100	I/O	TTL	Capture/Compare/PWM 1.
	CCP2	6 11 25 46 53 67 75 91 95 98	I/O	TTL	Capture/Compare/PWM 2.
	CCP3	6 23 24 35 41 61 72 74 97	I/O	TTL	Capture/Compare/PWM 3.
	CCP4	22 25 35 42 52 95 98	I/O	TTL	Capture/Compare/PWM 4.
	CCP5	5 12 25 36 40 90 91	I/O	TTL	Capture/Compare/PWM 5.
	CCP6	10 12 50 75 86 91	I/O	TTL	Capture/Compare/PWM 6.

Function	Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
	CCP7	11 13 85 90 96	I/O	TTL	Capture/Compare/PWM 7.
I2C	I2C0SCL	72	I/O	OD	I <sup>2</sup> C module 0 clock.
	I2C0SDA	65	I/O	OD	I <sup>2</sup> C module 0 data.
	I2C1SCL	14 19 26 34	I/O	OD	I <sup>2</sup> C module 1 clock.
	I2C1SDA	18 27 35 87	I/O	OD	I <sup>2</sup> C module 1 data.
I2S	I2S0RXMCLK	16 29 98	I/O	TTL	I <sup>2</sup> S module 0 receive master clock.
	I2S0RXSCK	10 40	I/O	TTL	I <sup>2</sup> S module 0 receive clock.
	I2S0RXSD	17 28 97	I/O	TTL	I <sup>2</sup> S module 0 receive data.
	I2S0RXWS	11 37	I/O	TTL	I <sup>2</sup> S module 0 receive word select.
	I2S0TXMCLK	43 61	I/O	TTL	I <sup>2</sup> S module 0 transmit master clock.
	I2S0TXSCK	30 90 99	I/O	TTL	I <sup>2</sup> S module 0 transmit clock.
	I2S0TXSD	5 47	I/O	TTL	I <sup>2</sup> S module 0 transmit data.
	I2S0TXWS	6 31 100	I/O	TTL	I <sup>2</sup> S module 0 transmit word select.
JTAG/SWD/SWO	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	TTL	JTAG TMS and SWDIO.



Function	Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
PWM	Fault0	6 16 17 39 58 65 75 83 99	I	TTL	PWM Fault 0.
	Fault1	37 40 41 42 90	I	TTL	PWM Fault 1.
	Fault2	16 24 63	I	TTL	PWM Fault 2.
	Fault3	65 84	I	TTL	PWM Fault 3.
	PWM0	10 14 17 19 34 47	O	TTL	PWM 0.
	PWM1	11 16 18 35 61 87	O	TTL	PWM 1.
	PWM2	12 60 66 86	O	TTL	PWM 2.
	PWM3	13 59 67 85	O	TTL	PWM 3.
	PWM4	2 19 28 34 60 62 74 86	O	TTL	PWM 4.
	PWM5	1 15 18 29 35 59 75 85	O	TTL	PWM 5.
PWM6		O	TTL	PWM 6.	

Function	Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
		25 30 37 41			
	PWM7	23 31 36 40	O	TTL	PWM 7.
Power	GND	9 21 33 45 57 69 82 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.
	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 VDDC pins at the board level in addition to the decoupling capacitor(s).
	VDD	8 20 32 44 56 68 81 93	-	Power	Positive supply for I/O and some logic.
	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.
	VDDC	38 88	-	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 <sup>a</sup>	Description
QEI	IDX0	10 40 72 90 92 100	I	TTL	QEI module 0 index.
	IDX1	17 61 84	I	TTL	QEI module 1 index.
	PhA0	11 25 43 95	I	TTL	QEI module 0 phase A.
	PhA1	37 96	I	TTL	QEI module 1 phase A.
	PhB0	22 23 42 47 83 96	I	TTL	QEI module 0 phase B.
	PhB1	11 36 95	I	TTL	QEI module 1 phase B.
	SSI	SSI0Clk	28	I/O	TTL
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.
SSI1Clk		60 74 76	I/O	TTL	SSI module 1 clock.
SSI1Fss		59 63 75	I/O	TTL	SSI module 1 frame.
SSI1Rx		58 62 95	I	TTL	SSI module 1 receive.
SSI1Tx		15 46 96	O	TTL	SSI module 1 transmit.
System Control & Clocks	NMI	89	I	TTL	Non-maskable interrupt.
	OSC0	48	I	Analog	Main oscillator crystal input or an external clock reference input.
	OSC1	49	O	Analog	Main oscillator crystal output.
	RST	64	I	TTL	System reset input.

Function	Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
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.
	U1CTS	2 10 34 50	I	TTL	UART module 1 Clear To Send modem status input signal.
	U1DCD	1 11 35 52	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
	U1DSR	47 53	I	TTL	UART module 1 Data Set Ready modem output control line.
	U1DTR	40 55 100	O	TTL	UART module 1 Data Terminal Ready modem status input signal.
	U1RI	37 41 97	I	TTL	UART module 1 Ring Indicator modem status input signal.
	U1RTS	43 54 61	O	TTL	UART module 1 Request to Send modem output control line.
	U1Rx	10 12 23 26 66 92	I	TTL	UART module 1 receive.
	U1Tx	11 13 22 27 67 91	O	TTL	UART module 1 transmit.
	U2Rx	10 19 92 98	I	TTL	UART module 2 receive.
	U2Tx	6 11 18 99	O	TTL	UART module 2 transmit.

Function	Pin Name	Pin Number	Pin Type	Buffer Type <sup>a</sup>	Description
USB	USB0DM	70	I/O	Analog	Bidirectional differential data pin (D- per USB specification).
	USB0DP	71	I/O	Analog	Bidirectional differential data pin (D+ per USB specification).
	USB0EPEN	19 24 34 72 83	O	TTL	Used in Host mode to control an external power source to supply power to the USB bus.
	USB0ID	66	I	Analog	This signal senses the state of the USB ID signal. The USB PHY enables an integrated pull-up, and an external element (USB connector) indicates the initial state of the USB controller (pulled down is an A device and not pulled down is a B device).
	USB0PFLT	22 23 35 65 74 76 87	I	TTL	Used in Host mode by an external power source to indicate an error state by that power source.
	USB0RBIAS	73	O	Analog	9.1-kΩ resistor (1% precision) used internally for USB analog circuitry.
	USB0VBUS	67	I/O	Analog	This signal is used during the session negotiation protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing.

a. The TTL designation indicates the pin is TTL-compatible.

**Table 24-5. GPIO Pins and Alternate Functions**

GPIO	Pin	enc=1	enc=2	enc=3	enc=4	enc=5	enc=6	enc=7	enc=8	enc=9	enc=10	enc=11
PA0	26	U0Rx	-	-	-	-	-	-	I2C1SCL	U1Rx	-	-
PA1	27	U0Tx	-	-	-	-	-	-	I2C1SDA	U1Tx	-	-
PA2	28	SSI0Clk	-	-	PWM4	-	-	-	-	I2S0RXSD	-	-
PA3	29	SSI0Fss	-	-	PWM5	-	-	-	-	I2S0RXCLK	-	-
PA4	30	SSI0Rx	-	-	PWM6	CAN0Rx	-	-	-	I2S0TXSCK	-	-
PA5	31	SSI0Tx	-	-	PWM7	CAN0Tx	-	-	-	I2S0TXWS	-	-
PA6	34	I2C1SCL	CCP1	-	PWM0	PWM4	CAN0Rx	-	USB0EPEN	U1CTS	-	-
PA7	35	I2C1SDA	CCP4	-	PWM1	PWM5	CAN0Tx	CCP3	USB0PFLT	U1DCD	-	-
PB0	66	CCP0	PWM2	-	-	U1Rx	-	-	-	-	-	-
PB1	67	CCP2	PWM3	-	CCP1	U1Tx	-	-	-	-	-	-
PB2	72	I2C0SCL	IDX0	-	CCP3	CCP0	-	-	USB0EPEN	-	-	-
PB3	65	I2C0SDA	Fault0	-	Fault3	-	-	-	USB0PFLT	-	-	-
PB4	92	-	-	-	U2Rx	CAN0Rx	IDX0	U1Rx	EPI0S23	-	-	-
PB5	91	C0o	CCP5	CCP6	CCP0	CAN0Tx	CCP2	U1Tx	EPI0S22	-	-	-
PB6	90	CCP1	CCP7	C0o	Fault1	IDX0	CCP5	-	-	I2S0TXSCK	-	-
PB7	89	-	-	-	NMI	-	-	-	-	-	-	-
PC0	80	-	-	TCK SWCLK	-	-	-	-	-	-	-	-

GPIO	Pin	enc=1	enc=2	enc=3	enc=4	enc=5	enc=6	enc=7	enc=8	enc=9	enc=10	enc=11
PC1	79	-	-	TMS SWDIO	-	-	-	-	-	-	-	-
PC2	78	-	-	TDI	-	-	-	-	-	-	-	-
PC3	77	-	-	TDO SWO	-	-	-	-	-	-	-	-
PC4	25	CCP5	PhA0	-	PWM6	CCP2	CCP4	-	EPI0S2	CCP1	-	-
PC5	24	CCP1	C1o	C0o	Fault2	CCP3	USB0EPEN	-	EPI0S3	-	-	-
PC6	23	CCP3	PhB0	C2o	PWM7	U1Rx	CCP0	USB0PFLT	EPI0S4	-	-	-
PC7	22	CCP4	PhB0	-	CCP0	U1Tx	USB0PFLT	C1o	EPI0S5	-	-	-
PD0	10	PWM0	CAN0Rx	IDX0	U2Rx	U1Rx	CCP6	-	I2S0RXSCK	U1CTS	-	-
PD1	11	PWM1	CAN0Tx	PhA0	U2Tx	U1Tx	CCP7	-	I2S0RXWS	U1DCD	CCP2	PhB1
PD2	12	U1Rx	CCP6	PWM2	CCP5	-	-	-	EPI0S20	-	-	-
PD3	13	U1Tx	CCP7	PWM3	CCP0	-	-	-	EPI0S21	-	-	-
PD4	97	CCP0	CCP3	-	-	-	-	-	I2S0RXSD	U1RI	EPI0S19	-
PD5	98	CCP2	CCP4	-	-	-	-	-	I2S0RXCLK	U2Rx	EPI0S28	-
PD6	99	Fault0	-	-	-	-	-	-	I2S0TXSCK	U2Tx	EPI0S29	-
PD7	100	IDX0	C0o	CCP1	-	-	-	-	I2S0TXWS	U1DTR	EPI0S30	-
PE0	74	PWM4	SSI1Clk	CCP3	-	-	-	-	EPI0S8	USB0PFLT	-	-
PE1	75	PWM5	SSI1Fss	Fault0	CCP2	CCP6	-	-	EPI0S9	-	-	-
PE2	95	CCP4	SSI1Rx	PhB1	PhA0	CCP2	-	-	EPI0S24	-	-	-
PE3	96	CCP1	SSI1Tx	PhA1	PhB0	CCP7	-	-	EPI0S25	-	-	-
PE4	6	CCP3	-	-	Fault0	U2Tx	CCP2	-	-	I2S0TXWS	-	-
PE5	5	CCP5	-	-	-	-	-	-	-	I2S0TXSD	-	-
PE6	2	PWM4	C1o	-	-	-	-	-	-	U1CTS	-	-
PE7	1	PWM5	C2o	-	-	-	-	-	-	U1DCD	-	-
PF0	47	CAN1Rx	PhB0	PWM0	-	-	-	-	I2S0TXSD	U1DSR	-	-
PF1	61	CAN1Tx	IDX1	PWM1	-	-	-	-	I2S0TXCLK	U1RTS	CCP3	-
PF2	60	LED1	PWM4	-	PWM2	-	-	-	-	SSI1Clk	-	-
PF3	59	LED0	PWM5	-	PWM3	-	-	-	-	SSI1Fss	-	-
PF4	58	CCP0	C0o	-	Fault0	-	-	-	EPI0S12	SSI1Rx	-	-
PF5	46	CCP2	C1o	-	-	-	-	-	EPI0S15	SSI1Tx	-	-
PF6	43	CCP1	C2o	-	PhA0	-	-	-	-	I2S0TXCLK	U1RTS	-
PF7	42	CCP4	-	-	PhB0	-	-	-	EPI0S12	Fault1	-	-
PG0	19	U2Rx	PWM0	I2C1SCL	PWM4	-	-	USB0EPEN	EPI0S13	-	-	-
PG1	18	U2Tx	PWM1	I2C1SDA	PWM5	-	-	-	EPI0S14	-	-	-
PG2	17	PWM0	-	-	Fault0	-	-	-	IDX1	I2S0RXSD	-	-
PG3	16	PWM1	-	-	Fault2	-	-	-	Fault0	I2S0RXCLK	-	-
PG4	41	CCP3	-	-	Fault1	-	-	-	EPI0S15	PWM6	U1RI	-
PG5	40	CCP5	-	-	IDX0	Fault1	-	-	PWM7	I2S0RXSCK	U1DTR	-
PG6	37	PhA1	-	-	PWM6	-	-	-	Fault1	I2S0RXWS	U1RI	-
PG7	36	PhB1	-	-	PWM7	-	-	-	CCP5	EPI0S31	-	-
PH0	86	CCP6	PWM2	-	-	-	-	-	EPI0S6	PWM4	-	-
PH1	85	CCP7	PWM3	-	-	-	-	-	EPI0S7	PWM5	-	-
PH2	84	IDX1	C1o	-	Fault3	-	-	-	EPI0S1	-	-	-

GPIO	Pin	enc=1	enc=2	enc=3	enc=4	enc=5	enc=6	enc=7	enc=8	enc=9	enc=10	enc=11
PH3	83	PhB0	Fault0	-	USB0EPEN	-	-	-	EPIOS0	-	-	-
PH4	76	-	-	-	USB0PFLT	-	-	-	EPIOS10	-	-	SSI1Clk
PH5	63	-	-	-	-	-	-	-	EPIOS11	-	Fault2	SSI1Fss
PH6	62	-	-	-	-	-	-	-	EPIOS26	-	PWM4	SSI1Rx
PH7	15	-	-	-	-	-	-	-	EPIOS27	-	PWM5	SSI1Tx

## 25 Operating Characteristics

**Table 25-1. Temperature Characteristics**

Characteristic <sup>a</sup>	Symbol	Value	Unit
Industrial operating temperature range	$T_A$	-40 to +85	°C

a. Maximum storage temperature is 150°C.

**Table 25-2. Thermal Characteristics**

Characteristic	Symbol	Value	Unit
Thermal resistance (junction to ambient) <sup>a</sup>	$\Theta_{JA}$	34	°C/W
Average junction temperature <sup>b</sup>	$T_J$	$T_A + (P_{AVG} \cdot \Theta_{JA})$	°C

a. Junction to ambient thermal resistance  $\Theta_{JA}$  numbers are determined by a package simulator.

b. Power dissipation is a function of temperature.

**Table 25-3. ESD Absolute Maximum Ratings**

Parameter Name	Min	Nom	Max	Unit
$V_{ESDHBM}$				V
$V_{ESDCDM}$				V



## 26 Electrical Characteristics

### 26.1 DC Characteristics

#### 26.1.1 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 26-1. Maximum Ratings**

Parameter	Parameter Name <sup>a</sup>	Value		Unit
		Min	Max	
V <sub>DD</sub>	I/O supply voltage (V <sub>DD</sub> )	0	4	V
V <sub>DDA</sub>	Analog supply voltage (V <sub>DDA</sub> )	0	4	V
V <sub>IN</sub>	Input voltage	-0.3	5.5	V
I	Maximum current per output pins	-	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<sub>DD</sub>).

#### 26.1.2 Recommended DC Operating Conditions

For special high-current applications, the GPIO output buffers may be used with the following restrictions. With the GPIO pins configured as 8-mA output drivers, a total of four GPIO outputs may be used to sink current loads up to 18 mA each. At 18-mA sink current loading, the V<sub>OL</sub> value is specified as 1.2 V. The high-current GPIO package pins must be selected such that there are only a maximum of two per side of the physical package with the total number of high-current GPIO outputs not exceeding four for the entire package.

**Table 26-2. Recommended DC Operating Conditions**

Parameter	Parameter Name	Min	Nom	Max	Unit
V <sub>DD</sub>	I/O supply voltage	3.0	3.3	3.6	V
V <sub>DDA</sub>	Analog supply voltage	3.0	3.3	3.6	V
V <sub>DDC</sub> <sup>a</sup>	Core supply voltage	1.08	1.2	1.32	V
V <sub>REFA</sub>	External voltage reference for ADC	pending <sup>b</sup>	3.0 <sup>cd</sup>	pending <sub>b</sub>	V
V <sub>IH</sub>	High-level input voltage	2.0	-	5.0	V
V <sub>IL</sub>	Low-level input voltage	-0.3	-	1.3	V
V <sub>SIH</sub>	High-level input voltage for Schmitt trigger inputs	0.8 * V <sub>DD</sub>	-	V <sub>DD</sub>	V
V <sub>SIL</sub>	Low-level input voltage for Schmitt trigger inputs	0	-	0.2 * V <sub>DD</sub>	V
V <sub>OH</sub> <sup>e</sup>	High-level output voltage	2.4	-	-	V
V <sub>OL</sub> <sup>a</sup>	Low-level output voltage	-	-	0.4	V

Parameter	Parameter Name	Min	Nom	Max	Unit
I <sub>OH</sub>	High-level source current, V <sub>OH</sub> =2.4 V				
	2-mA Drive	2.0	-	-	mA
	4-mA Drive	4.0	-	-	mA
	8-mA Drive	8.0	-	-	mA
I <sub>OL</sub>	Low-level sink current, V <sub>OL</sub> =0.4 V				
	2-mA Drive	2.0	-	-	mA
	4-mA Drive	4.0	-	-	mA
	8-mA Drive	8.0	-	-	mA

- a. V<sub>DDC</sub> is supplied from the output of the LDO.
- b. Pending characterization completion.
- c. Ground is always used as the reference level for the minimum conversion value.
- d. Care must be taken to supply a reference voltage of acceptable quality.
- e. V<sub>OL</sub> and V<sub>OH</sub> shift to 1.2 V when using high-current GPIOs.

### 26.1.3 On-Chip Low Drop-Out (LDO) Regulator Characteristics

Table 26-3. LDO Regulator Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
C <sub>LDO</sub>	External filter capacitor size for internal power supply	1.0	-	3.0	μF
V <sub>LDO</sub>	LDO output voltage	1.08	1.2	1.32	V

### 26.1.4 Flash Memory Characteristics

Table 26-4. Flash Memory Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
PE <sub>CYC</sub>	Number of guaranteed mass program/erase cycles before failure <sup>ab</sup>	15,000	pending <sup>c</sup>	-	cycles
T <sub>RET</sub>	Data retention at average operating temperature of 125°C	20	-	-	years
T <sub>PROG</sub>	Word program time	-	-	1	ms
T <sub>BPROG</sub>	Buffer program time	-	-	1	ms
T <sub>ERASE</sub>	Page erase time	-	-	16	ms
T <sub>ME</sub>	Mass erase time	-	-	16	ms

- a. A program/erase cycle is defined as switching the bits from 1-> 0 -> 1.
- b. Caution should be used when performing block erases, as repeated block erases can shorten the number of guaranteed erase cycles.
- c. Pending characterization completion.

### 26.1.5 GPIO Module Characteristics

Table 26-5. GPIO Module DC Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
R <sub>GPIOU</sub>	GPIO internal pull-up resistor	50	-	110	kΩ
R <sub>GPIOPD</sub>	GPIO internal pull-down resistor	55	-	180	kΩ

## 26.1.6 USB Module Characteristics

The Stellaris<sup>®</sup> USB controller DC electrical specifications are compliant with the “Universal Serial Bus Specification Rev. 2.0” (full-speed and low-speed support) and the “On-The-Go Supplement to the USB 2.0 Specification Rev. 1.0”. Some components of the USB system are integrated within the LM3S5791 microcontroller and specific to the Stellaris<sup>®</sup> microcontroller design. An external component resistor is needed as specified in Table 26-6.

**Table 26-6. USB Controller DC Characteristics**

Parameter	Parameter Name	Value	Unit
R <sub>UBIAS</sub>	Value of the pull-down resistor on the USB0RBIAS pin	9.1K ± 1 %	Ω

## 26.1.7 Current Specifications

This section provides information on typical and maximum power consumption under various conditions.

### 26.1.7.1 Nominal and Maximum Current Specifications

The current measurements specified in the table that follows are run on the core processor using SRAM with the following specifications (except as noted):

- $V_{DD} = 3.3 \text{ V}$
- $V_{DDC} = 1.2 \text{ V}$
- $V_{DDA} = 3.3 \text{ V}$
- Temperature = 25°C
- Clock Source (MOSC) = 3.579545 MHz Crystal Oscillator
- Main oscillator (MOSC) = enabled
- Precision Internal oscillator (PIOSC) = disabled

---

**Important:** The next 2 tables should be filled in for all of the following situations:

- MOSC (from a crystal) 16 MHz, 12 MHz, 8 MHz, and 3.579545 MHz
  - PIOSC SYSDIV=1, 2, 4
  - MOSC (from a crystal) with PLL 100 MHz, 80 MHz, SYSDIV = 8, 64
  - PIOSC with PLL 100 MHz, 80 MHz, SYSDIV = 8, 64
  - 4.19 MHz HIB clock with PLL 100 MHz 80 MHz, SYSDIV = 8, 64
-

Table 26-7. Detailed Current Specifications

Parameter	Parameter Name	Conditions	3.3 V $V_{DD}$ , $V_{DDA}$		Unit
			Nom	Max	
$I_{DD\_RUN}$	Run mode 1 (Flash loop)	$V_{DD} = 3.3\text{ V}$ Code= while(1){} executed in Flash Peripherals = All ON System Clock = 50 MHz (with PLL)	pending <sup>a</sup>	pending <sup>a</sup>	mA
	Run mode 2 (Flash loop)	$V_{DD} = 3.3\text{ V}$ Code= while(1){} executed in Flash Peripherals = All OFF System Clock = 50 MHz (with PLL)	pending <sup>a</sup>	pending <sup>a</sup>	
	Run mode 1 (SRAM loop)	$V_{DD} = 3.3\text{ V}$ Code= while(1){} executed in SRAM Peripherals = All ON System Clock = 50 MHz (with PLL)	pending <sup>a</sup>	pending <sup>a</sup>	
	Run mode 2 (SRAM loop)	$V_{DD} = 3.3\text{ V}$ Code= while(1){} executed in SRAM Peripherals = All OFF System Clock = 50 MHz (with PLL)	pending <sup>a</sup>	pending <sup>a</sup>	
$I_{DD\_SLEEP}$	Sleep mode	$V_{DD} = 3.3\text{ V}$ Peripherals = All OFF System Clock = 50 MHz (with PLL)	pending <sup>a</sup>	pending <sup>a</sup>	mA
$I_{DD\_DEEPSLEEP}$	Deep-Sleep mode	LDO = 2.25 V Peripherals = All OFF System Clock = IOS30KHZ/64	pending <sup>a</sup>	pending <sup>a</sup>	mA

a. Pending characterization completion.

### 26.1.7.2 Typical Current Consumption vs. Frequency

Figure 26-1 on page 945 shows how typical current when running out of Flash memory varies with frequency. Data is provided across frequency for all peripherals on and all peripherals off. The microcontroller is clocked by MOSC using the PLL.

**Figure 26-1. Typical Current Across Frequency**

**Pending**

### **26.1.7.3 Typical Current Consumption vs. Temperature**

Figure 26-2 on page 946 shows how typical current varies across temperature when running out of Flash memory varies with frequency. Data is provided for all peripherals off. The microcontroller is clocked by MOSC using the PLL.

Figure 26-2. Typical Current Across Temperature

**Pending**

#### 26.1.7.4 Typical Peripheral Current Consumption

The current consumption of the on-chip peripherals is given in . Data is provided for the following conditions:

- I/O pins are in input mode with a static value at  $V_{DD}$  or ground and no load.
- All peripherals are not clocked except for the peripheral listed.
- Specified temperature and voltage

Table 26-8. Typical Peripheral Current Consumption

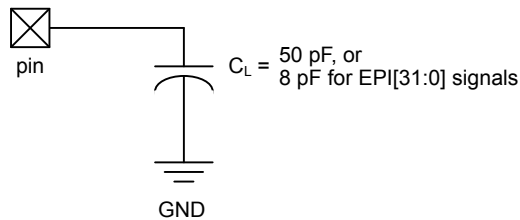
Peripheral	Current	Unit

## 26.2 AC Characteristics

### 26.2.1 Load Conditions

Unless otherwise specified, the following conditions are true for all timing measurements.

Figure 26-3. Load Conditions



## 26.2.2 Clocks

The following sections provide specifications on the various clock sources and mode.

### 26.2.2.1 PLL Specifications

The following tables provide specifications for using the PLL.

**Table 26-9. Phase Locked Loop (PLL) Characteristics**

Parameter	Parameter Name	Min	Nom	Max	Unit
$f_{\text{REF\_XTAL}}$	Crystal reference <sup>a</sup>	3.579545	-	16.384	MHz
$f_{\text{REF\_EXT}}$	External clock reference <sup>a</sup>	3.579545	-	16.384	MHz
$f_{\text{PLL}}$	PLL frequency <sup>b</sup>	-	400	-	MHz
$T_{\text{READY}}$	PLL lock time	0.562 <sup>c</sup>	-	1.38 <sup>d</sup>	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.

c. Using a 16.384-MHz crystal

d. Using 3.5795-MHz crystal

Table 26-10 on page 947 shows the actual frequency of the PLL based on the crystal frequency used (defined by the XTAL field in the **RCC** register).

**Table 26-10. Actual PLL Frequency**

XTAL	Crystal Frequency (MHz)	PLL Frequency (MHz)	Error
0x04	3.5795	400.904	0.0023%
0x05	3.6864	398.1312	0.0047%
0x06	4.0	400	-
0x07	4.096	401.408	0.0035%
0x08	4.9152	398.1312	0.0047%
0x09	5.0	400	-
0x0A	5.12	399.36	0.0016%
0x0B	6.0	400	-
0x0C	6.144	399.36	0.0016%
0x0D	7.3728	398.1312	0.0047%
0x0E	8.0	400	0.0047%
0x0F	8.192	398.6773333	0.0033%
0x10	10.0	400	-
0x11	12.0	400	-

XTAL	Crystal Frequency (MHz)	PLL Frequency (MHz)	Error
0x12	12.288	401.408	0.0035%
0x13	13.56	397.76	0.0056%
0x14	14.318	400.90904	0.0023%
0x15	16.0	400	-
0x16	16.384	404.1386667	0.010%

### 26.2.2.2 PIOSC Specifications

Table 26-11. PIOSC Clock Characteristics

Parameter	Parameter Name	Min	Max	Unit
f <sub>PIOSC</sub>	Internal 16-MHz precision oscillator frequency variance, factory calibrated at 25 °C <sup>a</sup>	-	±1%	-

a. Variance is ±3% across temperature.

### 26.2.2.3 Internal 30-kHz Oscillator Specifications

Table 26-12. 30-kHz Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
f <sub>IOSC30KHZ</sub>	Internal 30-KHz oscillator frequency	15	30	45	KHz

### 26.2.2.4 Main Oscillator Specifications

Table 26-13. Main Oscillator Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
f <sub>MOSC</sub>	Main oscillator frequency	1	-	16.384	MHz
t <sub>MOSC_PER</sub>	Main oscillator period	61	-	1000	ns
t <sub>MOSC_SETTLE</sub>	Main oscillator settling time	17.5	-	20	ms
f <sub>REF_XTAL_BYPASS</sub>	Crystal reference using the main oscillator (PLL in BYPASS mode) <sup>a</sup>	1	-	16.384	MHz
f <sub>REF_EXT_BYPASS</sub>	External clock reference (PLL in BYPASS mode) <sup>a</sup>	0	-	80	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 26-14. MOSC Oscillator Input Characteristics

Name	Value						Condition
Frequency	16	12	8	6	4	3.5	MHz
Frequency tolerance	±100	±100	±100	±100	±100	±100	PPM
Oscillation mode	parallel	parallel	parallel	parallel	parallel	parallel	-
Equivalent series resistance (max)	70	90	120	160	200	220	Ω
Load capacitance	16	16	16	16	16	16	pF
Drive level (typ)	100	100	100	100	100	100	μw



## 26.2.3 JTAG and Boundary Scan

Table 26-15. 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	pending <sup>a</sup>	-	pending <sup>a</sup>	ns	
J6	$t_{TCK\_F}$	TCK fall time	pending <sup>a</sup>	-	pending <sup>a</sup>	ns	
J7	$t_{TMS\_SU}$	TMS setup time to TCK rise	20	-	-	ns	
J8	$t_{TMS\_HLD}$	TMS hold time from TCK rise	20	-	-	ns	
J9	$t_{TDI\_SU}$	TDI setup time to TCK rise	25	-	-	ns	
J10	$t_{TDI\_HLD}$	TDI hold time from TCK rise	25	-	-	ns	
J11 $t_{TDO\_ZDV}$	TCK fall to Data Valid from High-Z	2-mA drive	-	23	35	ns	
		4-mA drive			15	26	ns
		8-mA drive			14	25	ns
		8-mA drive with slew rate control			18	29	ns
J12 $t_{TDO\_DV}$	TCK fall to Data Valid from Data Valid	2-mA drive	-	21	35	ns	
		4-mA drive			14	25	ns
		8-mA drive			13	24	ns
		8-mA drive with slew rate control			18	28	ns
J13 $t_{TDO\_DVZ}$	TCK fall to High-Z from Data Valid	2-mA drive	-	9	11	ns	
		4-mA drive			7	9	ns
		8-mA drive			6	8	ns
		8-mA drive with slew rate control			7	9	ns

a. Pending characterization completion.

Figure 26-4. JTAG Test Clock Input Timing

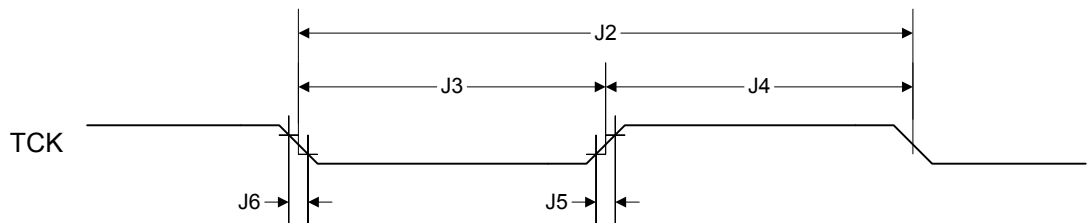
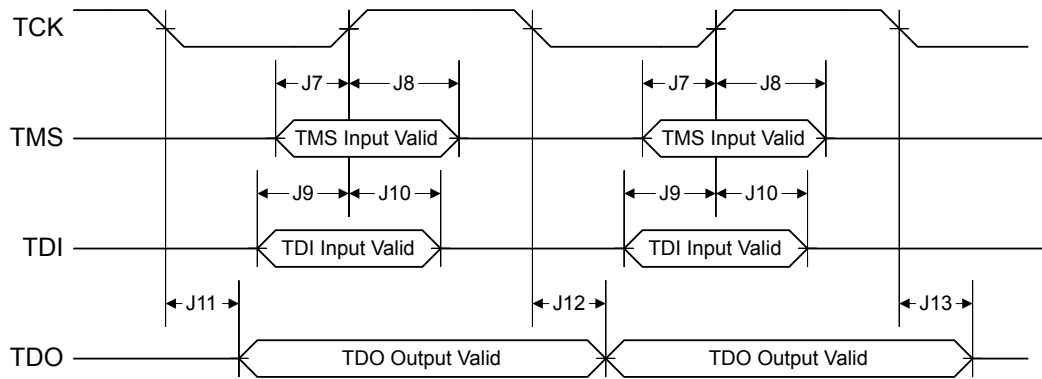


Figure 26-5. JTAG Test Access Port (TAP) Timing



### 26.2.4 Reset

Table 26-16. 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	-	-	95	system clocks
R6	$T_{IRBOR}$	Internal reset timeout after BOR	-	-	7	system clocks
R7	$T_{IRHWR}$	Internal reset timeout after hardware reset ( $\overline{RST}$ pin)	-	-	7	system clocks
R8	$T_{IRSWR}$	Internal reset timeout after software-initiated system reset	-	-	16	system clocks
R9	$T_{IRWDR}$	Internal reset timeout after watchdog reset	-	-	16	system clocks
R10	$T_{IRMFR}$	Internal reset timeout after MOSC failure reset	-	-	32	system clocks
R11	$T_{VDDRISE}$	Supply voltage ( $V_{DD}$ ) rise time (0V-3.3V)	-	-	100	ms
R12	$T_{MIN}$	Minimum $\overline{RST}$ pulse width	2	-	-	$\mu$ s

Figure 26-6. External Reset Timing ( $\overline{RST}$ )

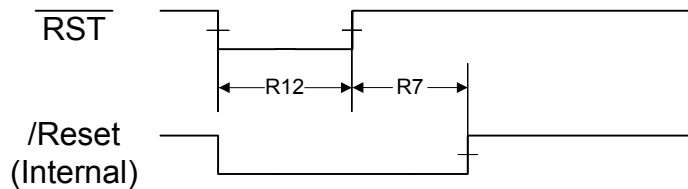


Figure 26-7. Power-On Reset Timing

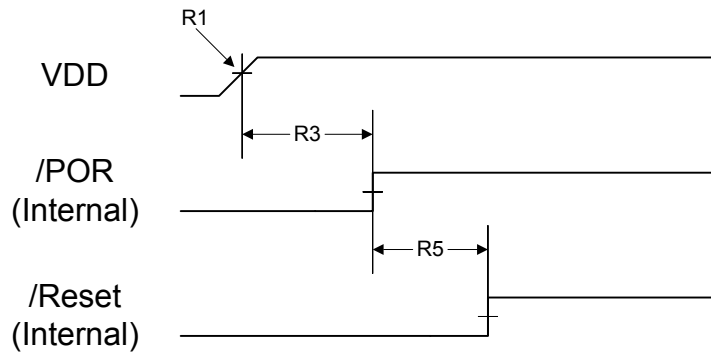


Figure 26-8. Brown-Out Reset Timing

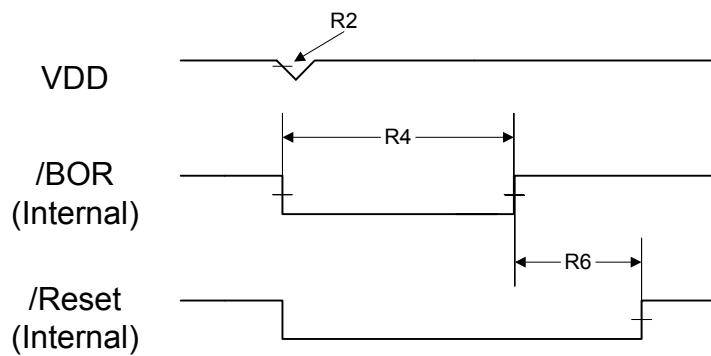


Figure 26-9. Software Reset Timing

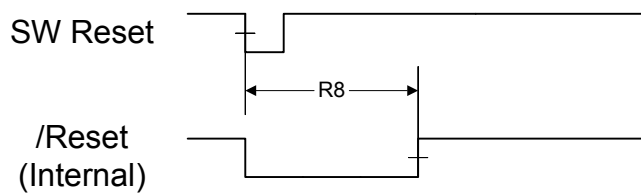


Figure 26-10. Watchdog Reset Timing

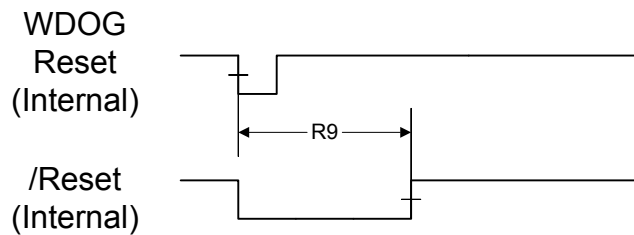
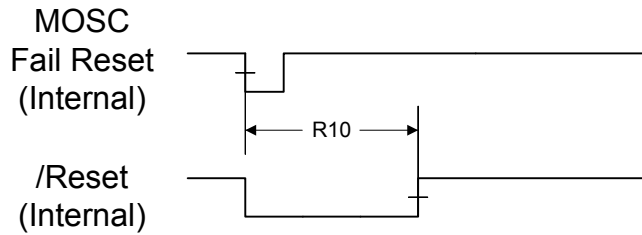


Figure 26-11. MOSC Failure Reset Timing



### 26.2.5 General-Purpose I/O (GPIO)

**Note:** All GPIOs are 5-V tolerant.

Table 26-17. GPIO Characteristics

Parameter	Parameter Name	Condition	Min	Nom	Max	Unit
$t_{GPIO\text{R}}$	GPIO Rise Time (from 20% to 80% of $V_{DD}$ )	2-mA drive	-	14	20	ns
		4-mA drive		7	10	ns
		8-mA drive		4	5	ns
		8-mA drive with slew rate control		6	8	ns
$t_{GPIO\text{F}}$	GPIO Fall Time (from 80% to 20% of $V_{DD}$ )	2-mA drive	-	14	21	ns
		4-mA drive		7	11	ns
		8-mA drive		4	6	ns
		8-mA drive with slew rate control		6	8	ns

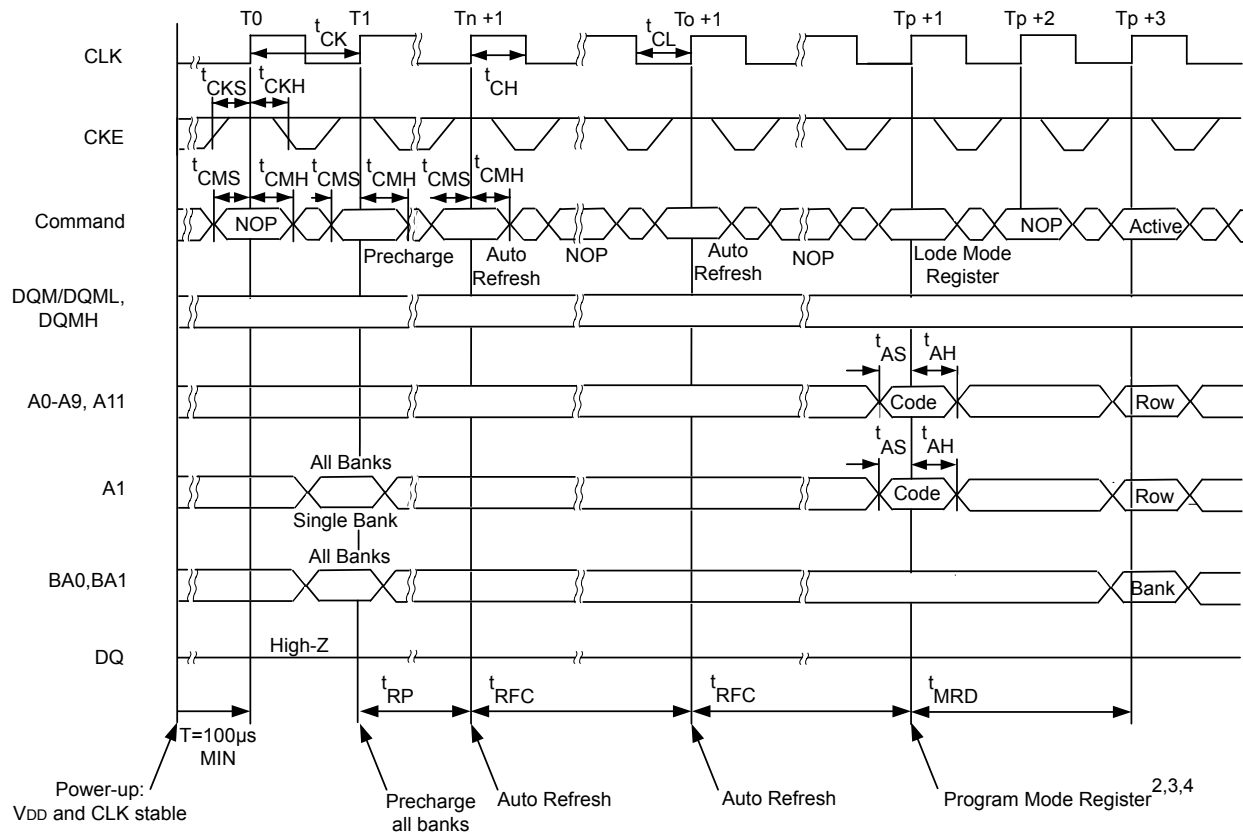
### 26.2.6 External Peripheral Interface (EPI)

Table 26-18. EPI Characteristics<sup>a</sup>

Parameter	Parameter Name	Condition	Min	Nom	Max	Unit
$t_{EPI\text{R}}$	EPI Rise Time (from 20% to 80% of $V_{DD}$ )	2-mA drive	-	3.3	4.4	ns
		4-mA drive		1.6	2.3	ns
		8-mA drive		1.1	1.5	ns
		8-mA drive with slew rate control		2.6	3.0	ns
$t_{EPI\text{F}}$	EPI Fall Time (from 80% to 20% of $V_{DD}$ )	2-mA drive	-	3.1	4.8	ns
		4-mA drive		1.8	2.7	ns
		8-mA drive		1.5	2.3	ns
		8-mA drive with slew rate control		2.3	3.4	ns

a. Load conditions when using EPI:  $C_L$  is 8 pF.

Figure 26-12. SDRAM Initialization and Load Mode Register Timing



Notes:

1. If CS is high at clock high time, all applied commands are NOP.
2. The **Mode** register may be loaded prior to the auto refresh cycles if desired.
3. JEDEC and PC100 specify three clocks.
4. Outputs are guaranteed High-Z after command is issued.



Figure 26-13. SDRAM Read Command Timing

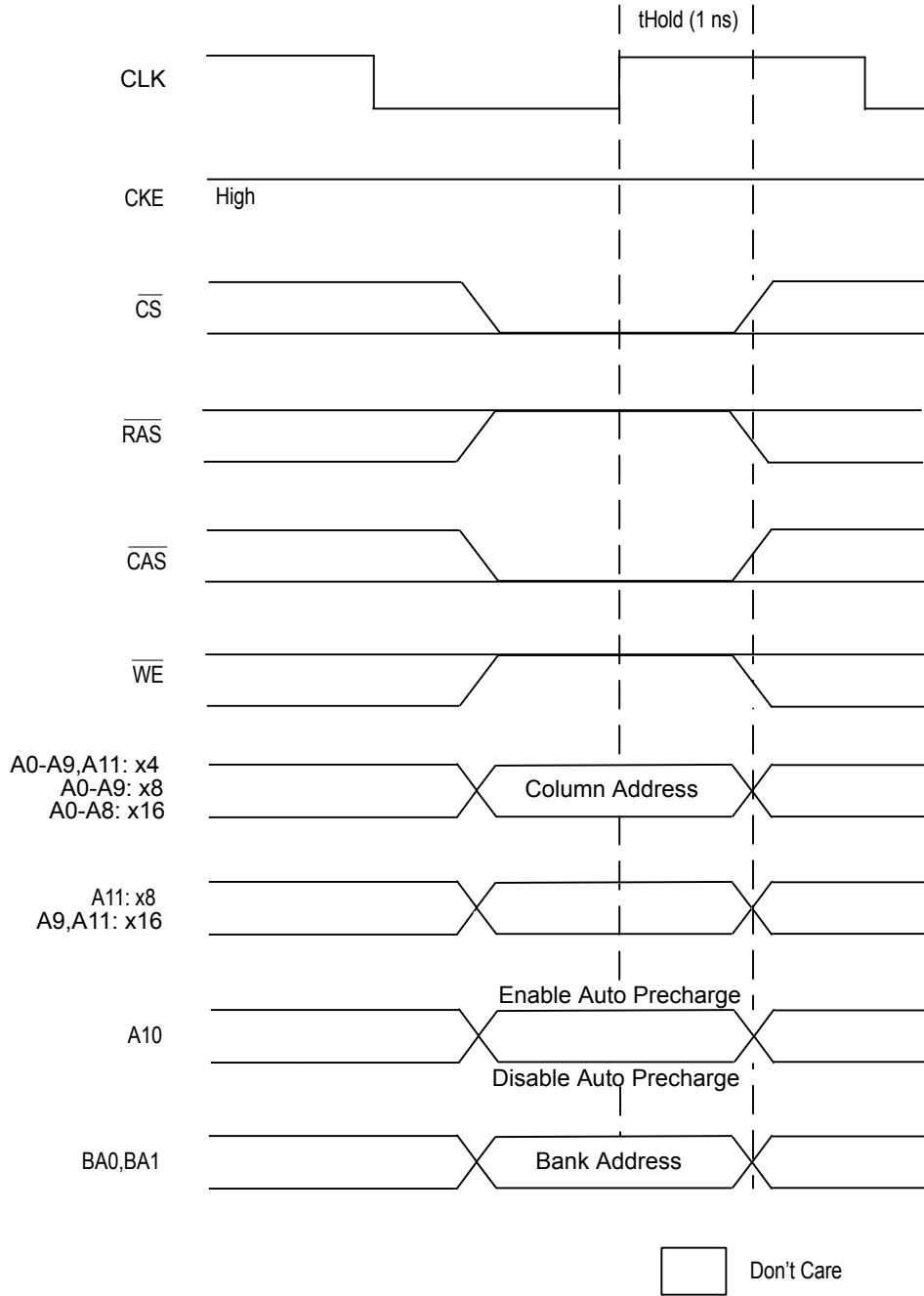
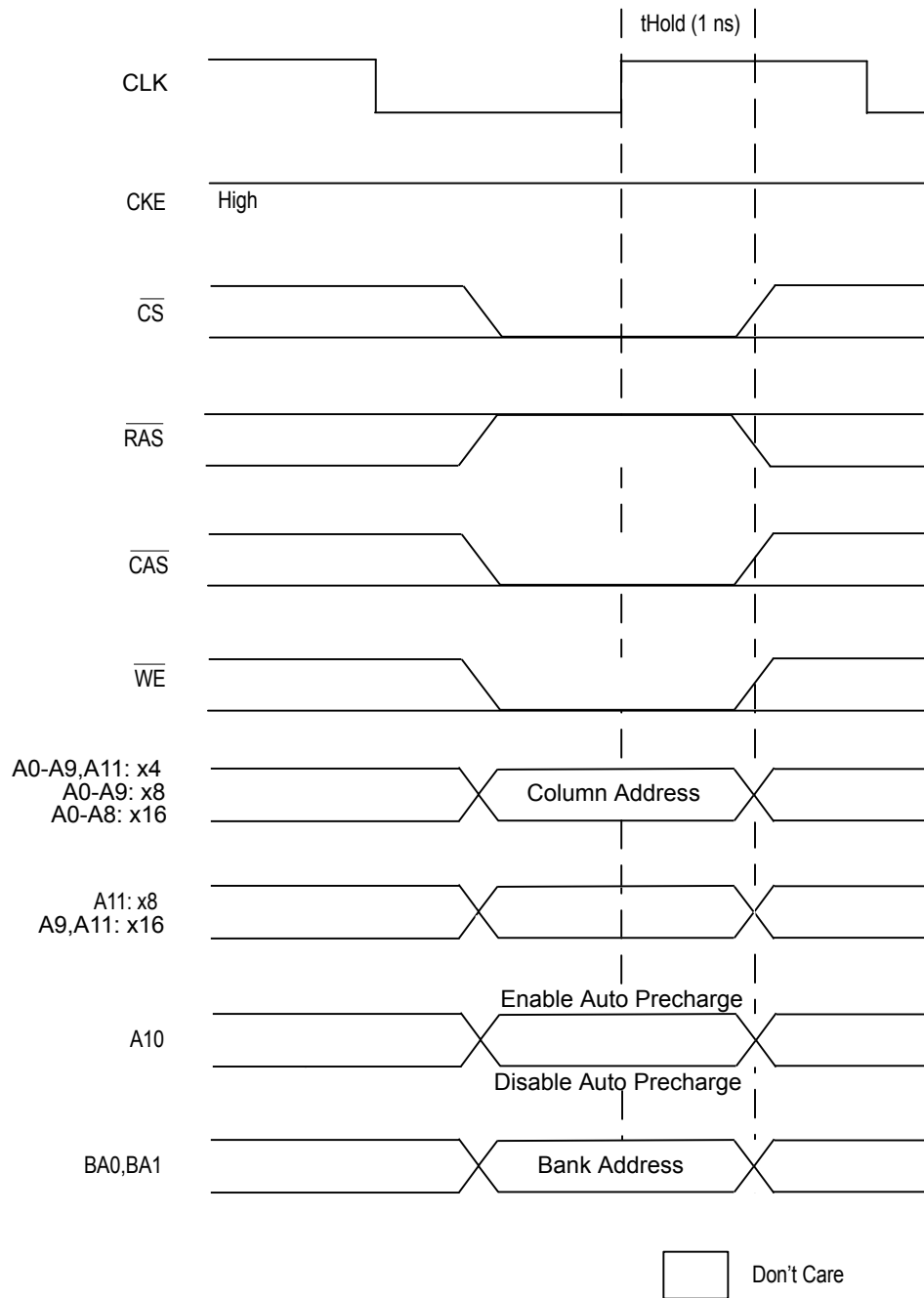


Figure 26-14. SDRAM Write Command Timing



**Figure 26-15. SDRAM Write Burst Timing**

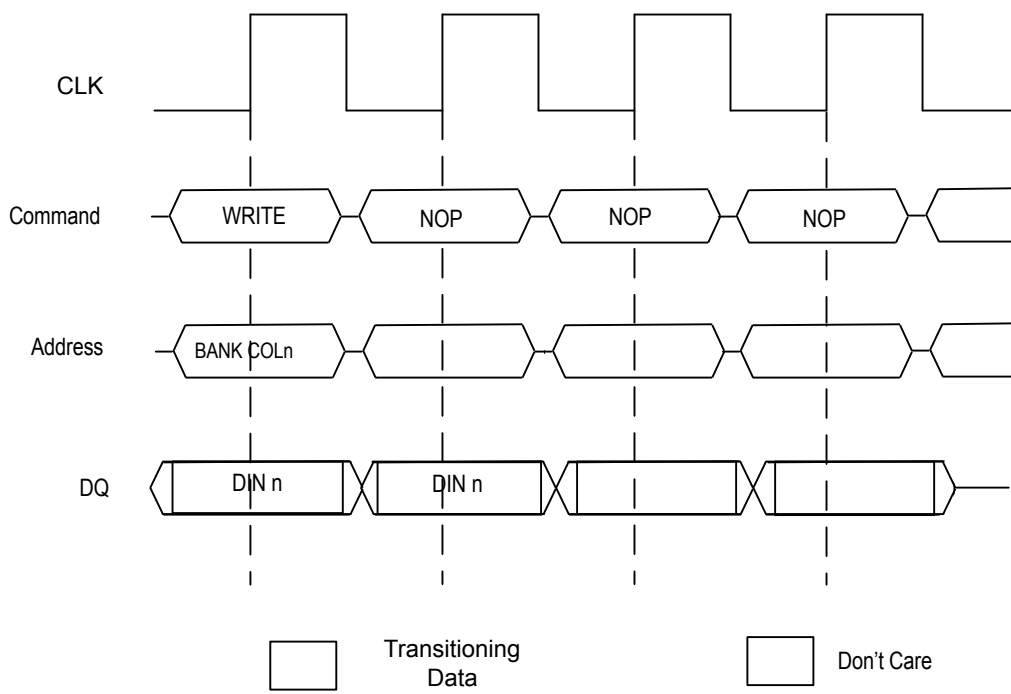




Figure 26-16. SDRAM Precharge Command Timing

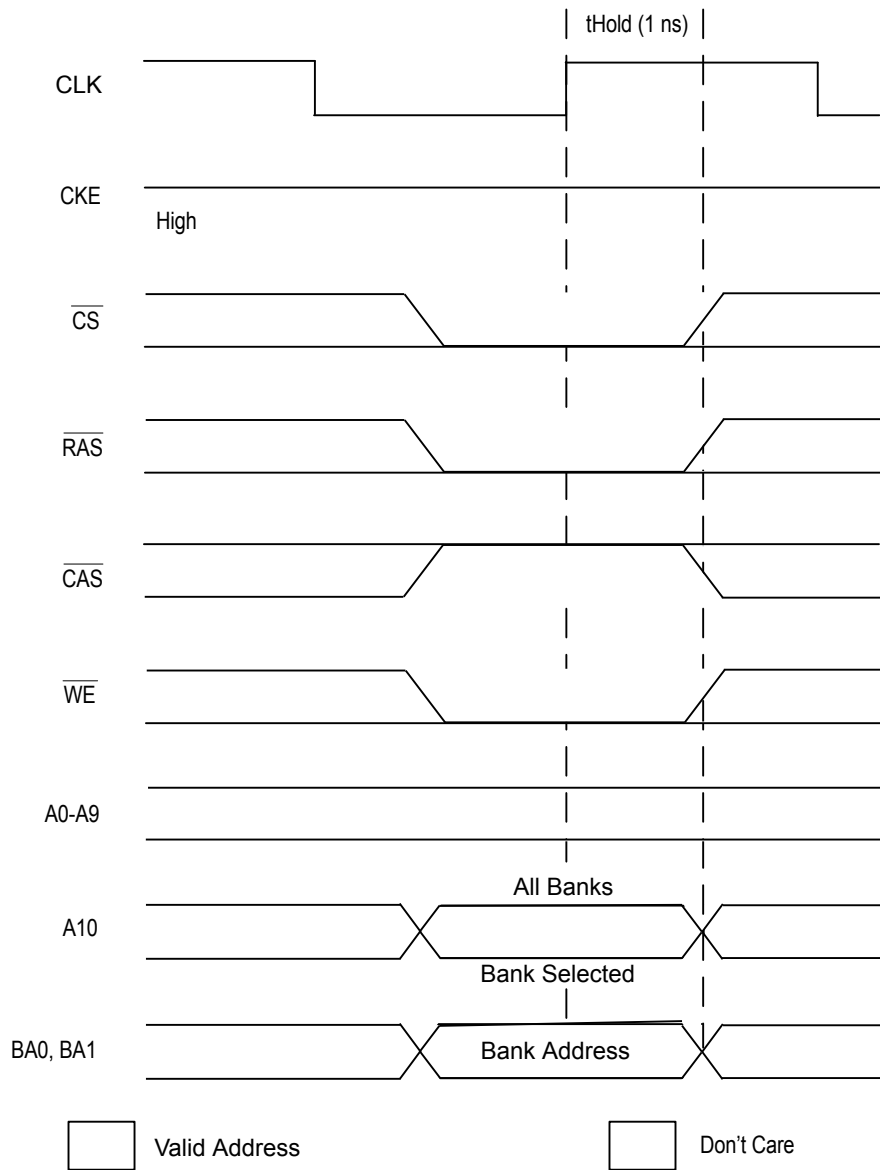
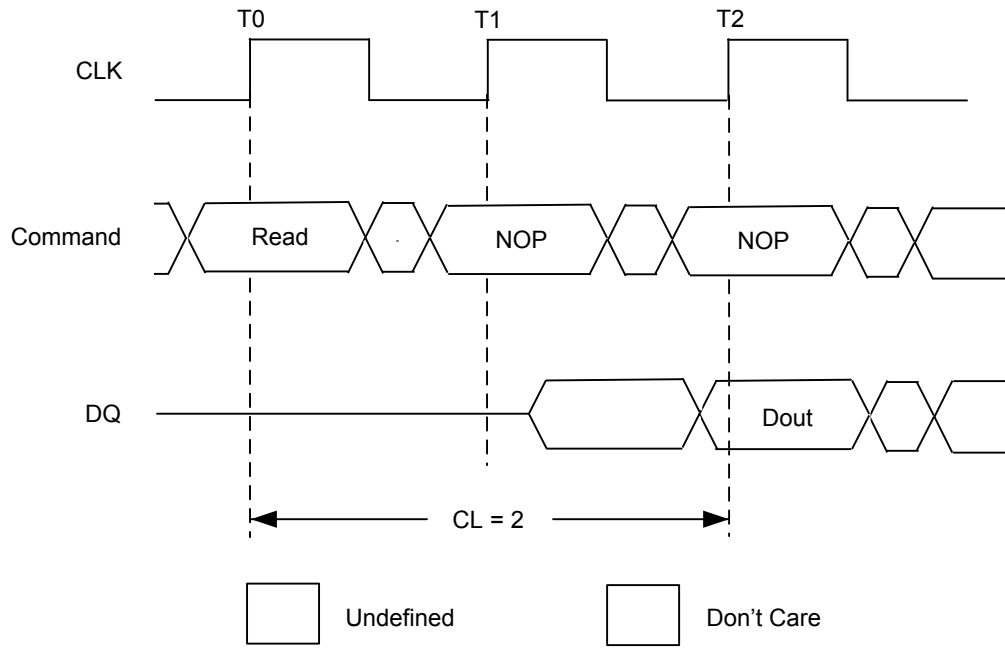
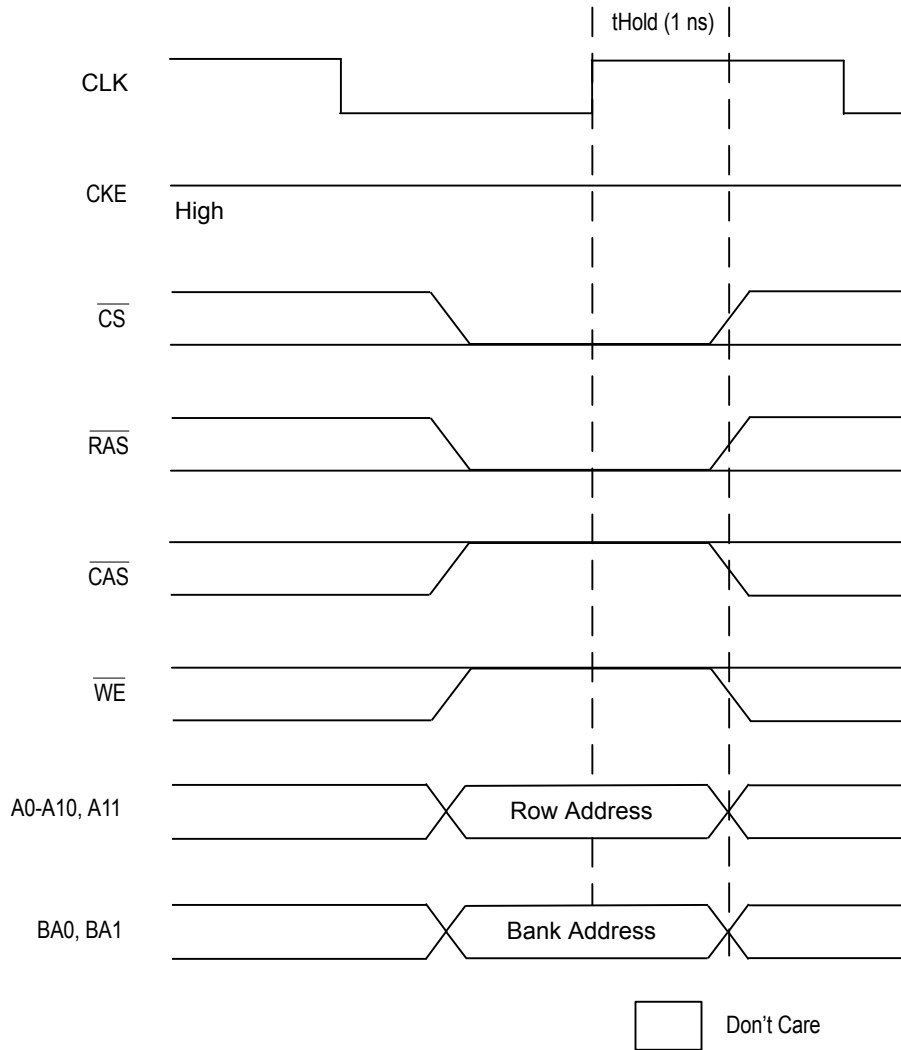


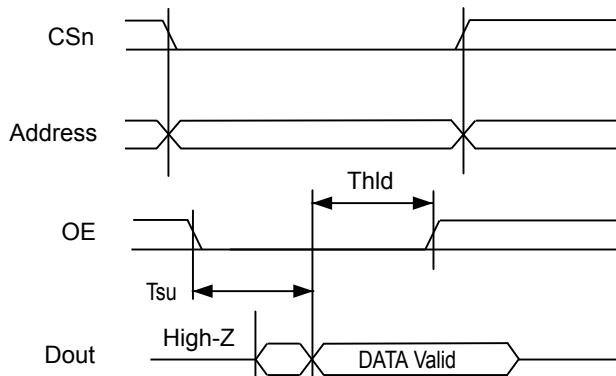
Figure 26-17. SDRAM CAS Latency Timing



**Figure 26-18. SDRAM Active Row Bank Timing**

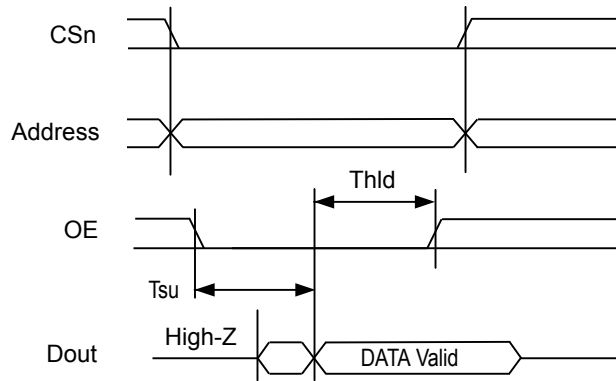


**Figure 26-19. SRAM Nor Read Timing**



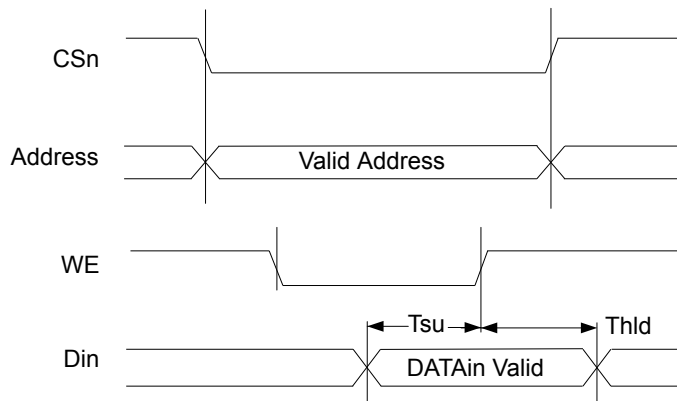
Note:  $T_{su} = T_{hld} = 1$  baud clock period

Figure 26-20. General-Purpose Mode Read Timing



Note:  $T_{su} = T_{hld} = 1$  baud clock period

Figure 26-21. General-Purpose Mode Write Timing



Note:  $T_{su} = T_{hld} = 1$  baud clock period

## 26.2.7 Analog-to-Digital Converter

Table 26-19. ADC Characteristics<sup>a</sup>

Parameter	Parameter Name	Min	Nom	Max	Unit
$V_{ADCIN}$	Maximum single-ended, full-scale analog input voltage	pending <sup>b</sup>	pending <sup>b</sup>	pending <sup>b</sup>	V
	Minimum single-ended, full-scale analog input voltage	pending <sup>b</sup>	pending <sup>b</sup>	pending <sup>b</sup>	V
	Maximum differential, full-scale analog input voltage	pending <sup>b</sup>	pending <sup>b</sup>	pending <sup>b</sup>	V
	Minimum differential, full-scale analog input voltage	pending <sup>b</sup>	pending <sup>b</sup>	pending <sup>b</sup>	V
$C_{ADCIN}$	Equivalent input capacitance	pending <sup>b</sup>	pending <sup>b</sup>	pending <sup>b</sup>	pF
N	Resolution	pending <sup>b</sup>	pending <sup>b</sup>	pending <sup>b</sup>	bits
$f_{ADC}$ <sub>c</sub>	ADC internal clock frequency	pending <sup>b</sup>	pending <sup>b</sup>	pending <sup>b</sup>	MHz

Parameter	Parameter Name	Min	Nom	Max	Unit
$t_{ADCCONV}$	Conversion time	pending <sup>b</sup>	pending <sup>b</sup>	pending <sup>b</sup>	$t_{ADC}$ cycles <sup>d</sup>
$f_{ADCCONV}$	Conversion rate	pending <sup>b</sup>	pending <sup>b</sup>	pending <sup>b</sup>	k samples/s
INL	Integral nonlinearity	pending <sup>b</sup>	pending <sup>b</sup>	pending <sup>b</sup>	LSB
DNL	Differential nonlinearity	pending <sup>b</sup>	pending <sup>b</sup>	pending <sup>b</sup>	LSB
OFF	Offset	pending <sup>b</sup>	pending <sup>b</sup>	pending <sup>b</sup>	LSB
GAIN	Gain	pending <sup>b</sup>	pending <sup>b</sup>	pending <sup>b</sup>	LSB

a. The ADC reference voltage is 3.0 V. This reference voltage is internally generated from the 3.3 VDDA supply by a band gap circuit.

b. Pending characterization completion.

c. The ADC must be clocked from the PLL or directly from a 14-MHz to 18-MHz clock source to operate properly.

d.  $t_{ADC} = 1/f_{ADC\ clock}$

## 26.2.8 Synchronous Serial Interface (SSI)

Table 26-20. SSI Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
S1	$t_{CLK\_PER}$	SSIClk cycle time	2	-	65024	system clocks
S2	$t_{CLK\_HIGH}$	SSIClk high time	-	0.5	-	t clk_per
S3	$t_{CLK\_LOW}$	SSIClk low time	-	0.5	-	t clk_per
S4	$t_{CLKRF}$	SSIClk rise/fall time	-	7.4	26	ns
S5	$t_{DMD}$	Data from master valid delay time	0	-	20	ns
S6	$t_{DMS}$	Data from master setup time	20	-	-	ns
S7	$t_{DMH}$	Data from master hold time	40	-	-	ns
S8	$t_{DSS}$	Data from slave setup time	20	-	-	ns
S9	$t_{DSH}$	Data from slave hold time	40	-	-	ns

Figure 26-22. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement

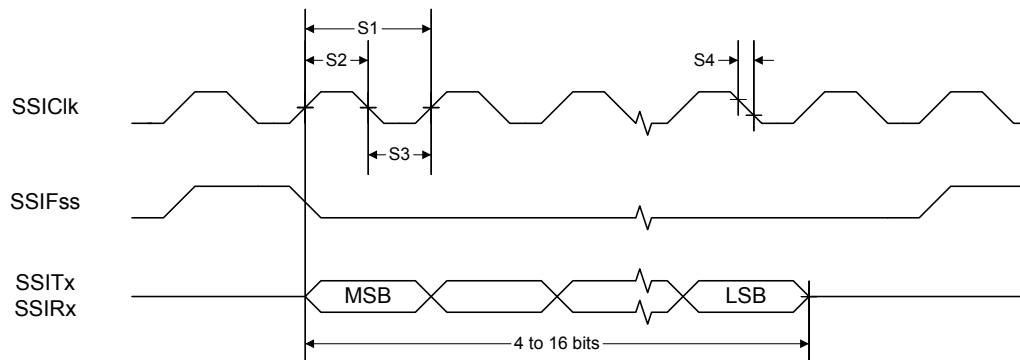


Figure 26-23. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer

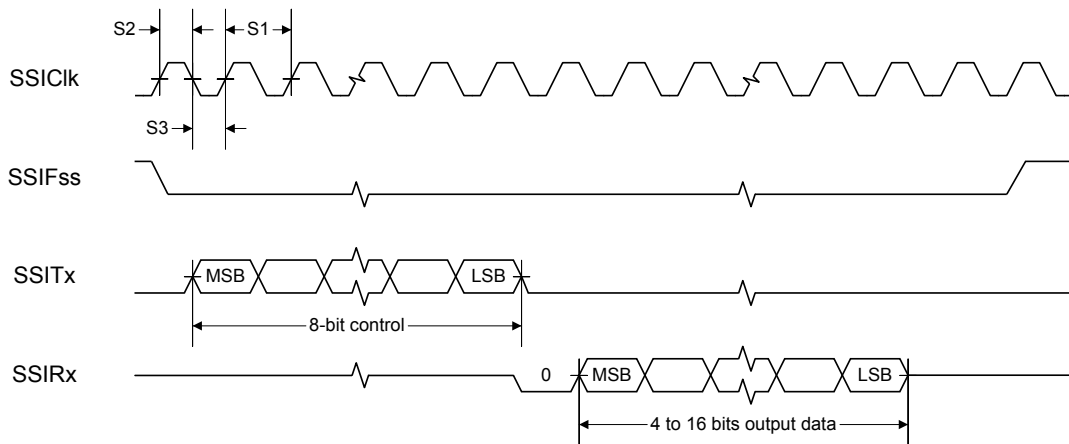
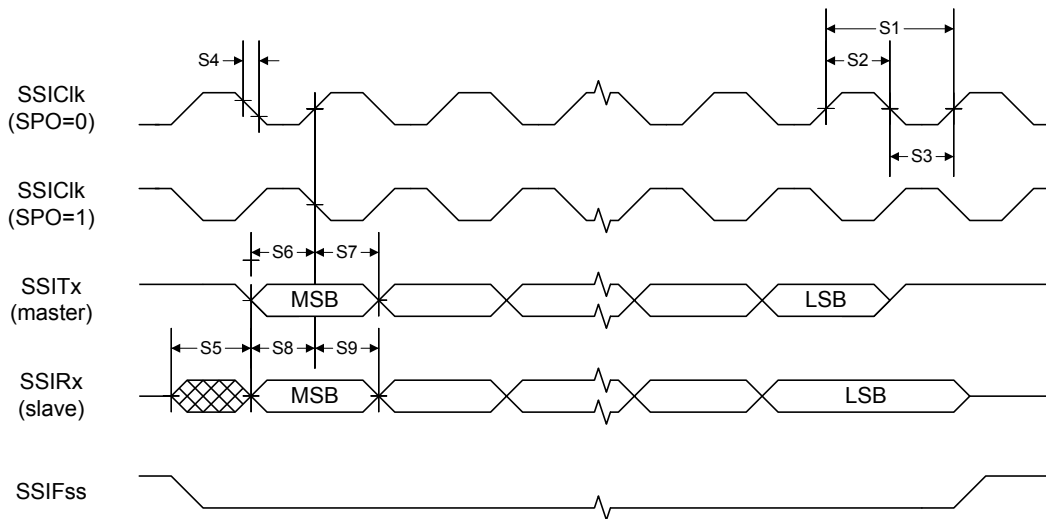
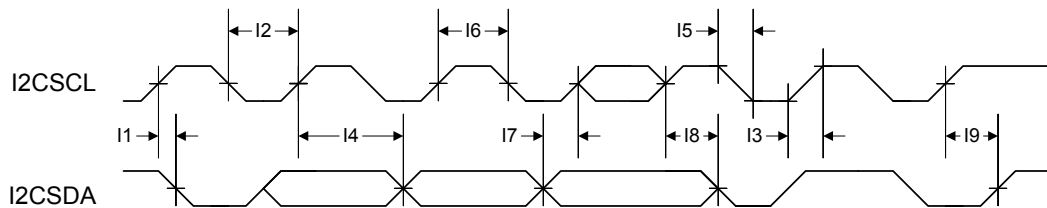


Figure 26-24. SSI Timing for SPI Frame Format (FRF=00), with SPH=1



## 26.2.9 Inter-Integrated Circuit (I<sup>2</sup>C) Interface

Figure 26-25. I<sup>2</sup>C Timing



## 26.2.10 Inter-Integrated Circuit Sound (I<sup>2</sup>S) Interface

**Table 26-21. I2S Master Clock (Receive and Transmit)**

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
M1	t <sub>MCLK_PER</sub>	Cycle time	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns
M2	t <sub>MCLKRF</sub>	Rise/fall time	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns
M3	t <sub>MCLK_HIGH</sub>	High time	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns
M4	t <sub>MCLK_LOW</sub>	Low time	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns
M5	t <sub>MDC</sub>	Duty cycle	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns
M6	t <sub>MJITTER</sub>	Jitter	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns

a. Pending characterization completion.

**Table 26-22. I2S Slave Clock (Receive and Transmit)**

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
M7	t <sub>SCLK_PER</sub>	Cycle time	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns
M8	t <sub>SCLK_HIGH</sub>	High time	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns
M9	t <sub>SCLK_LOW</sub>	Low time	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns
M10	t <sub>SDC</sub>	Duty cycle	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns

a. Pending characterization completion.

**Table 26-23. I2S Master Mode**

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
M11	t <sub>MFALL</sub>	SCK fall to WS valid	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns
M12	t <sub>MRISE</sub>	SCK rise to TXSD valid	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns
M13	t <sub>MRXSD</sub>	RXSD setup time to SCK rise	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns
M14	t <sub>MTXSD</sub>	RXSD hold time to SCK rise	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns

a. Pending characterization completion.

**Table 26-24. I2S Slave Mode**

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
M15	t <sub>SCLK_PER</sub>	Cycle time	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns
M16	t <sub>SCLK_HIGH</sub>	High time	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns
M17	t <sub>SCLK_LOW</sub>	Low time	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns
M18	t <sub>SDC</sub>	Duty cycle	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns
M19	t <sub>SSETUP</sub>	WS setup time to SCK fall	pending <sup>a</sup>	pending <sup>b</sup>	pending <sup>a</sup>	ns
M20	t <sub>SHOLD</sub>	WS hold time to SCK fall	pending <sup>a</sup>	pending <sup>c</sup>	pending <sup>a</sup>	ns
M21	t <sub>SRISE</sub>	SCK rise to TXSD valid	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns
M22	t <sub>SRXSD</sub>	RXSD setup time to SCK rise	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns
M23	t <sub>STXSD</sub>	RXSD hold time to SCK rise	pending <sup>a</sup>	pending <sup>a</sup>	pending <sup>a</sup>	ns

a. Pending characterization completion.

b. Pending characterization completion.

c. Pending characterization completion.

### 26.2.11 Universal Serial Bus (USB) Controller

The Stellaris<sup>®</sup> USB controller AC electrical specifications are compliant with the “Universal Serial Bus Specification Rev. 2.0” (full-speed and low-speed support) and the “On-The-Go Supplement to the USB 2.0 Specification Rev. 1.0”.

### 26.2.12 Analog Comparator

**Table 26-25. Analog Comparator Characteristics**

Parameter	Parameter Name	Min	Nom	Max	Unit
V <sub>OS</sub>	Input offset voltage	-	±10	±25	mV
V <sub>CM</sub>	Input common mode voltage range	0	-	V <sub>DD</sub> -1.5	V
C <sub>MRR</sub>	Common mode rejection ratio	50	-	-	dB
T <sub>RT</sub>	Response time	-	-	1	μs
T <sub>MC</sub>	Comparator mode change to Output Valid	-	-	10	μs

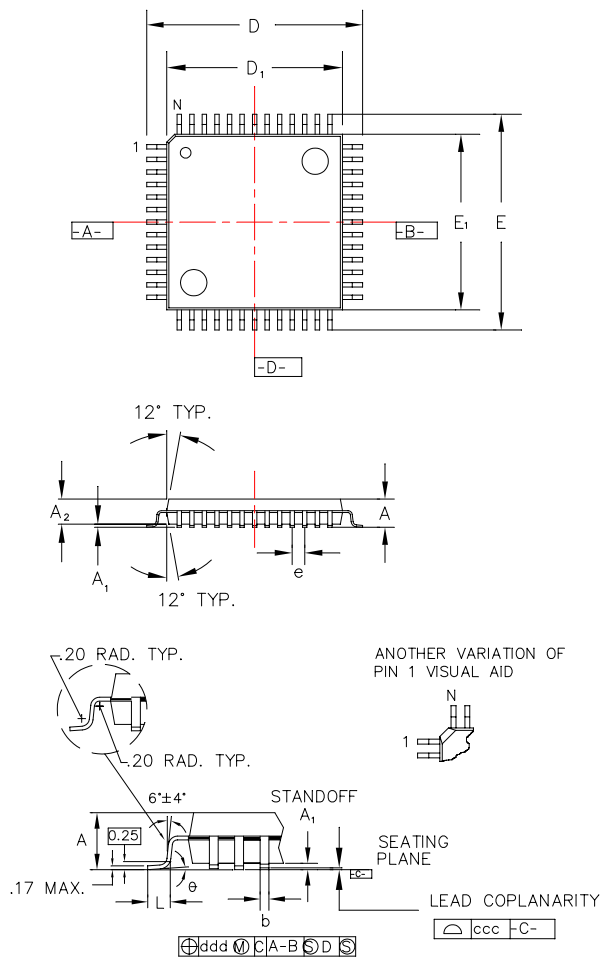
**Table 26-26. Analog Comparator Voltage Reference Characteristics**

Parameter	Parameter Name	Min	Nom	Max	Unit
R <sub>HR</sub>	Resolution high range	-	V <sub>DD</sub> /31	-	LSB
R <sub>LR</sub>	Resolution low range	-	V <sub>DD</sub> /23	-	LSB
A <sub>HR</sub>	Absolute accuracy high range	-	-	±1/2	LSB
A <sub>LR</sub>	Absolute accuracy low range	-	-	±1/4	LSB



## 27 Package Information

Figure 27-1. 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 <sub>1</sub>	-	0.05 Min./0.15 Max.
A <sub>2</sub>	±0.05	1.40
D	±0.20	16.00
D <sub>1</sub>	±0.05	14.00
E	±0.20	16.00
E <sub>1</sub>	±0.05	14.00
L	+0.15/-0.10	0.60
e	Basic	0.50
b	+0.05	0.22
θ	-	0°-7°
ddd	Max.	0.08
ccc	Max.	0.08
JEDEC Reference Drawing		MS-026
Variation Designator		BED

## A Boot Loader

### A.1 Boot Loader

The Stellaris<sup>®</sup> Boot Loader is executed from the ROM when flash is empty and is used to download code to the flash memory of a device without the use of a debug interface. The boot loader uses a simple packet interface to provide synchronous communication with the device. The boot loader runs off the internal oscillator and does not enable the PLL, so its speed is determined by the speed of the internal oscillator. The following serial interfaces can be used:

- UART0
- SSIO
- I<sup>2</sup>C0

For simplicity, both the data format and communication protocol are identical for all serial interfaces. See the *Stellaris<sup>®</sup> Boot Loader User's Guide* for information on the boot loader software.

### A.2 Interfaces

Once communication with the boot loader is established via one of the serial interfaces, that interface is used until the boot loader is reset or new code takes over. For example, once you start communicating using the SSI port, communications with the boot loader via the UART are disabled until the device is reset.

#### A.2.1 UART

The Universal Asynchronous Receivers/Transmitters (UART) communication uses a fixed serial format of 8 bits of data, no parity, and 1 stop bit. The baud rate used for communication is automatically detected by the boot loader and can be any valid baud rate supported by the host and the device. The auto detection sequence requires that the baud rate should be no more than 1/32 the internal oscillator frequency of the board that is running the boot loader (which is at least 8.4 MHz, providing support for up to 262,500 baud). This is actually the same as the hardware limitation for the maximum baud rate for any UART on a Stellaris<sup>®</sup> device which is calculated as follows:

$$\text{Max Baud Rate} = \text{System Clock Frequency} / 16$$

In order to determine the baud rate, the boot loader needs to determine the relationship between the internal oscillator and the baud rate. This is enough information for the boot loader to configure its UART to the same baud rate as the host. This automatic baud-rate detection allows the host to use any valid baud rate that it wants to communicate with the device.

The method used to perform this automatic synchronization relies on the host sending the boot loader two bytes that are both 0x55. This generates a series of pulses to the boot loader that it can use to calculate the ratios needed to program the UART to match the host's baud rate. After the host sends the pattern, it attempts to read back one byte of data from the UART. The boot loader returns the value of 0xCC to indicate successful detection of the baud rate. If this byte is not received after at least twice the time required to transfer the two bytes, the host can resend another pattern of 0x55, 0x55, and wait for the 0xCC byte again until the boot loader acknowledges that it has received a synchronization pattern correctly. For example, the time to wait for data back from the boot loader should be calculated as at least  $2 * (20 \text{ (bits/sync)} / \text{baud rate (bits/sec)})$ . 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 566 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 internal oscillator frequency of the board running the boot loader (which is at least 8.4 MHz, providing support for up to 700 KHz).. Since the host device is the master, the SSI on the boot loader device does not need to determine the clock as it is provided directly by the host.

## A.2.3 I<sup>2</sup>C

The Inter-Integrated Circuit (I<sup>2</sup>C) port operates in slave mode with a slave address of 0x42. The I<sup>2</sup>C port will work at both 100 KHz and 400 KHz I<sup>2</sup>C clock frequency. Since the host device is the master, the I<sup>2</sup>C on the boot loader device does not need to determine the clock as it is provided directly by the host.

## A.3 Packet Handling

All communications, with the exception of the UART auto-baud, are done via defined packets that are acknowledged (ACK) or not acknowledged (NAK) by the devices. The packets use the same format for receiving and sending packets, including the method used to acknowledge successful or unsuccessful reception of a packet.

### A.3.1 Packet Format

All packets sent and received from the device use the following byte-packed format.

```
struct
{
  unsigned char ucSize;
  unsigned char ucChecksum;
  unsigned char Data[];
};
```

ucSize	The first byte received holds the total size of the transfer including the size and checksum bytes.
ucChecksum	This holds a simple checksum of the bytes in the data buffer only. The algorithm is Data[0]+Data[1]+...+ Data[ucSize-3].
Data	This is the raw data intended for the device, which is formatted in some form of command interface. There should be ucSize-2 bytes of data provided in this buffer to or from the device.

### A.3.2 Sending Packets

The actual bytes of the packet can be sent individually or all at once; the only limitation is that commands that cause flash memory access should limit the download sizes to prevent losing bytes during flash programming. This limitation is discussed further in the section that describes the boot loader command, COMMAND\_SEND\_DATA (see “COMMAND\_SEND\_DATA (0x24)” on page 970).

Once the packet has been formatted correctly by the host, it should be sent out over the UART or SSI interface. Then the host should poll the UART or SSI interface for the first non-zero data returned from the device. The first non-zero byte will either be an ACK (0xCC) or a NAK (0x33) byte from

the device indicating the packet was received successfully (ACK) or unsuccessfully (NAK). This does not indicate that the actual contents of the command issued in the data portion of the packet were valid, just that the packet was received correctly.

### A.3.3 Receiving Packets

The boot loader sends a packet of data in the same format that it receives a packet. The boot loader may transfer leading zero data before the first actual byte of data is sent out. The first non-zero byte is the size of the packet followed by a checksum byte, and finally followed by the data itself. There is no break in the data after the first non-zero byte is sent from the boot loader. Once the device communicating with the boot loader receives all the bytes, it must either ACK or NAK the packet to indicate that the transmission was successful. The appropriate response after sending a NAK to the boot loader is to resend the command that failed and request the data again. If needed, the host may send leading zeros before sending down the ACK/NAK signal to the boot loader, as the boot loader only accepts the first non-zero data as a valid response. This zero padding is needed by the SSI interface in order to receive data to or from the boot loader.

## A.4 Commands

The next section defines the list of commands that can be sent to the boot loader. The first byte of the data should always be one of the defined commands, followed by data or parameters as determined by the command that is sent.

### A.4.1 COMMAND\_PING (0X20)

This command simply accepts the command and sets the global status to success. The format of the packet is as follows:

```
Byte[0] = 0x03;
Byte[1] = checksum(Byte[2]);
Byte[2] = COMMAND_PING;
```

The ping command has 3 bytes and the value for `COMMAND_PING` is 0x20 and the checksum of one byte is that same byte, making `Byte[1]` also 0x20. Since the ping command has no real return status, the receipt of an ACK can be interpreted as a successful ping to the boot loader.

### A.4.2 COMMAND\_DOWNLOAD (0x21)

This command is sent to the boot loader to indicate where to store data and how many bytes will be sent by the `COMMAND_SEND_DATA` commands that follow. The command consists of two 32-bit values that are both transferred MSB first. The first 32-bit value is the address to start programming data into, while the second is the 32-bit size of the data that will be sent. This command also triggers an erase of the full area to be programmed so this command takes longer than other commands. This results in a longer time to receive the ACK/NAK back from the board. This command should be followed by a `COMMAND_GET_STATUS` to ensure that the Program Address and Program size are valid for the device running the boot loader.

The format of the packet to send this command is as follows:

```
Byte[0] = 11
Byte[1] = checksum(Bytes[2:10])
Byte[2] = COMMAND_DOWNLOAD
Byte[3] = Program Address [31:24]
Byte[4] = Program Address [23:16]
Byte[5] = Program Address [15:8]
```

```
Byte[6] = Program Address [7:0]
Byte[7] = Program Size [31:24]
Byte[8] = Program Size [23:16]
Byte[9] = Program Size [15:8]
Byte[10] = Program Size [7:0]
```

### **A.4.3 COMMAND\_RUN (0x22)**

This command is used to tell the boot loader to execute from the address passed as the parameter in this command. This command consists of a single 32-bit value that is interpreted as the address to execute. The 32-bit value is transmitted MSB first and the boot loader responds with an ACK signal back to the host device before actually executing the code at the given address. This allows the host to know that the command was received successfully and the code is now running.

```
Byte[0] = 7
Byte[1] = checksum(Bytes[2:6])
Byte[2] = COMMAND_RUN
Byte[3] = Execute Address[31:24]
Byte[4] = Execute Address[23:16]
Byte[5] = Execute Address[15:8]
Byte[6] = Execute Address[7:0]
```

### **A.4.4 COMMAND\_GET\_STATUS (0x23)**

This command returns the status of the last command that was issued. Typically, this command should be sent after every command to ensure that the previous command was successful or to properly respond to a failure. The command requires one byte in the data of the packet and should be followed by reading a packet with one byte of data that contains a status code. The last step is to ACK or NAK the received data so the boot loader knows that the data has been read.

```
Byte[0] = 0x03
Byte[1] = checksum(Byte[2])
Byte[2] = COMMAND_GET_STATUS
```

### **A.4.5 COMMAND\_SEND\_DATA (0x24)**

This command should only follow a `COMMAND_DOWNLOAD` command or another `COMMAND_SEND_DATA` command if more data is needed. Consecutive send data commands automatically increment address and continue programming from the previous location. For packets which do not contain the final portion of the downloaded data, a multiple of four bytes should always be transferred. The command terminates programming once the number of bytes indicated by the `COMMAND_DOWNLOAD` command has been received. Each time this function is called it should be followed by a `COMMAND_GET_STATUS` to ensure that the data was successfully programmed into the flash. If the boot loader sends a NAK to this command, the boot loader does not increment the current address to allow retransmission of the previous data. The following example shows a `COMMAND_SEND_DATA` packet with 8 bytes of packet data:

```
Byte[0] = 11
Byte[1] = checksum(Bytes[2:10])
Byte[2] = COMMAND_SEND_DATA
Byte[3] = Data[0]
Byte[4] = Data[1]
Byte[5] = Data[2]
Byte[6] = Data[3]
```

```
Byte[7] = Data[4]  
Byte[8] = Data[5]  
Byte[9] = Data[6]  
Byte[10] = Data[7]
```

#### **A.4.6 COMMAND\_RESET (0x25)**

This command is used to tell the boot loader device to reset. Unlike the `COMMAND_RUN` command, this allows the initial stack pointer to be read by the hardware and set up for the new code. It can also be used to reset the boot loader if a critical error occurs and the host device wants to restart communication with the boot loader.

```
Byte[0] = 3  
Byte[1] = checksum(Byte[2])  
Byte[2] = COMMAND_RESET
```

The boot loader responds with an ACK signal back to the host device before actually executing the software reset to the device running the boot loader. This allows the host to know that the command was received successfully and the part will be reset.

## B ROM DriverLib Functions

### B.1 DriverLib Functions Included in the Integrated ROM

The Stellaris<sup>®</sup> Peripheral Driver Library (DriverLib) APIs that are available in the integrated ROM of the Stellaris<sup>®</sup> family of devices are listed below. The detailed description of each function is available in the *Stellaris<sup>®</sup> ROM User's Guide*.

```
ROM_ADCHardwareOversampleConfigure
    // Configures the hardware oversampling factor of the ADC.

ROM_ADCIntClear
    // Clears sample sequence interrupt source.

ROM_ADCIntDisable
    // Disables a sample sequence interrupt.

ROM_ADCIntEnable
    // Enables a sample sequence interrupt.

ROM_ADCIntStatus
    // Gets the current interrupt status.

ROM_ADCProcessorTrigger
    // Causes a processor trigger for a sample sequence.

ROM_ADCSequenceConfigure
    // Configures the trigger source and priority of a sample sequence.

ROM_ADCSequenceDataGet
    // Gets the captured data for a sample sequence.

ROM_ADCSequenceDisable
    // Disables a sample sequence.

ROM_ADCSequenceEnable
    // Enables a sample sequence.

ROM_ADCSequenceOverflow
    // Determines if a sample sequence overflow occurred.

ROM_ADCSequenceOverflowClear
    // Clears the overflow condition on a sample sequence.

ROM_ADCSequenceStepConfigure
    // Configure a step of the sample sequencer.

ROM_ADCSequenceUnderflow
    // Determines if a sample sequence underflow occurred.

ROM_ADCSequenceUnderflowClear
    // Clears the underflow condition on a sample sequence.
```



---

```
ROM_CANBitTimingGet
    // Reads the current settings for the CAN controller bit timing.

ROM_CANBitTimingSet
    // Configures the CAN controller bit timing.

ROM_CANDisable
    // Disables the CAN controller.

ROM_CANEnable
    // Enables the CAN controller.

ROM_CANErrCntGet
    // Reads the CAN controller error counter register.

ROM_CANInit
    // Initializes the CAN controller after reset.

ROM_CANIntClear
    // Clears a CAN interrupt source.

ROM_CANIntDisable
    // Disables individual CAN controller interrupt sources.

ROM_CANIntEnable
    // Enables individual CAN controller interrupt sources.

ROM_CANIntStatus
    // Returns the current CAN controller interrupt status.

ROM_CANMessageClear
    // Clears a message object so that it is no longer used.

ROM_CANMessageGet
    // Reads a CAN message from one of the message object buffers.

ROM_CANMessageSet
    // Configures a message object in the CAN controller.

ROM_CANRetryGet
    // Returns the current setting for automatic retransmission.

ROM_CANRetrySet
    // Sets the CAN controller automatic retransmission behavior.

ROM_CANStatusGet
    // Reads one of the controller status registers.

ROM_ComparatorConfigure
    // Configures a comparator.

ROM_ComparatorIntClear
    // Clears a comparator interrupt.
```

ROM\_ComparatorIntDisable  
// Disables the comparator interrupt.

ROM\_ComparatorIntEnable  
// Enables the comparator interrupt.

ROM\_ComparatorIntStatus  
// Gets the current interrupt status.

ROM\_ComparatorRefSet  
// Sets the internal reference voltage.

ROM\_ComparatorValueGet  
// Gets the current comparator output value.

ROM\_FlashErase  
// Erases a block of flash.

ROM\_FlashIntClear  
// Clears flash controller interrupt sources.

ROM\_FlashIntDisable  
// Disables individual flash controller interrupt sources.

ROM\_FlashIntEnable  
// Enables individual flash controller interrupt sources.

ROM\_FlashIntGetStatus  
// Gets the current interrupt status.

ROM\_FlashProgram  
// Programs flash.

ROM\_FlashProtectGet  
// Gets the protection setting for a block of flash.

ROM\_FlashProtectSave  
// Saves the flash protection settings.

ROM\_FlashProtectSet  
// Sets the protection setting for a block of flash.

ROM\_FlashUsecGet  
// Gets the number of processor clocks per micro-second.

ROM\_FlashUsecSet  
// Sets the number of processor clocks per micro-second.

ROM\_FlashUserGet  
// Gets the user registers.

ROM\_FlashUserSave  
// Saves the user registers.

```
ROM_FlashUserSet
    // Sets the user registers.

ROM_GPIODirModeGet
    // Gets the direction and mode of a pin.

ROM_GPIODirModeSet
    // Sets the direction and mode of the specified pin(s).

ROM_GPIOWriteTypeGet
    // Gets the interrupt type for a pin.

ROM_GPIOWriteTypeSet
    // Sets the interrupt type for the specified pin(s).

ROM_GPIOPadConfigGet
    // Gets the pad configuration for a pin.

ROM_GPIOPadConfigSet
    // Sets the pad configuration for the specified pin(s).

ROM_GPIOPinIntClear
    // Clears the interrupt for the specified pin(s).

ROM_GPIOPinIntDisable
    // Disables interrupts for the specified pin(s).

ROM_GPIOPinIntEnable
    // Enables interrupts for the specified pin(s).

ROM_GPIOPinIntStatus
    // Gets interrupt status for the specified GPIO port.

ROM_GPIOPinRead
    // Reads the values present of the specified pin(s).

ROM_GPIOPinTypeADC
    // Configures pin(s) for use as analog-to-digital converter inputs.

ROM_GPIOPinTypeCAN
    // Configures pin(s) for use as a CAN device.

ROM_GPIOPinTypeComparator
    // Configures pin(s) for use as an analog comparator input.

ROM_GPIOPinTypeGPIOInput
    // Configures pin(s) for use as GPIO inputs.

ROM_GPIOPinTypeGPIOOutput
    // Configures pin(s) for use as GPIO outputs.

ROM_GPIOPinTypeGPIOOutputOD
    // Configures pin(s) for use as GPIO open drain outputs.
```

ROM\_GPIOPinTypeI2C  
// Configures pin(s) for use by the I2C peripheral.

ROM\_GPIOPinTypePWM  
// Configures pin(s) for use by the PWM peripheral.

ROM\_GPIOPinTypeQEI  
// Configures pin(s) for use by the QEI peripheral.

ROM\_GPIOPinTypeSSI  
// Configures pin(s) for use by the SSI peripheral.

ROM\_GPIOPinTypeTimer  
// Configures pin(s) for use by the Timer peripheral.

ROM\_GPIOPinTypeUART  
// Configures pin(s) for use by the UART peripheral.

ROM\_GPIOPinTypeUSBDigital  
// Configures pin(s) for use by the USB peripheral.

ROM\_GPIOPinWrite  
// Writes a value to the specified pin(s).

ROM\_I2CMasterBusBusy  
// Indicates whether or not the I2C bus is busy.

ROM\_I2CMasterBusy  
// Indicates whether or not the I2C Master is busy.

ROM\_I2CMasterControl  
// Controls the state of the I2C Master module.

ROM\_I2CMasterDataGet  
// Receives a byte that has been sent to the I2C Master.

ROM\_I2CMasterDataPut  
// Transmits a byte from the I2C Master.

ROM\_I2CMasterDisable  
// Disables the I2C master block.

ROM\_I2CMasterEnable  
// Enables the I2C Master block.

ROM\_I2CMasterErr  
// Gets the error status of the I2C Master module.

ROM\_I2CMasterInitExpClk  
// Initializes the I2C Master block.

ROM\_I2CMasterIntClear  
// Clears I2C Master interrupt sources.

```
ROM_I2CMasterIntDisable
    // Disables the I2C Master interrupt.

ROM_I2CMasterIntEnable
    // Enables the I2C Master interrupt.

ROM_I2CMasterIntStatus
    // Gets the current I2C Master interrupt status.

ROM_I2CMasterSlaveAddrSet
    // Sets the address that the I2C Master will place on the bus.

ROM_I2CSlaveDataGet
    // Receives a byte that has been sent to the I2C Slave.

ROM_I2CSlaveDataPut
    // Transmits a byte from the I2C Slave.

ROM_I2CSlaveDisable
    // Disables the I2C slave block.

ROM_I2CSlaveEnable
    // Enables the I2C Slave block.

ROM_I2CSlaveInit
    // Initializes the I2C Slave block.

ROM_I2CSlaveIntClear
    // Clears I2C Slave interrupt sources.

ROM_I2CSlaveIntDisable
    // Disables the I2C Slave interrupt.

ROM_I2CSlaveIntEnable
    // Enables the I2C Slave interrupt.

ROM_I2CSlaveIntStatus
    // Gets the current I2C Slave interrupt status.

ROM_I2CSlaveStatus
    // Gets the I2C Slave module status.

ROM_IntDisable
    // Disables an interrupt.

ROM_IntEnable
    // Enables an interrupt.

ROM_IntMasterDisable
    // Disables the processor interrupt.

ROM_IntMasterEnable
    // Enables the processor interrupt.
```

ROM\_IntPriorityGet  
// Gets the priority of an interrupt.

ROM\_IntPriorityGroupingGet  
// Gets the priority grouping of the interrupt controller.

ROM\_IntPriorityGroupingSet  
// Sets the priority grouping of the interrupt controller.

ROM\_IntPrioritySet  
// Sets the priority of an interrupt.

ROM\_MPUDisable  
// Disables the MPU for use.

ROM\_MPUEnable  
// Enables and configures the MPU for use.

ROM\_MPURegionCountGet  
// Gets the count of regions supported by th MPU.

ROM\_MPURegionDisable  
// Disables a specific region.

ROM\_MPURegionEnable  
// Enables a specific region.

ROM\_MPURegionGet  
// Gets the current settings for a specific region.

ROM\_MPURegionSet  
// Sets up the access rules for a specific region.

ROM\_PWMDeadBandDisable  
// Disables the PWM dead band output.

ROM\_PWMDeadBandEnable  
// Enables the PWM dead band output, and sets the dead band delays.

ROM\_PWMFaultIntClear  
// Clears the fault interrupt for a PWM module.

ROM\_PWMFaultIntClearExt  
// Clears the fault interrupt for a PWM module.

ROM\_PWMGenConfigure  
// Configures a PWM generator.

ROM\_PWMGenDisable  
// Disables the timer/counter for a PWM generator block.

ROM\_PWMGenEnable  
// Enables the timer/counter for a PWM generator block.

ROM\_PWMGenFaultClear  
// Clears one or more latched fault triggers for a given PWM generator.

ROM\_PWMGenFaultConfigure  
// Configures the minimum fault period and fault pin senses for a given PWM generator.

ROM\_PWMGenFaultStatus  
// Returns the current state of the fault triggers for a given PWM generator.

ROM\_PWMGenFaultTriggerGet  
// Returns the set of fault triggers currently configured for a given PWM generator.

ROM\_PWMGenFaultTriggerSet  
// Configures the set of fault triggers for a given PWM generator.

ROM\_PWMGenIntClear  
// Clears the specified interrupt(s) for the specified PWM generator block.

ROM\_PWMGenIntStatus  
// Gets interrupt status for the specified PWM generator block.

ROM\_PWMGenIntTrigDisable  
// Disables interrupts for the specified PWM generator block.

ROM\_PWMGenIntTrigEnable  
// Enables interrupts and triggers for the specified PWM generator block.

ROM\_PWMGenPeriodGet  
// Gets the period of a PWM generator block.

ROM\_PWMGenPeriodSet  
// Set the period of a PWM generator.

ROM\_PWMIntDisable  
// Disables generator and fault interrupts for a PWM module.

ROM\_PWMIntEnable  
// Enables generator and fault interrupts for a PWM module.

ROM\_PWMIntStatus  
// Gets the interrupt status for a PWM module.

ROM\_PWMOutputFault  
// Specifies the state of PWM outputs in response to a fault condition.

ROM\_PWMOutputFaultLevel  
// Specifies the level of PWM outputs suppressed in response to a fault condition.

ROM\_PWMOutputInvert  
// Selects the inversion mode for PWM outputs.

ROM\_PWMOutputState  
// Enables or disables PWM outputs.

ROM\_PWMPulseWidthGet  
// Gets the pulse width of a PWM output.

ROM\_PWMPulseWidthSet  
// Sets the pulse width for the specified PWM output.

ROM\_PWMSyncTimeBase  
// Synchronizes the counters in one or multiple PWM generator blocks.

ROM\_PWMSyncUpdate  
// Synchronizes all pending updates.

ROM\_QEIConfigure  
// Configures the quadrature encoder.

ROM\_QEIDirectionGet  
// Gets the current direction of rotation.

ROM\_QEIDisable  
// Disables the quadrature encoder.

ROM\_QEIEnable  
// Enables the quadrature encoder.

ROM\_QEIErrorGet  
// Gets the encoder error indicator.

ROM\_QEIIntClear  
// Clears quadrature encoder interrupt sources.

ROM\_QEIIntDisable  
// Disables individual quadrature encoder interrupt sources.

ROM\_QEIIntEnable  
// Enables individual quadrature encoder interrupt sources.

ROM\_QEIIntStatus  
// Gets the current interrupt status.

ROM\_QEIPositionGet  
// Gets the current encoder position.

ROM\_QEIPositionSet  
// Sets the current encoder position.

ROM\_QEIVelocityConfigure  
// Configures the velocity capture.

ROM\_QEIVelocityDisable  
// Disables the velocity capture.

ROM\_QEIVelocityEnable  
// Enables the velocity capture.



ROM\_QEIVelocityGet  
// Gets the current encoder speed.

ROM\_SSIConfigSetExpClk  
// Configures the synchronous serial interface.

ROM\_SSIDataGet  
// Gets a data element from the SSI receive FIFO.

ROM\_SSIDataGetNonBlocking  
// Gets a data element from the SSI receive FIFO.

ROM\_SSIDataPut  
// Puts a data element into the SSI transmit FIFO.

ROM\_SSIDataPutNonBlocking  
// Puts a data element into the SSI transmit FIFO.

ROM\_SSIDisable  
// Disables the synchronous serial interface.

ROM\_SSIDMADisable  
// Disable SSI DMA operation.

ROM\_SSIDMAEnable  
// Enable SSI DMA operation.

ROM\_SSIEnable  
// Enables the synchronous serial interface.

ROM\_SSIIntClear  
// Clears SSI interrupt sources.

ROM\_SSIIntDisable  
// Disables individual SSI interrupt sources.

ROM\_SSIIntEnable  
// Enables individual SSI interrupt sources.

ROM\_SSIIntStatus  
// Gets the current interrupt status.

ROM\_SysCtlADCSpeedGet  
// Gets the sample rate of the ADC.

ROM\_SysCtlADCSpeedSet  
// Sets the sample rate of the ADC.

ROM\_SysCtlClockGet  
// Gets the processor clock rate.

ROM\_SysCtlClockSet  
// Sets the clocking of the device.

ROM\_SysCtlDeepSleep  
// Puts the processor into deep-sleep mode.

ROM\_SysCtlFlashSizeGet  
// Gets the size of the flash.

ROM\_SysCtlGPIOAHBDisable  
// Disables a GPIO peripheral for access from the AHB.

ROM\_SysCtlGPIOAHBEnable  
// Enables a GPIO peripheral for access from the AHB.

ROM\_SysCtlIntClear  
// Clears system control interrupt sources.

ROM\_SysCtlIntDisable  
// Disables individual system control interrupt sources.

ROM\_SysCtlIntEnable  
// Enables individual system control interrupt sources.

ROM\_SysCtlIntStatus  
// Gets the current interrupt status.

ROM\_SysCtlLDOGet  
// Gets the output voltage of the LDO.

ROM\_SysCtlLDOSet  
// Sets the output voltage of the LDO.

ROM\_SysCtlPeripheralClockGating  
// Controls peripheral clock gating in sleep and deep-sleep mode.

ROM\_SysCtlPeripheralDeepSleepDisable  
// Disables a peripheral in deep-sleep mode.

ROM\_SysCtlPeripheralDeepSleepEnable  
// Enables a peripheral in deep-sleep mode.

ROM\_SysCtlPeripheralDisable  
// Disables a peripheral.

ROM\_SysCtlPeripheralEnable  
// Enables a peripheral.

ROM\_SysCtlPeripheralPresent  
// Determines if a peripheral is present.

ROM\_SysCtlPeripheralReset  
// Performs a software reset of a peripheral.

ROM\_SysCtlPeripheralSleepDisable  
// Disables a peripheral in sleep mode.

```
ROM_SysCtlPeripheralSleepEnable
    // Enables a peripheral in sleep mode.

ROM_SysCtlPinPresent
    // Determines if a pin is present.

ROM_SysCtlPWMClockGet
    // Gets the current PWM clock configuration.

ROM_SysCtlPWMClockSet
    // Sets the PWM clock configuration.

ROM_SysCtlReset
    // Resets the device.

ROM_SysCtlResetCauseClear
    // Clears reset reasons.

ROM_SysCtlResetCauseGet
    // Gets the reason for a reset.

ROM_SysCtlSleep
    // Puts the processor into sleep mode.

ROM_SysCtlSRAMSizeGet
    // Gets the size of the SRAM.

ROM_SysCtlUSBPLLDisable
    // Powers down the USB PLL.

ROM_SysCtlUSBPLLEnable
    // Powers up the USB PLL.

ROM_SysTickDisable
    // Disables the SysTick counter.

ROM_SysTickEnable
    // Enables the SysTick counter.

ROM_SysTickIntDisable
    // Disables the SysTick interrupt.

ROM_SysTickIntEnable
    // Enables the SysTick interrupt.

ROM_SysTickPeriodGet
    // Gets the period of the SysTick counter.

ROM_SysTickPeriodSet
    // Sets the period of the SysTick counter.

ROM_SysTickValueGet
    // Gets the current value of the SysTick counter.
```

ROM\_TimerConfigure  
// Configures the timer(s).

ROM\_TimerControlEvent  
// Controls the event type.

ROM\_TimerControlLevel  
// Controls the output level.

ROM\_TimerControlStall  
// Controls the stall handling.

ROM\_TimerControlTrigger  
// Enables or disables the trigger output.

ROM\_TimerDisable  
// Disables the timer(s).

ROM\_TimerEnable  
// Enables the timer(s).

ROM\_TimerIntClear  
// Clears timer interrupt sources.

ROM\_TimerIntDisable  
// Disables individual timer interrupt sources.

ROM\_TimerIntEnable  
// Enables individual timer interrupt sources.

ROM\_TimerIntStatus  
// Gets the current interrupt status.

ROM\_TimerLoadGet  
// Gets the timer load value.

ROM\_TimerLoadSet  
// Sets the timer load value.

ROM\_TimerMatchGet  
// Gets the timer match value.

ROM\_TimerMatchSet  
// Sets the timer match value.

ROM\_TimerPrescaleGet  
// Get the timer prescale value.

ROM\_TimerPrescaleSet  
// Set the timer prescale value.

ROM\_TimerRTCDisable  
// Disable RTC counting.

```
ROM_TimerRTCEnable
    // Enable RTC counting.

ROM_TimerValueGet
    // Gets the current timer value.

ROM_UARTBreakCtl
    // Causes a BREAK to be sent.

ROM_UARTCharGet
    // Waits for a character from the specified port.

ROM_UARTCharGetNonBlocking
    // Receives a character from the specified port.

ROM_UARTCharPut
    // Waits to send a character from the specified port.

ROM_UARTCharPutNonBlocking
    // Sends a character to the specified port.

ROM_UARTCharsAvail
    // Determines if there are any characters in the receive FIFO.

ROM_UARTConfigGetExpClk
    // Gets the current configuration of a UART.

ROM_UARTConfigSetExpClk
    // Sets the configuration of a UART.

ROM_UARTDisable
    // Disables transmitting and receiving.

ROM_UARTDisableSIR
    // Disables SIR (IrDA) mode on the specified UART.

ROM_UARTDMADisable
    // Disable UART DMA operation.

ROM_UARTDMAEnable
    // Enable UART DMA operation.

ROM_UARTEnable
    // Enables transmitting and receiving.

ROM_UARTEnableSIR
    // Enables SIR (IrDA) mode on specified UART.

ROM_UARTFIFOLevelGet
    // Gets the FIFO level at which interrupts are generated.

ROM_UARTFIFOLevelSet
    // Sets the FIFO level at which interrupts are generated.
```

ROM\_UARTIntClear  
// Clears UART interrupt sources.

ROM\_UARTIntDisable  
// Disables individual UART interrupt sources.

ROM\_UARTIntEnable  
// Enables individual UART interrupt sources.

ROM\_UARTIntStatus  
// Gets the current interrupt status.

ROM\_UARTParityModeGet  
// Gets the type of parity currently being used.

ROM\_UARTParityModeSet  
// Sets the type of parity.

ROM\_UARTSpaceAvail  
// Determines if there is any space in the transmit FIFO.

ROM\_uDMAChannelAttributeDisable  
// Disables attributes of a uDMA channel.

ROM\_uDMAChannelAttributeEnable  
// Enables attributes of a uDMA channel.

ROM\_uDMAChannelAttributeGet  
// Gets the enabled attributes of a uDMA channel.

ROM\_uDMAChannelControlSet  
// Sets the control parameters for a uDMA channel.

ROM\_uDMAChannelDisable  
// Disables a uDMA channel for operation.

ROM\_uDMAChannelEnable  
// Enables a uDMA channel for operation.

ROM\_uDMAChannelsEnabled  
// Checks if a uDMA channel is enabled for operation.

ROM\_uDMAChannelModeGet  
// Gets the transfer mode for a uDMA channel.

ROM\_uDMAChannelRequest  
// Requests a uDMA channel to start a transfer.

ROM\_uDMAChannelSizeGet  
// Gets the current transfer size for a uDMA channel.

ROM\_uDMAChannelTransferSet  
// Sets the transfer parameters for a uDMA channel.

```
ROM_uDMAControlBaseGet
    // Gets the base address for the channel control table.

ROM_uDMAControlBaseSet
    // Sets the base address for the channel control table.

ROM_uDMADisable
    // Disables the uDMA controller for use.

ROM_uDMAEnable
    // Enables the uDMA controller for use.

ROM_uDMAErrorStatusClear
    // Clears the uDMA error interrupt.

ROM_uDMAErrorStatusGet
    // Gets the uDMA error status.

ROM_UpdateI2C
    // Starts an update over the I2C0 interface.

ROM_UpdateSSI
    // Starts an update over the SSI0 interface.

ROM_UpdateUART
    // Starts an update over the UART0 interface.

ROM_USBDevAddrGet
    // Returns the current device address in device mode.

ROM_USBDevAddrSet
    // Sets the address in device mode.

ROM_USBDevConnect
    // Connects the USB controller to the bus in device mode.

ROM_USBDevDisconnect
    // Removes the USB controller from the bus in device mode.

ROM_USBDevEndpointConfig
    // Sets the configuration for an endpoint.

ROM_USBDevEndpointDataAck
    // Acknowledge that data was read from the given endpoint's FIFO in device mode.

ROM_USBDevEndpointStall
    // Stalls the specified endpoint in device mode.

ROM_USBDevEndpointStallClear
    // Clears the stall condition on the specified endpoint in device mode.

ROM_USBDevEndpointStatusClear
    // Clears the status bits in this endpoint in device mode.
```

ROM\_USBEndpointDataGet  
// Retrieves data from the given endpoint's FIFO.

ROM\_USBEndpointDataPut  
// Puts data into the given endpoint's FIFO.

ROM\_USBEndpointDataSend  
// Starts the transfer of data from an endpoint's FIFO.

ROM\_USBEndpointDataToggleClear  
// Sets the Data toggle on an endpoint to zero.

ROM\_USBEndpointStatus  
// Returns the current status of an endpoint.

ROM\_USBFIFOAddrGet  
// Returns the absolute FIFO address for a given endpoint.

ROM\_USBFIFOConfigGet  
// Returns the FIFO configuration for an endpoint.

ROM\_USBFIFOConfigSet  
// Sets the FIFO configuration for an endpoint.

ROM\_USBFIFOFlush  
// Forces a flush of an endpoint's FIFO.

ROM\_USBFrameNumberGet  
// Get the current frame number.

ROM\_USBHostAddrGet  
// Gets the current functional device address for an endpoint.

ROM\_USBHostAddrSet  
// Sets the functional address for the device that is connected to an endpoint in host mode.

ROM\_USBHostEndpointConfig  
// Sets the base configuration for a host endpoint.

ROM\_USBHostEndpointDataAck  
// Acknowledge that data was read from the given endpoint's FIFO in host mode.

ROM\_USBHostEndpointDataToggle  
// Sets the value data toggle on an endpoint in host mode.

ROM\_USBHostEndpointStatusClear  
// Clears the status bits in this endpoint in host mode.

ROM\_USBHostHubAddrGet  
// Get the current device hub address for this endpoint.

ROM\_USBHostHubAddrSet  
// Set the hub address for the device that is connected to an endpoint.



ROM\_USBHostPwrDisable  
// Disables the external power pin.

ROM\_USBHostPwrEnable  
// Enables the external power pin.

ROM\_USBHostPwrFaultConfig  
// Sets the configuration for USB power fault.

ROM\_USBHostPwrFaultDisable  
// Disables power fault detection.

ROM\_USBHostPwrFaultEnable  
// Enables power fault detection.

ROM\_USBHostRequestIN  
// Schedules a request for an IN transaction on an endpoint in host mode.

ROM\_USBHostRequestStatus  
// Issues a request for a status IN transaction on endpoint zero.

ROM\_USBHostReset  
// Handles the USB bus reset condition.

ROM\_USBHostResume  
// Handles the USB bus resume condition.

ROM\_USBHostSpeedGet  
// Returns the current speed of the USB device connected.

ROM\_USBHostSuspend  
// Puts the USB bus in a suspended state.

ROM\_USBIntDisable  
// Disables the sources for USB interrupts.

ROM\_USBIntEnable  
// Enables the sources for USB interrupts.

ROM\_USBIntStatus  
// Returns the status of the USB interrupts.

ROM\_WatchdogEnable  
// Enables the watchdog timer.

ROM\_WatchdogIntClear  
// Clears the watchdog timer interrupt.

ROM\_WatchdogIntEnable  
// Enables the watchdog timer interrupt.

ROM\_WatchdogIntStatus  
// Gets the current watchdog timer interrupt status.

ROM\_WatchdogLock  
// Enables the watchdog timer lock mechanism.

ROM\_WatchdogLockState  
// Gets the state of the watchdog timer lock mechanism.

ROM\_WatchdogReloadGet  
// Gets the watchdog timer reload value.

ROM\_WatchdogReloadSet  
// Sets the watchdog timer reload value.

ROM\_WatchdogResetDisable  
// Disables the watchdog timer reset.

ROM\_WatchdogResetEnable  
// Enables the watchdog timer reset.

ROM\_WatchdogRunning  
// Determines if the watchdog timer is enabled.

ROM\_WatchdogStallDisable  
// Disables stalling of the watchdog timer during debug events.

ROM\_WatchdogStallEnable  
// Enables stalling of the watchdog timer during debug events.

ROM\_WatchdogUnlock  
// Disables the watchdog timer lock mechanism.

ROM\_WatchdogValueGet  
// Gets the current watchdog timer value.

## C Advance Encryption Standard and Cyclic Redundancy Check Software in ROM

AES and CRC software is available in the integrated ROM of the LM3S5791 microcontroller at 0x0100.5000. For more information on this software, see *Stellaris® ROM User's Guide*.

### C.1 Advanced Encryption Standard Software

The Advanced Encryption Standard (AES) is a publicly defined encryption standard used by the U.S. Government. It is a strong encryption method with reasonable performance and size. AES is fast in both hardware and software, is fairly easy to implement, and requires little memory. AES is ideal for applications that can use pre-arranged keys, such as setup during manufacturing or configuration.

### C.2 Cyclic Redundancy Check Software

CRC (Cyclic Redundancy Check) is a technique to validate a span of data has the same contents as when previously checked. This technique can be used to validate correct receipt of messages (nothing lost or modified in transit), to validate data after decompression, to validate that Flash memory contents have not been changed, and for other cases where the data needs to be validated. A CRC is preferred over a simple checksum (e.g. XOR all bits) because it catches changes more readily.

## D Register Quick Reference

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
<b>System Control</b>																															
Base 0x400F.E000																															
<b>DID0, type RO, offset 0x000, reset -</b>																															
VER								CLASS																							
MAJOR								MINOR																							
<b>PBORCTL, type R/W, offset 0x030, reset 0x0000.7FFD</b>																															
														EOPOR																	
<b>RIS, type RO, offset 0x050, reset 0x0000.0000</b>																															
														EORIS																	
<b>IMC, type R/W, offset 0x054, reset 0x0000.0000</b>																															
														EORIM																	
<b>MISC, type R/W1C, offset 0x058, reset 0x0000.0000</b>																															
														EORMS																	
<b>RESC, type R/W, offset 0x05C, reset -</b>																															
														MSCRA																	
														reservedWDT1		SW	WDT0	BOR	POR	EXT											
<b>RCC, type R/W, offset 0x060, reset 0x078E.3AD1</b>																															
				ACG				SYSDIV				LSESELV		LSESELV		PWMDIV															
PWFDN				BYPASS				XTAL				OSCSRC				CBCDS MSCDS															
<b>PLLCFG, type RO, offset 0x064, reset -</b>																															
F														R																	
<b>GPIOHCTL, type R/W, offset 0x06C, reset 0x0000.0000</b>																															
														FORIJ		FORIH		FORIG		FORIF		FORIE		FORID		FORIC		FORIB		FORIA	
<b>RCC2, type R/W, offset 0x070, reset 0x0780.6810</b>																															
LSERC2		LSEACT		SYSDIV2				FRFCT																							
LSEVDN		PWFD2		BYPAS2				OSCSRC2																							
<b>MOSCCCTL, type R/W, offset 0x07C, reset 0x0000.0000</b>																															
														CVAL																	
<b>DSLCLKCFG, type R/W, offset 0x144, reset 0x0780.0000</b>																															
DSDIVORIDE														DSOSCSRC																	
<b>DSFLASHCFG, type R/W, offset 0x14C, reset 0x0000.0000</b>																															
														SDW																	
<b>PIOCCAL, type R/W, offset 0x150, reset 0x0000.0000</b>																															
UTEN														UT																	
														UPDME																	
<b>I2SMCLKCFG, type R/W, offset 0x170, reset 0x0000.0000</b>																															
RXEN				RXI				RXF																							
TXEN				TXI				TXF																							
<b>DID1, type RO, offset 0x004, reset -</b>																															
VER				FAM				PARTNO																							
PINCOUNT				TEMP				PKG		ROHS		QUAL																			

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>DC0, type RO, offset 0x008, reset 0x00FF.003F</b>																
SRAMSZ																
FLASHSZ																
<b>DC1, type RO, offset 0x010, reset -</b>																
			WDT1			CAN1	CAN0				PWM			ADC1	ADC0	
			MINSYSDIV		MAXADC1SPD	MAXADC0SPD	MPU		EMUS	PLL	WDT0	SWO	SWD	JTAG		
<b>DC2, type RO, offset 0x014, reset 0x570F.5337</b>																
	EPI0		I2S0		COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0	
	I2C1		I2C0			QE1	QE10			SSI1	SSI0		UART2	UART1	UART0	
<b>DC3, type RO, offset 0x018, reset 0xBFFF.B6FF</b>																
32KHZ		CCP5	CCP4	CCP3	CCP2	CCP1	CCP0	ADCM7	ADCM6	ADCM5	ADCM4	ADCM3	ADCM2	ADCM1	ADCM0	
PWM1		CPLS	CMUS		CPLS	CMUS		CPLS	CMUS	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0	
<b>DC4, type RO, offset 0x01C, reset 0x0000.F1FF</b>																
	CCP7	CCP6	UDMA	ROM				GPIU	GPIH	GPIG	GPIF	GPIE	GPID	GPIC	GPIB	GPIA
<b>DC5, type RO, offset 0x020, reset 0x0F30.00FF</b>																
					PWM10	PWM9	PWM8	PWM7			PWM3	PWM2				
								PWM7	PWM6	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0	
<b>DC6, type RO, offset 0x024, reset 0x0000.0013</b>																
											USBPY				USB0	
<b>DC7, type RO, offset 0x028, reset 0xFFFF.FFFF</b>																
reserved	DAC9	DAC8	DAC7	DAC6	DAC5	DAC4	DAC3	DAC2	DAC1	DAC0	DAC9	DAC8	DAC7	DAC6	DAC5	
	DAC4	DAC3	DAC2	DAC1	DAC0	DAC9	DAC8	DAC7	DAC6	DAC5	DAC4	DAC3	DAC2	DAC1	DAC0	
<b>DC8, type RO, offset 0x02C, reset 0xFFFF.FFFF</b>																
ADCM5	ADCM4	ADCM3	ADCM2	ADCM1	ADCM0	ADCM9	ADCM8	ADCM7	ADCM6	ADCM5	ADCM4	ADCM3	ADCM2	ADCM1	ADCM0	
ADCM5	ADCM4	ADCM3	ADCM2	ADCM1	ADCM0	ADCM9	ADCM8	ADCM7	ADCM6	ADCM5	ADCM4	ADCM3	ADCM2	ADCM1	ADCM0	
<b>DC9, type RO, offset 0x190, reset 0x00FF.00FF</b>																
								ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	
								ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	
<b>NVMSTAT, type RO, offset 0x1A0, reset 0x0000.0001</b>																
															FWB	
<b>RCGC0, type R/W, offset 0x100, reset 0x00000040</b>																
			WDT1			CAN1	CAN0				PWM			ADC1	ADC0	
					MAXADC1SPD	MAXADC0SPD						WDT0				
<b>SCGC0, type R/W, offset 0x110, reset 0x00000040</b>																
			WDT1			CAN1	CAN0				PWM			ADC1	ADC0	
					MAXADC1SPD	MAXADC0SPD						WDT0				
<b>DCGC0, type R/W, offset 0x120, reset 0x00000040</b>																
			WDT1			CAN1	CAN0				PWM			ADC1	ADC0	
					MAXADC1SPD	MAXADC0SPD						WDT0				
<b>RCGC1, type R/W, offset 0x104, reset 0x00000000</b>																
	EPI0		I2S0		COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0	
	I2C1		I2C0			QE1	QE10			SSI1	SSI0		UART2	UART1	UART0	
<b>SCGC1, type R/W, offset 0x114, reset 0x00000000</b>																
	EPI0		I2S0		COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0	
	I2C1		I2C0			QE1	QE10			SSI1	SSI0		UART2	UART1	UART0	
<b>DCGC1, type R/W, offset 0x124, reset 0x00000000</b>																
	EPI0		I2S0		COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0	
	I2C1		I2C0			QE1	QE10			SSI1	SSI0		UART2	UART1	UART0	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
<b>RCGC2, type R/W, offset 0x108, reset 0x00000000</b>																
															USB0	
			UDMA					GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
<b>SCGC2, type R/W, offset 0x118, reset 0x00000000</b>																
															USB0	
			UDMA					GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
<b>DCGC2, type R/W, offset 0x128, reset 0x00000000</b>																
															USB0	
			UDMA					GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
<b>SRCR0, type R/W, offset 0x040, reset 0x00000000</b>																
				WDT1			CAN1	CAN0				PWM			ADC1	ADC0
													WDT0			
<b>SRCR1, type R/W, offset 0x044, reset 0x00000000</b>																
		EPI0		I2S0		COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
		I2C1		I2C0			QE1	QE10			SSI1	SSI0		UART2	UART1	UART0
<b>SRCR2, type R/W, offset 0x048, reset 0x00000000</b>																
																USB0
			UDMA					GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
<b>Internal Memory</b>																
<b>Flash Registers (Flash Control Offset)</b>																
Base 0x400F.D000																
<b>FMA, type R/W, offset 0x000, reset 0x0000.0000</b>																
																OFFSET
OFFSET																
<b>FMD, type R/W, offset 0x004, reset 0x0000.0000</b>																
DATA																
DATA																
<b>FMC, type R/W, offset 0x008, reset 0x0000.0000</b>																
WRKEY																
												COMT	MERGE	ERASE	WRITE	
<b>FCRIS, type RO, offset 0x00C, reset 0x0000.0000</b>																
															PRIS	ARIS
<b>FCIM, type R/W, offset 0x010, reset 0x0000.0000</b>																
															R/WSK	A/WSK
<b>FCMISC, type R/W1C, offset 0x014, reset 0x0000.0000</b>																
															FMSC	AMSC
<b>FMC2, type R/W, offset 0x020, reset 0x0000.0000</b>																
WRKEY																
																WRBUF
<b>FWBVAL, type R/W, offset 0x030, reset 0x0000.0000</b>																
FWB[n]																
FWB[n]																
<b>FWBn, type R/W, offset 0x100 - 0x13C, reset 0x0000.0000</b>																
DATA																
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>Internal Memory</b>															
<b>Memory Registers (System Control Offset)</b>															
Base 0x400F.E000															
<b>RMCTL, type R/W1C, offset 0x0F0, reset -</b>															
															BA
<b>RMVER, type RO, offset 0x0F4, reset 0x0202.5400</b>															
CONT								SIZE							
VER								REV							
<b>FMPRE0, type R/W, offset 0x130 and 0x200, reset 0xFFFF.FFFF</b>															
READ_ENABLE								READ_ENABLE							
<b>FMPPE0, type R/W, offset 0x134 and 0x400, reset 0xFFFF.FFFF</b>															
PROG_ENABLE								PROG_ENABLE							
<b>USER_DBG, type R/W, offset 0x1D0, reset 0xFFFF.FFFE</b>															
NW	DATA														
	DATA												DBG1	DBG0	
<b>USER_REG0, type R/W, offset 0x1E0, reset 0xFFFF.FFFF</b>															
NW	DATA														
	DATA														
<b>USER_REG1, type R/W, offset 0x1E4, reset 0xFFFF.FFFF</b>															
NW	DATA														
	DATA														
<b>USER_REG2, type R/W, offset 0x1E8, reset 0xFFFF.FFFF</b>															
NW	DATA														
	DATA														
<b>USER_REG3, type R/W, offset 0x1EC, reset 0xFFFF.FFFF</b>															
NW	DATA														
	DATA														
<b>FMPRE1, type R/W, offset 0x204, reset 0xFFFF.FFFF</b>															
READ_ENABLE								READ_ENABLE							
<b>FMPRE2, type R/W, offset 0x208, reset 0x0000.0000</b>															
READ_ENABLE								READ_ENABLE							
<b>FMPRE3, type R/W, offset 0x20C, reset 0x0000.0000</b>															
READ_ENABLE								READ_ENABLE							
<b>FMPPE1, type R/W, offset 0x404, reset 0xFFFF.FFFF</b>															
PROG_ENABLE								PROG_ENABLE							
<b>FMPPE2, type R/W, offset 0x408, reset 0x0000.0000</b>															
PROG_ENABLE								PROG_ENABLE							
<b>FMPPE3, type R/W, offset 0x40C, reset 0x0000.0000</b>															
PROG_ENABLE								PROG_ENABLE							

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>Micro Direct Memory Access (μDMA)</b>																			
<b>μDMA Channel Control Structure</b>																			
Base n/a																			
<b>DMASRCENDP, type R/W, offset 0x000, reset -</b>																			
ADDR																			
ADDR																			
<b>DMADSTENDP, type R/W, offset 0x004, reset -</b>																			
ADDR																			
ADDR																			
<b>DMACHCTL, type R/W, offset 0x008, reset -</b>																			
DSTINC				DSTSIZE				SRCINC				SRCSIZE				ARBSIZE			
ARBSIZE				XFERSIZE				XFERMODE											
<b>Micro Direct Memory Access (μDMA)</b>																			
<b>μDMA Registers</b>																			
Base 0x400F.F000																			
<b>DMASTAT, type RO, offset 0x000, reset 0x001F.0000</b>																			
STATE																			
DMACHANS																			
MSEN																			
<b>DMACFG, type WO, offset 0x004, reset -</b>																			
MSEN																			
<b>DMACTLBASE, type R/W, offset 0x008, reset 0x0000.0000</b>																			
ADDR																			
ADDR																			
<b>DMAALTBASE, type RO, offset 0x00C, reset 0x0000.0200</b>																			
ADDR																			
ADDR																			
<b>DMAWAITSTAT, type RO, offset 0x010, reset 0x0000.0000</b>																			
WAITREQ[n]																			
WAITREQ[n]																			
<b>DMASWREQ, type WO, offset 0x014, reset -</b>																			
SWREQ[n]																			
SWREQ[n]																			
<b>DMAUSEBURSTSET, type RO, offset 0x018, reset 0x0000.0000 (Reads)</b>																			
SET[n]																			
SET[n]																			
<b>DMAUSEBURSTSET, type WO, offset 0x018, reset 0x0000.0000 (Writes)</b>																			
SET[n]																			
SET[n]																			
<b>DMAUSEBURSTCLR, type WO, offset 0x01C, reset -</b>																			
CLR[n]																			
CLR[n]																			
<b>DMAREQMASKSET, type RO, offset 0x020, reset 0x0000.0000 (Reads)</b>																			
SET[n]																			
SET[n]																			
<b>DMAREQMASKSET, type WO, offset 0x020, reset 0x0000.0000 (Writes)</b>																			
SET[n]																			
SET[n]																			
<b>DMAREQMASKCLR, type WO, offset 0x024, reset -</b>																			
CLR[n]																			
CLR[n]																			



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DMAENASET, type RO, offset 0x028, reset 0x0000.0000 (Reads)</b>															
SET[n]															
SET[n]															
<b>DMAENASET, type WO, offset 0x028, reset 0x0000.0000 (Writes)</b>															
SET[n]															
SET[n]															
<b>DMAENACL, type WO, offset 0x02C, reset -</b>															
CLR[n]															
CLR[n]															
<b>DMAALTSET, type RO, offset 0x030, reset 0x0000.0000 (Reads)</b>															
SET[n]															
SET[n]															
<b>DMAALTSET, type WO, offset 0x030, reset 0x0000.0000 (Writes)</b>															
SET[n]															
SET[n]															
<b>DMAALTCLR, type WO, offset 0x034, reset -</b>															
CLR[n]															
CLR[n]															
<b>DMAPRIOSET, type RO, offset 0x038, reset 0x0000.0000 (Reads)</b>															
SET[n]															
SET[n]															
<b>DMAPRIOSET, type WO, offset 0x038, reset 0x0000.0000 (Writes)</b>															
SET[n]															
SET[n]															
<b>DMAPRIOCLR, type WO, offset 0x03C, reset -</b>															
CLR[n]															
CLR[n]															
<b>DMAERRCLR, type RO, offset 0x04C, reset 0x0000.0000 (Reads)</b>															
															ERRR
<b>DMAERRCLR, type WO, offset 0x04C, reset 0x0000.0000 (Writes)</b>															
															ERRR
<b>DMACHALT, type R/W, offset 0x500, reset 0x0000.0000</b>															
CHALT[n]															
CHALT[n]															
<b>DMACHIS, type R/W1C, offset 0x504, reset 0x0000.0000</b>															
CHIS[n]															
CHIS[n]															
<b>DMAPeriphID0, type RO, offset 0xFE0, reset 0x0000.0030</b>															
															PID0
<b>DMAPeriphID1, type RO, offset 0xFE4, reset 0x0000.00B2</b>															
															PID1
<b>DMAPeriphID2, type RO, offset 0xFE8, reset 0x0000.000B</b>															
															PID2
<b>DMAPeriphID3, type RO, offset 0xFEC, reset 0x0000.0000</b>															
															PID3

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DMAPeriphID4, type RO, offset 0xFD0, reset 0x0000.0004</b>															
PID4															
<b>DMAPCellID0, type RO, offset 0xFF0, reset 0x0000.000D</b>															
CID0															
<b>DMAPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0</b>															
CID1															
<b>DMAPCellID2, type RO, offset 0xFF8, reset 0x0000.0005</b>															
CID2															
<b>DMAPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1</b>															
CID3															
<b>General-Purpose Input/Outputs (GPIOs)</b>															
GPIO Port A (APB) base: 0x4000.4000															
GPIO Port A (AHB) base: 0x4005.8000															
GPIO Port B (APB) base: 0x4000.5000															
GPIO Port B (AHB) base: 0x4005.9000															
GPIO Port C (APB) base: 0x4000.6000															
GPIO Port C (AHB) base: 0x4005.A000															
GPIO Port D (APB) base: 0x4000.7000															
GPIO Port D (AHB) base: 0x4005.B000															
GPIO Port E (APB) base: 0x4002.4000															
GPIO Port E (AHB) base: 0x4005.C000															
GPIO Port F (APB) base: 0x4002.5000															
GPIO Port F (AHB) base: 0x4005.D000															
GPIO Port G (APB) base: 0x4002.6000															
GPIO Port G (AHB) base: 0x4005.E000															
GPIO Port H (APB) base: 0x4002.7000															
GPIO Port H (AHB) base: 0x4005.F000															
GPIO Port J (APB) base: 0x4003.D000															
GPIO Port J (AHB) base: 0x4006.0000															
<b>GPIO_DATA, type R/W, offset 0x000, reset 0x0000.0000</b>															
DATA															
<b>GPIO_DIR, type R/W, offset 0x400, reset 0x0000.0000</b>															
DIR															
<b>GPIO_IS, type R/W, offset 0x404, reset 0x0000.0000</b>															
IS															
<b>GPIO_IBE, type R/W, offset 0x408, reset 0x0000.0000</b>															
IBE															
<b>GPIO_IEV, type R/W, offset 0x40C, reset 0x0000.0000</b>															
IEV															
<b>GPIO_IM, type R/W, offset 0x410, reset 0x0000.0000</b>															
IME															
<b>GPIO_RIS, type RO, offset 0x414, reset 0x0000.0000</b>															
RIS															
<b>GPIO_MIS, type RO, offset 0x418, reset 0x0000.0000</b>															
MIS															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<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>GIPIODEN, type R/W, offset 0x51C, reset -</b>															
DEN															
<b>GPIOLOCK, type R/W, offset 0x520, reset 0x0000.0001</b>															
LOCK															
LOCK															
<b>GPIOCR, type -, offset 0x524, reset -</b>															
CR															
<b>GPIOAMSEL, type R/W, offset 0x528, reset 0x0000.0000</b>															
GPIOAMSEL															
<b>GPIOPCTL, type R/W, offset 0x52C, reset -</b>															
PMC7				PMC6				PMC5				PMC4			
PMC3				PMC2				PMC1				PMC0			
<b>GPIOPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000</b>															
PID4															
<b>GPIOPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000</b>															
PID5															
<b>GPIOPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000</b>															
PID6															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
GPIOPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000																			
												PID7							
GPIOPeriphID0, type RO, offset 0xFE0, reset 0x0000.0061																			
												PID0							
GPIOPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000																			
												PID1							
GPIOPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018																			
												PID2							
GPIOPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001																			
												PID3							
GPIOCellID0, type RO, offset 0xFF0, reset 0x0000.000D																			
												CID0							
GPIOCellID1, type RO, offset 0xFF4, reset 0x0000.00F0																			
												CID1							
GPIOCellID2, type RO, offset 0xFF8, reset 0x0000.0005																			
												CID2							
GPIOCellID3, type RO, offset 0xFFC, reset 0x0000.00B1																			
												CID3							
<b>External Peripheral Interface (EPI)</b>																			
Base 0x400D.0000																			
EPICFG, type R/W, offset 0x000, reset 0x0000.0000																			
												ELKEN		MODE					
EPIBAUD, type R/W, offset 0x004, reset 0x0000.0000																			
												COUNT							
EPIDRAMCFG, type R/W, offset 0x010, reset 0x42EE.0000																			
FREQ				SLEEP				RFSH				SIZE							
EPIHB8CFG, type R/W, offset 0x010, reset 0x0000.FF00																			
MAXWAIT				XFFEN		XFEEN		WRGH		RDGH		MODE							
EPIGPCFG, type R/W, offset 0x010, reset 0x0000.FF00																			
QLEN		QGE		FDEN		FRMN		FRMD		FRMCNT		RW		WDC		RDC			
MAXWAIT								ASIZE				DSIZE							
EPIHB8CFG2, type R/W, offset 0x014, reset 0x0000.0000																			
WORD								CSCFGreserved											
EPIGPCFG2, type R/W, offset 0x014, reset 0x0000.0000																			
WORD																			
EPIADDRMAP, type R/W, offset 0x01C, reset 0x0000.0000																			
				EPSZ				EPADR				ERSZ				ERADR			



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>EPIWFIFOCNT, type R, offset 0x204, reset 0x0000.0000</b>																					
<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:40%;"></td> <td style="width:40%;"></td> <td style="width:20%;"></td> </tr> <tr> <td></td><td></td><td style="text-align: right;">WTAV</td> </tr> </table>																					WTAV
		WTAV																			
<b>EPIIM, type R/W, offset 0x210, reset 0x0000.0000</b>																					
<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:40%;"></td> <td style="width:40%;"></td> <td style="width:20%;"></td> </tr> <tr> <td></td><td></td><td style="text-align: right;">WRIM RDIM ERRM</td> </tr> </table>																					WRIM RDIM ERRM
		WRIM RDIM ERRM																			
<b>EPIRIS, type R, offset 0x214, reset 0x0000.0000</b>																					
<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:40%;"></td> <td style="width:40%;"></td> <td style="width:20%;"></td> </tr> <tr> <td></td><td></td><td style="text-align: right;">WRFS RDRS ERRS</td> </tr> </table>																					WRFS RDRS ERRS
		WRFS RDRS ERRS																			
<b>EPIMIS, type R, offset 0x218, reset 0x0000.0000</b>																					
<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:40%;"></td> <td style="width:40%;"></td> <td style="width:20%;"></td> </tr> <tr> <td></td><td></td><td style="text-align: right;">WRMS RDMS ERRMS</td> </tr> </table>																					WRMS RDMS ERRMS
		WRMS RDMS ERRMS																			
<b>EPIEISC, type R/W1C, offset 0x21C, reset 0x0000.0000</b>																					
<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:40%;"></td> <td style="width:40%;"></td> <td style="width:20%;"></td> </tr> <tr> <td></td><td></td><td style="text-align: right;">WRUL RSPAL TOUT</td> </tr> </table>																					WRUL RSPAL TOUT
		WRUL RSPAL TOUT																			
<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>																					
<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:40%;"></td> <td style="width:40%;"></td> <td style="width:20%;"></td> </tr> <tr> <td></td><td></td><td style="text-align: right;">GPTMCFG</td> </tr> </table>																					GPTMCFG
		GPTMCFG																			
<b>GPTMTAMR, type R/W, offset 0x004, reset 0x0000.0000</b>																					
<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:40%;"></td> <td style="width:40%;"></td> <td style="width:20%;"></td> </tr> <tr> <td></td><td></td><td style="text-align: right;">TAMS TAOT TAME TACR TAVS TAVR TAMR</td> </tr> </table>																					TAMS TAOT TAME TACR TAVS TAVR TAMR
		TAMS TAOT TAME TACR TAVS TAVR TAMR																			
<b>GPTMTBMR, type R/W, offset 0x008, reset 0x0000.0000</b>																					
<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:40%;"></td> <td style="width:40%;"></td> <td style="width:20%;"></td> </tr> <tr> <td></td><td></td><td style="text-align: right;">TBMS TBOT TBME TBCR TBVS TBVR TBMR</td> </tr> </table>																					TBMS TBOT TBME TBCR TBVS TBVR TBMR
		TBMS TBOT TBME TBCR TBVS TBVR TBMR																			
<b>GPTMCTL, type R/W, offset 0x00C, reset 0x0000.0000</b>																					
<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:40%;"></td> <td style="width:40%;"></td> <td style="width:20%;"></td> </tr> <tr> <td></td><td></td><td style="text-align: right;">TBWL TBOE TBEVENT TBSPL TBEN TBWL TAOTE RICE TAEVENT TASP TAEN</td> </tr> </table>																					TBWL TBOE TBEVENT TBSPL TBEN TBWL TAOTE RICE TAEVENT TASP TAEN
		TBWL TBOE TBEVENT TBSPL TBEN TBWL TAOTE RICE TAEVENT TASP TAEN																			
<b>GPTMIMR, type R/W, offset 0x018, reset 0x0000.0000</b>																					
<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:40%;"></td> <td style="width:40%;"></td> <td style="width:20%;"></td> </tr> <tr> <td></td><td></td><td style="text-align: right;">TBMM CBEM CBMM TBMM TAVM RTOM CAEM CAMM TAVM</td> </tr> </table>																					TBMM CBEM CBMM TBMM TAVM RTOM CAEM CAMM TAVM
		TBMM CBEM CBMM TBMM TAVM RTOM CAEM CAMM TAVM																			
<b>GPTMRIS, type RO, offset 0x01C, reset 0x0000.0000</b>																					
<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:40%;"></td> <td style="width:40%;"></td> <td style="width:20%;"></td> </tr> <tr> <td></td><td></td><td style="text-align: right;">TBMS CBFS CBMS TBMS TAVS RCRS CAFS CAMS TAVS</td> </tr> </table>																					TBMS CBFS CBMS TBMS TAVS RCRS CAFS CAMS TAVS
		TBMS CBFS CBMS TBMS TAVS RCRS CAFS CAMS TAVS																			
<b>GPTMMIS, type RO, offset 0x020, reset 0x0000.0000</b>																					
<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:40%;"></td> <td style="width:40%;"></td> <td style="width:20%;"></td> </tr> <tr> <td></td><td></td><td style="text-align: right;">TBMS CBMS CBMS TBMS TAVS RCRS CAFS CAMS TAVS</td> </tr> </table>																					TBMS CBMS CBMS TBMS TAVS RCRS CAFS CAMS TAVS
		TBMS CBMS CBMS TBMS TAVS RCRS CAFS CAMS TAVS																			
<b>GPTMICR, type W1C, offset 0x024, reset 0x0000.0000</b>																					
<table border="1" style="width:100%; border-collapse: collapse;"> <tr> <td style="width:40%;"></td> <td style="width:40%;"></td> <td style="width:20%;"></td> </tr> <tr> <td></td><td></td><td style="text-align: right;">TBMT CBMT CBMT TBMT TAVM RCRM CAEM CAMM TAVM</td> </tr> </table>																					TBMT CBMT CBMT TBMT TAVM RCRM CAEM CAMM TAVM
		TBMT CBMT CBMT TBMT TAVM RCRM CAEM CAMM TAVM																			
<b>GPTMTAILR, type R/W, offset 0x028, reset 0xFFFF.FFFF</b>																					
TAILRH																					
TAILRL																					
<b>GPTMTBILR, type R/W, offset 0x02C, reset 0x0000.FFFF</b>																					
TBILRL																					
<b>GPTMTAMATCHR, type R/W, offset 0x030, reset 0xFFFF.FFFF</b>																					
TAMRH																					
TAMRL																					

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>GPTMTBMATCHR, type R/W, offset 0x034, reset 0x0000.FFFF</b>															
TBMR															
<b>GPTMTAPR, type R/W, offset 0x038, reset 0x0000.0000</b>															
TAPSR															
<b>GPTMTBPR, type R/W, offset 0x03C, reset 0x0000.0000</b>															
TBPSR															
<b>GPTMTAR, type RO, offset 0x048, reset 0xFFFF.FFFF</b>															
TARH															
TARL															
<b>GPTMTBR, type RO, offset 0x04C, reset 0x0000.FFFF</b>															
TBRL															
<b>GPTMTAV, type RO, offset 0x050, reset 0xFFFF.FFFF</b>															
TAVH															
TAVL															
<b>GPTMTBV, type RO, offset 0x054, reset 0x0000.FFFF</b>															
TBVL															
<b>Watchdog Timer</b>															
WDT0 base: 0x4000.0000															
WDT1 base: 0x4000.1000															
<b>WDTLOAD, type R/W, offset 0x000, reset 0xFFFF.FFFF</b>															
WDTLOAD															
WDTLOAD															
<b>WDTVALUE, type RO, offset 0x004, reset 0xFFFF.FFFF</b>															
WDTVALUE															
WDTVALUE															
<b>WDTCTL, type R/W, offset 0x008, reset 0x0000.0000 for WDT0, 0x8000.0000 for WDT1</b>															
WRC															
RESEN INTEN															
<b>WDTICR, type WO, offset 0x00C, reset -</b>															
WDTINTCLR															
WDTINTCLR															
<b>WDTRIS, type RO, offset 0x010, reset 0x0000.0000</b>															
WDTRIS															
<b>WDTMIS, type RO, offset 0x014, reset 0x0000.0000</b>															
WDTMIS															
<b>WDTTEST, type R/W, offset 0x418, reset 0x0000.0000</b>															
STALL															
<b>WDTLOCK, type R/W, offset 0xC00, reset 0x0000.0000</b>															
WDTLOCK															
WDTLOCK															
<b>WDTPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000</b>															
PID4															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>WDTPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000</b>															
												PID5			
<b>WDTPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000</b>															
												PID6			
<b>WDTPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000</b>															
												PID7			
<b>WDTPeriphID0, type RO, offset 0xFE0, reset 0x0000.0005</b>															
												PID0			
<b>WDTPeriphID1, type RO, offset 0xFE4, reset 0x0000.0018</b>															
												PID1			
<b>WDTPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018</b>															
												PID2			
<b>WDTPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001</b>															
												PID3			
<b>WDTPCellID0, type RO, offset 0xFF0, reset 0x0000.000D</b>															
												CID0			
<b>WDTPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0</b>															
												CID1			
<b>WDTPCellID2, type RO, offset 0xFF8, reset 0x0000.0006</b>															
												CID2			
<b>WDTPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1</b>															
												CID3			
<b>Analog-to-Digital Converter (ADC)</b>															
ADC0 base: 0x4003.8000															
ADC1 base: 0x4003.9000															
<b>ADCACTSS, type R/W, offset 0x000, reset 0x0000.0000</b>															
												ASEN3	ASEN2	ASEN1	ASEN0
<b>ADCRIS, type RO, offset 0x004, reset 0x0000.0000</b>															
												INR3	INR2	INR1	INR0
<b>ADCIM, type R/W, offset 0x008, reset 0x0000.0000</b>															
												IN3	IN2	IN1	IN0
<b>ADCISC, type R/W1C, offset 0x00C, reset 0x0000.0000</b>															
												OV3	OV2	OV1	OV0
<b>ADCSTAT, type R/W1C, offset 0x010, reset 0x0000.0000</b>															
												OV3	OV2	OV1	OV0



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>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>															
GSMC				SMA								SS3 SS2 SS1 SS0			
<b>ADCSAC, type R/W, offset 0x030, reset 0x0000.0000</b>															
												AVG			
<b>ADDCISC, type R/W1C, offset 0x034, reset 0x0000.0000</b>															
								DCN7	DCN6	DCN5	DCN4	DCN3	DCN2	DCN1	DCN0
<b>ADCCTL, type R/W, offset 0x038, reset 0x0000.0000</b>															
												VREF			
<b>ADCSSMUX0, type R/W, offset 0x040, reset 0x0000.0000</b>															
MUX7				MUX6				MUX5				MUX4			
MUX3				MUX2				MUX1				MUX0			
<b>ADCSSCTL0, type R/W, offset 0x044, reset 0x0000.0000</b>															
TS7	IE7	END7	D7	TS6	IE6	END6	D6	TS5	IE5	END5	D5	TS4	IE4	END4	D4
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
<b>ADCSSFIFO0, type RO, offset 0x048, reset 0x0000.0000</b>															
												DATA			
<b>ADCSSFIFO1, type RO, offset 0x068, reset 0x0000.0000</b>															
												DATA			
<b>ADCSSFIFO2, type RO, offset 0x088, reset 0x0000.0000</b>															
												DATA			
<b>ADCSSFIFO3, type RO, offset 0x0A8, reset 0x0000.0000</b>															
												DATA			
<b>ADCSSFSTAT0, type RO, offset 0x04C, reset 0x0000.0100</b>															
FULL				EMPTY				HPTR				TPTR			
<b>ADCSSFSTAT1, type RO, offset 0x06C, reset 0x0000.0100</b>															
FULL				EMPTY				HPTR				TPTR			
<b>ADCSSFSTAT2, type RO, offset 0x08C, reset 0x0000.0100</b>															
FULL				EMPTY				HPTR				TPTR			
<b>ADCSSFSTAT3, type RO, offset 0x0AC, reset 0x0000.0100</b>															
FULL				EMPTY				HPTR				TPTR			

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>ADCSSOP0, type R/W, offset 0x050, reset 0x0000.0000</b>																							
			STOP				STOP				STOP				STOP								
			STOP				STOP				STOP				STOP								
<b>ADCSSDC0, type R/W, offset 0x054, reset 0x0000.0000</b>																							
S7DCSEL				S6DCSEL				S5DCSEL				S4DCSEL											
S3DCSEL				S2DCSEL				S1DCSEL				S0DCSEL											
<b>ADCSSMUX1, type R/W, offset 0x060, reset 0x0000.0000</b>																							
MUX3				MUX2				MUX1				MUX0											
<b>ADCSSMUX2, type R/W, offset 0x080, reset 0x0000.0000</b>																							
MUX3				MUX2				MUX1				MUX0											
<b>ADCSSCTL1, type R/W, offset 0x064, reset 0x0000.0000</b>																							
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0								
<b>ADCSSCTL2, type R/W, offset 0x084, reset 0x0000.0000</b>																							
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0								
<b>ADCSSOP1, type R/W, offset 0x070, reset 0x0000.0000</b>																							
			STOP				STOP				STOP				STOP								
<b>ADCSSOP2, type R/W, offset 0x090, reset 0x0000.0000</b>																							
			STOP				STOP				STOP				STOP								
<b>ADCSSDC1, type R/W, offset 0x074, reset 0x0000.0000</b>																							
S3DCSEL				S2DCSEL				S1DCSEL				S0DCSEL											
<b>ADCSSDC2, type R/W, offset 0x094, reset 0x0000.0000</b>																							
S3DCSEL				S2DCSEL				S1DCSEL				S0DCSEL											
<b>ADCSSMUX3, type R/W, offset 0x0A0, reset 0x0000.0000</b>																							
												MUX0											
<b>ADCSSCTL3, type R/W, offset 0x0A4, reset 0x0000.0002</b>																							
												TS0	IE0	END0	D0								
<b>ADCSSOP3, type R/W, offset 0x0B0, reset 0x0000.0000</b>																							
															STOP								
<b>ADCSSDC3, type R/W, offset 0x0B4, reset 0x0000.0000</b>																							
												S0DCSEL											
<b>ADCDCRIC, type R/W, offset 0xD00, reset 0x0000.0000</b>																							
								DDP7	DDP6	DDP5	DDP4	DDP3	DDP2	DDP1	DDP0								
								DDN7	DDN6	DDN5	DDN4	DDN3	DDN2	DDN1	DDN0								
<b>ADCDCCTL0, type R/W, offset 0xE00, reset 0x0000.0000</b>																							
CTE				CTC				CTM				CIE				CIC				CIM			
<b>ADCDCCTL1, type R/W, offset 0xE04, reset 0x0000.0000</b>																							
CTE				CTC				CTM				CIE				CIC				CIM			



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
UARTSR/UARTECR, type WO, offset 0x004, reset 0x0000.0000 (Writes)															
DATA															
UARTFR, type RO, offset 0x018, reset 0x0000.0090															
RI TXFE RXFF TXFF RXFE BUSY DCD DSR CTS															
UARTLPR, type R/W, offset 0x020, reset 0x0000.0000															
ILPDVSR															
UARTIBRD, type R/W, offset 0x024, reset 0x0000.0000															
DIVINT															
UARTFBRD, type R/W, offset 0x028, reset 0x0000.0000															
DIVFRAC															
UARTLCRH, type R/W, offset 0x02C, reset 0x0000.0000															
SPS WLEN FEN STP2 EPS PEN BRK															
UARTCTL, type R/W, offset 0x030, reset 0x0000.0300															
CISEN RISEN RTS DTR RXE TXE LBE <del>essetN</del> HSE EOT <del>SMART</del> SIRLP SREN UREN															
UARTIFLS, type R/W, offset 0x034, reset 0x0000.0012															
RXIFLSEL TXIFLSEL															
UARTIM, type R/W, offset 0x038, reset 0x0000.0000															
IMEIM IEMIM IEMIM OEIM BEIM PEIM FEIM RTIM TXIM RXIM															
UARTRIS, type RO, offset 0x03C, reset 0x0000.000F															
IMERS IMERS IMERS CERIS BERIS FERIS FERIS RTRIS TXRIS RXRIS															
UARTMIS, type RO, offset 0x040, reset 0x0000.0000															
IMEIS IMEIS IMEIS CEVIS BEVIS FEVIS FEVIS RTVIS TXVIS RXVIS															
UARTICR, type W1C, offset 0x044, reset 0x0000.0000															
OEIC BEIC PEIC FEIC RTIC TXIC RXIC															
UARTDMACTL, type R/W, offset 0x048, reset 0x0000.0000															
DMAER TDMER RDMER															
UARTLCTL, type R/W, offset 0x090, reset 0x0000.0000															
BLEN MBSER															
UARTLSS, type RO, offset 0x094, reset 0x0000.0000															
TSS															
UARTLTIM, type RO, offset 0x098, reset 0x0000.0000															
TIMER															
UARTPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000															
PID4															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>UARTPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000</b>															
PID5															
<b>UARTPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000</b>															
PID6															
<b>UARTPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000</b>															
PID7															
<b>UARTPeriphID0, type RO, offset 0xFE0, reset 0x0000.0060</b>															
PID0															
<b>UARTPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000</b>															
PID1															
<b>UARTPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018</b>															
PID2															
<b>UARTPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001</b>															
PID3															
<b>UARTPCellID0, type RO, offset 0xFF0, reset 0x0000.000D</b>															
CID0															
<b>UARTPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0</b>															
CID1															
<b>UARTPCellID2, type RO, offset 0xFF8, reset 0x0000.0005</b>															
CID2															
<b>UARTPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1</b>															
CID3															
<b>Synchronous Serial Interface (SSI)</b>															
SSI0 base: 0x4000.8000															
SSI1 base: 0x4000.9000															
<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>															
EOT   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>															
CPSDVSR															

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>SSIIM, type R/W, offset 0x014, reset 0x0000.0000</b>																
													TXIM	RXIM	RTIM	FORM
<b>SSIRIS, type RO, offset 0x018, reset 0x0000.0008</b>																
													TXRIS	RXRIS	RTRIS	FORIS
<b>SSIMIS, type RO, offset 0x01C, reset 0x0000.0000</b>																
													TXMIS	RMIS	RTMIS	FORMS
<b>SSIICR, type W1C, offset 0x020, reset 0x0000.0000</b>																
															RTIC	FORC
<b>SSIDMACTL, type R/W, offset 0x024, reset 0x0000.0000</b>																
															TXME	FORME
<b>SSIPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000</b>																
																PID4
<b>SSIPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000</b>																
																PID5
<b>SSIPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000</b>																
																PID6
<b>SSIPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000</b>																
																PID7
<b>SSIPeriphID0, type RO, offset 0xFE0, reset 0x0000.0022</b>																
																PID0
<b>SSIPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000</b>																
																PID1
<b>SSIPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018</b>																
																PID2
<b>SSIPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001</b>																
																PID3
<b>SSIPCellID0, type RO, offset 0xFF0, reset 0x0000.000D</b>																
																CID0
<b>SSIPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0</b>																
																CID1
<b>SSIPCellID2, type RO, offset 0xFF8, reset 0x0000.0005</b>																
																CID2
<b>SSIPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1</b>																
																CID3

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>Inter-Integrated Circuit (I<sup>2</sup>C) Interface</b>															
<b>I<sup>2</sup>C Master</b>															
I2C Master 0 base: 0x4002.0000 I2C Master 1 base: 0x4002.1000															
<b>I2CMSA, type R/W, offset 0x000, reset 0x0000.0000</b>															
SA R/S															
<b>I2CMCS, type RO, offset 0x004, reset 0x0000.0000 (Reads)</b>															
BUSY IDLE ARBST DACK ADACK ERROR BUSY															
<b>I2CMCS, type WO, offset 0x004, reset 0x0000.0000 (Writes)</b>															
ACK STOP START RUN															
<b>I2CMDR, type R/W, offset 0x008, reset 0x0000.0000</b>															
DATA															
<b>I2CMTPR, type R/W, offset 0x00C, reset 0x0000.0001</b>															
TPR															
<b>I2CMIMR, type R/W, offset 0x010, reset 0x0000.0000</b>															
IM															
<b>I2CMRIS, type RO, offset 0x014, reset 0x0000.0000</b>															
RIS															
<b>I2CMMIS, type RO, offset 0x018, reset 0x0000.0000</b>															
MIS															
<b>I2CMICR, type WO, offset 0x01C, reset 0x0000.0000</b>															
IC															
<b>I2CMCR, type R/W, offset 0x020, reset 0x0000.0000</b>															
SFE MFE LPBK															
<b>Inter-Integrated Circuit (I<sup>2</sup>C) Interface</b>															
<b>I<sup>2</sup>C Slave</b>															
I2C Slave 0 base: 0x4002.0800 I2C Slave 1 base: 0x4002.1800															
<b>I2CSOAR, type R/W, offset 0x000, reset 0x0000.0000</b>															
OAR															
<b>I2CCSR, type RO, offset 0x004, reset 0x0000.0000 (Reads)</b>															
FBR TREQ RREQ															
<b>I2CCSR, type WO, offset 0x004, reset 0x0000.0000 (Writes)</b>															
DA															
<b>I2CSDR, type R/W, offset 0x008, reset 0x0000.0000</b>															
DATA															
<b>I2CSIMR, type R/W, offset 0x00C, reset 0x0000.0000</b>															
SICM SPIM DAPM															

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>I2CSRIS, type RO, offset 0x010, reset 0x0000.0000</b>																
														SOFS	SARS	DAFS
<b>I2CSMIS, type RO, offset 0x014, reset 0x0000.0000</b>																
														SOFS	SARS	DAFS
<b>I2CSICR, type WO, offset 0x018, reset 0x0000.0000</b>																
														SICF	SARC	DAFC
<b>Inter-Integrated Circuit Sound (I<sup>2</sup>S) Interface</b>																
Base 0x4005.4000																
<b>I2STXFIFO, type WO, offset 0x000, reset 0x0000.0000</b>																
TXFIFO																
TXFIFO																
<b>I2STXFIFOCFG, type R/W, offset 0x004, reset 0x0000.0000</b>																
														CSS	LRS	
<b>I2STXCFCG, type R/W, offset 0x008, reset 0x1400.7DF0</b>																
		JST	DLY	SCP	LRP		WM	FMT	MSL							
SSZ				SDSZ												
<b>I2STXLIMIT, type R/W, offset 0x00C, reset 0x0000.0000</b>																
																LIMIT
<b>I2STXISM, type R/W, offset 0x010, reset 0x0000.0000</b>																
																FFI
																FFM
<b>I2STXLEV, type RO, offset 0x018, reset 0x0000.0000</b>																
																LEVEL
<b>I2SRXFIFO, type RO, offset 0x800, reset 0x0000.0000</b>																
RXFIFO																
RXFIFO																
<b>I2SRXFIFOCFG, type R/W, offset 0x804, reset 0x0000.0000</b>																
														FMM	CSS	LRS
<b>I2SRXCFCG, type R/W, offset 0x808, reset 0x1400.7DF0</b>																
		JST	DLY	SCP	LRP		RM		MSL							
SSZ				SDSZ												
<b>I2SRXLIMIT, type R/W, offset 0x80C, reset 0x0000.7FFF</b>																
																LIMIT
<b>I2SRXISM, type R/W, offset 0x810, reset 0x0000.0000</b>																
																FFI
																FFM
<b>I2SRXLEV, type RO, offset 0x818, reset 0x0000.0000</b>																
																LEVEL
<b>I2SCFCG, type R/W, offset 0xC00, reset 0x0000.0000</b>																
										RSLV	TXSLV			RXEN	TXEN	



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>I2SIM, type R/W, offset 0xC10, reset 0x0000.0000</b>																	
										RXRE	RMSR			TXME	TXSR		
<b>I2SRIS, type RO, offset 0xC14, reset 0x0000.0000</b>																	
										RXRE	RMSR			TXME	TXSR		
<b>I2SMIS, type RO, offset 0xC18, reset 0x0000.0000</b>																	
										RXRE	RMSR			TXME	TXSR		
<b>I2SIC, type WO, offset 0xC1C, reset 0x0000.0000</b>																	
										RXRE				TXME			
<b>Controller Area Network (CAN) Module</b>																	
CAN0 base: 0x4004.0000																	
CAN1 base: 0x4004.1000																	
<b>CANCTL, type R/W, offset 0x000, reset 0x0000.0001</b>																	
								TEST	CCE	DAR			EIE	SIE	IE	INIT	
<b>CANSTS, type R/W, offset 0x004, reset 0x0000.0000</b>																	
								BOFF	BVRN	EPASS	RXOK	TXOK			LEC		
<b>CANERR, type RO, offset 0x008, reset 0x0000.0000</b>																	
RP					REC									TEC			
<b>CANBIT, type R/W, offset 0x00C, reset 0x0000.2301</b>																	
					TSEG2		TSEG1			SJW					BRP		
<b>CANINT, type RO, offset 0x010, reset 0x0000.0000</b>																	
																INTID	
<b>CANTST, type R/W, offset 0x014, reset 0x0000.0000</b>																	
										RX		TX		LBAC	SLENT	BASIC	
<b>CANBRPE, type R/W, offset 0x018, reset 0x0000.0000</b>																	
																BRPE	
<b>CANIF1CRQ, type R/W, offset 0x020, reset 0x0000.0001</b>																	
BUSY																MNUM	
<b>CANIF2CRQ, type R/W, offset 0x080, reset 0x0000.0001</b>																	
BUSY																MNUM	
<b>CANIF1CMSK, type R/W, offset 0x024, reset 0x0000.0000</b>																	
										VRND	MASK	ARB	CONF	QENFD	NEWDAT / TXRST	DATA	DATAB
<b>CANIF2CMSK, type R/W, offset 0x084, reset 0x0000.0000</b>																	
										VRND	MASK	ARB	CONF	QENFD	NEWDAT / TXRST	DATA	DATAB

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>CANIF1MSK1, type R/W, offset 0x028, reset 0x0000.FFFF</b>																			
MSK																			
<b>CANIF2MSK1, type R/W, offset 0x088, reset 0x0000.FFFF</b>																			
MSK																			
<b>CANIF1MSK2, type R/W, offset 0x02C, reset 0x0000.FFFF</b>																			
MXTD		MDIR		MSK															
<b>CANIF2MSK2, type R/W, offset 0x08C, reset 0x0000.FFFF</b>																			
MXTD		MDIR		MSK															
<b>CANIF1ARB1, type R/W, offset 0x030, reset 0x0000.0000</b>																			
ID																			
<b>CANIF2ARB1, type R/W, offset 0x090, reset 0x0000.0000</b>																			
ID																			
<b>CANIF1ARB2, type R/W, offset 0x034, reset 0x0000.0000</b>																			
MSA		XTD		DIR		ID													
<b>CANIF2ARB2, type R/W, offset 0x094, reset 0x0000.0000</b>																			
MSA		XTD		DIR		ID													
<b>CANIF1MCTL, type R/W, offset 0x038, reset 0x0000.0000</b>																			
NEWT		MSST		NIFND		UMSK		TXIE		RXIE		RMEN		TRCST		EOB		DLC	
<b>CANIF2MCTL, type R/W, offset 0x098, reset 0x0000.0000</b>																			
NEWT		MSST		NIFND		UMSK		TXIE		RXIE		RMEN		TRCST		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																			
<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																			

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>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>Universal Serial Bus (USB) Controller</b>															
Base 0x4005.0000															
<b>USBFADDR, type R/W, offset 0x000, reset 0x00</b>															
FUNCADDR															
<b>USBPOWER, type R/W, offset 0x001, reset 0x20 (Host Mode)</b>															
RESET RESUME SERRD PWRDY															
<b>USBPOWER, type R/W, offset 0x001, reset 0x20 (Device Mode)</b>															
ISUP SERRD RESET RESUME SERRD PWRDY															
<b>USBTXIS, type RO, offset 0x002, reset 0x0000</b>															
EP3 EP2 EP1 EP0															
<b>USBRIXIS, type RO, offset 0x004, reset 0x0000</b>															
EP3 EP2 EP1															
<b>USBTXIE, type R/W, offset 0x006, reset 0x000F</b>															
EP3 EP2 EP1 EP0															
<b>USBRIXIE, type R/W, offset 0x008, reset 0x000E</b>															
EP3 EP2 EP1															
<b>USBIS, type RO, offset 0x00A, reset 0x00 (Host Mode)</b>															
VBSR SERRD DECON CONN SOF EPRE RESUME															
<b>USBIS, type RO, offset 0x00A, reset 0x00 (Device Mode)</b>															
VBSR SERRD DECON SOF RESET RESUME SERRD															
<b>USBIE, type R/W, offset 0x00B, reset 0x06 (Host Mode)</b>															
VBSR SERRD DECON CONN SOF RESET RESUME SERRD															
<b>USBIE, type R/W, offset 0x00B, reset 0x06 (Device Mode)</b>															
VBSR SERRD DECON CONN SOF EPRE RESUME SERRD															

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>USBFRAME</b> , type RO, offset 0x00C, reset 0x0000																															
Frame																															
<b>USBEPIDX</b> , type R/W, offset 0x00E, reset 0x00																															
EPIDX																															
<b>USBTEST</b> , type R/W, offset 0x00F, reset 0x00 (Host Mode)																															
<table border="0" style="width:100%"> <tr> <td style="width:100px"></td> <td style="width:10px">RFRH</td> <td style="width:10px">RFOCC</td> <td style="width:10px">RFRBS</td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> </tr> </table>																	RFRH	RFOCC	RFRBS												
	RFRH	RFOCC	RFRBS																												
<b>USBTEST</b> , type R/W, offset 0x00F, reset 0x00 (Device Mode)																															
<table border="0" style="width:100%"> <tr> <td style="width:100px"></td> <td style="width:10px">RFOCC</td> <td style="width:10px">RFRBS</td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> </tr> </table>																	RFOCC	RFRBS													
	RFOCC	RFRBS																													
<b>USBFIFO0</b> , type R/W, offset 0x020, reset 0x0000.0000																															
EPDATA																															
EPDATA																															
<b>USBFIFO1</b> , type R/W, offset 0x024, reset 0x0000.0000																															
EPDATA																															
EPDATA																															
<b>USBFIFO2</b> , type R/W, offset 0x028, reset 0x0000.0000																															
EPDATA																															
EPDATA																															
<b>USBFIFO3</b> , type R/W, offset 0x02C, reset 0x0000.0000																															
EPDATA																															
EPDATA																															
<b>USBDEVCTL</b> , type R/W, offset 0x060, reset 0x80 (Host Mode)																															
<table border="0" style="width:100%"> <tr> <td style="width:100px"></td> <td style="width:10px">DEV</td> <td style="width:10px">FSEV</td> <td style="width:10px">LSEV</td> <td style="width:10px">VBUS</td> <td style="width:10px">HOST</td> <td style="width:10px">HSRQ</td> <td style="width:10px">SSDN</td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> </tr> </table>																	DEV	FSEV	LSEV	VBUS	HOST	HSRQ	SSDN								
	DEV	FSEV	LSEV	VBUS	HOST	HSRQ	SSDN																								
<b>USBDEVCTL</b> , type R/W, offset 0x060, reset 0x80 (Device Mode)																															
<table border="0" style="width:100%"> <tr> <td style="width:100px"></td> <td style="width:10px">DEV</td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px">VBUS</td> <td style="width:10px"></td> <td style="width:10px">HSRQ</td> <td style="width:10px">SSDN</td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> </tr> </table>																	DEV			VBUS		HSRQ	SSDN								
	DEV			VBUS		HSRQ	SSDN																								
<b>USBTXFIFOSZ</b> , type R/W, offset 0x062, reset 0x00																															
<table border="0" style="width:100%"> <tr> <td style="width:100px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px">DPB</td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> </tr> </table>																				DPB											
				DPB																											
<b>USBRXFIFOSZ</b> , type R/W, offset 0x063, reset 0x00																															
<table border="0" style="width:100%"> <tr> <td style="width:100px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px">DPB</td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> </tr> </table>																				DPB											
				DPB																											
<b>USBTXFIFOADD</b> , type R/W, offset 0x064, reset 0x0000																															
ADDR																															
<b>USBRXFIFOADD</b> , type R/W, offset 0x066, reset 0x0000																															
ADDR																															
<b>USBCONTIM</b> , type R/W, offset 0x07A, reset 0x5C																															
<table border="0" style="width:100%"> <tr> <td style="width:100px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> </tr> </table>																															
<b>USBVPLEN</b> , type R/W, offset 0x07B, reset 0x3C																															
VPLEN																															
<b>USBFSEOF</b> , type R/W, offset 0x07D, reset 0x77																															
FSEOFG																															
<b>USBLSEOF</b> , type R/W, offset 0x07E, reset 0x72																															
LSEOFG																															
<b>USBTXFUNCADDR0</b> , type R/W, offset 0x080, reset 0x00																															
ADDR																															
<b>USBTXFUNCADDR1</b> , type R/W, offset 0x088, reset 0x00																															
ADDR																															
<b>USBTXFUNCADDR2</b> , type R/W, offset 0x090, reset 0x00																															
ADDR																															
<b>USBTXFUNCADDR3</b> , type R/W, offset 0x098, reset 0x00																															
ADDR																															
<b>USBTXHUBADDR0</b> , type R/W, offset 0x082, reset 0x00																															
<table border="0" style="width:100%"> <tr> <td style="width:100px"></td> <td style="width:10px">MUEFN</td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> </tr> </table>																	MUEFN														
	MUEFN																														
ADDR																															
<b>USBTXHUBADDR1</b> , type R/W, offset 0x08A, reset 0x00																															
<table border="0" style="width:100%"> <tr> <td style="width:100px"></td> <td style="width:10px">MUEFN</td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> <td style="width:10px"></td> </tr> </table>																	MUEFN														
	MUEFN																														
ADDR																															

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	
USBTXHUBADDR2, type R/W, offset 0x092, reset 0x00																
								NAK/FN	ADDR							
USBTXHUBADDR3, type R/W, offset 0x09A, reset 0x00																
								NAK/FN	ADDR							
USBTXHUBPORT0, type R/W, offset 0x083, reset 0x00																
									PORT							
USBTXHUBPORT1, type R/W, offset 0x08B, reset 0x00																
									PORT							
USBTXHUBPORT2, type R/W, offset 0x093, reset 0x00																
									PORT							
USBTXHUBPORT3, type R/W, offset 0x09B, reset 0x00																
									PORT							
USBRXFUNCADDR1, type R/W, offset 0x08C, reset 0x00																
									ADDR							
USBRXFUNCADDR2, type R/W, offset 0x094, reset 0x00																
									ADDR							
USBRXFUNCADDR3, type R/W, offset 0x09C, reset 0x00																
									ADDR							
USBRXHUBADDR1, type R/W, offset 0x08E, reset 0x00																
								NAK/FN	ADDR							
USBRXHUBADDR2, type R/W, offset 0x096, reset 0x00																
								NAK/FN	ADDR							
USBRXHUBADDR3, type R/W, offset 0x09E, reset 0x00																
								NAK/FN	ADDR							
USBRXHUBPORT1, type R/W, offset 0x08F, reset 0x00																
									PORT							
USBRXHUBPORT2, type R/W, offset 0x097, reset 0x00																
									PORT							
USBRXHUBPORT3, type R/W, offset 0x09F, reset 0x00																
									PORT							
USBTXMAXP1, type R/W, offset 0x110, reset 0x0000																
									MAXLOAD							
USBTXMAXP2, type R/W, offset 0x120, reset 0x0000																
									MAXLOAD							
USBTXMAXP3, type R/W, offset 0x130, reset 0x0000																
									MAXLOAD							
USBCSRL0, type W1C, offset 0x102, reset 0x00 (Host Mode)																
								NAKIO	SPALS	REPT	ERROR	SETUP	SALED	TDRDY	RFRDY	
USBCSRL0, type W1C, offset 0x102, reset 0x00 (Device Mode)																
								SEEND	RKDC	STALL	SEEND	DAEND	SALED	TDRDY	RFRDY	
USBCSRH0, type W1C, offset 0x103, reset 0x00 (Host Mode)																
													DTWE	DT	FLUSH	
USBCSRH0, type W1C, offset 0x103, reset 0x00 (Device Mode)																
															FLUSH	
USBCOUNT0, type RO, offset 0x108, reset 0x00																
									COUNT							
USBTYP0, type R/W, offset 0x10A, reset 0x00																
								SPEED								
USBNAKLMT, type R/W, offset 0x10B, reset 0x00																
									NAKLMT							

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
USBTXCSRL1, type R/W, offset 0x112, reset 0x00 (Host Mode)															
								NAKTO / INCTX	CLFDT	SNALED	SETUP	FLUSH	ERROR	FFONE	TDRDY
USBTXCSRL2, type R/W, offset 0x122, reset 0x00 (Host Mode)															
								NAKTO / INCTX	CLFDT	SNALED	SETUP	FLUSH	ERROR	FFONE	TDRDY
USBTXCSRL3, type R/W, offset 0x132, reset 0x00 (Host Mode)															
								NAKTO / INCTX	CLFDT	SNALED	SETUP	FLUSH	ERROR	FFONE	TDRDY
USBTXCSR1, type R/W, offset 0x112, reset 0x00 (Device Mode)															
								INCTX	CLFDT	SNALED	STALL	FLUSH	UNDRN	FFONE	TDRDY
USBTXCSR2, type R/W, offset 0x122, reset 0x00 (Device Mode)															
								INCTX	CLFDT	SNALED	STALL	FLUSH	UNDRN	FFONE	TDRDY
USBTXCSR3, type R/W, offset 0x132, reset 0x00 (Device Mode)															
								INCTX	CLFDT	SNALED	STALL	FLUSH	UNDRN	FFONE	TDRDY
USBTXCSRH1, type R/W, offset 0x113, reset 0x00 (Host Mode)															
								AJCSF		MODE	DMEN	FDT	DAMD	DTWE	DT
USBTXCSRH2, type R/W, offset 0x123, reset 0x00 (Host Mode)															
								AJCSF		MODE	DMEN	FDT	DAMD	DTWE	DT
USBTXCSRH3, type R/W, offset 0x133, reset 0x00 (Host Mode)															
								AJCSF		MODE	DMEN	FDT	DAMD	DTWE	DT
USBTXCSR1, type R/W, offset 0x113, reset 0x00 (Device Mode)															
								AJCSF	ISO	MODE	DMEN	FDT	DAMD		
USBTXCSR2, type R/W, offset 0x123, reset 0x00 (Device Mode)															
								AJCSF	ISO	MODE	DMEN	FDT	DAMD		
USBTXCSR3, type R/W, offset 0x133, reset 0x00 (Device Mode)															
								AJCSF	ISO	MODE	DMEN	FDT	DAMD		
USBRXMAXP1, type R/W, offset 0x114, reset 0x0000															
								MAXLOAD							
USBRXMAXP2, type R/W, offset 0x124, reset 0x0000															
								MAXLOAD							
USBRXMAXP3, type R/W, offset 0x134, reset 0x0000															
								MAXLOAD							
USBRXCSRL1, type R/W, offset 0x116, reset 0x00 (Host Mode)															
								CLFDT	SNALED	REOPT	FLUSH	DAPERR / NAKTO	ERROR	FULL	RFRDY
USBRXCSRL2, type R/W, offset 0x126, reset 0x00 (Host Mode)															
								CLFDT	SNALED	REOPT	FLUSH	DAPERR / NAKTO	ERROR	FULL	RFRDY
USBRXCSRL3, type R/W, offset 0x136, reset 0x00 (Host Mode)															
								CLFDT	SNALED	REOPT	FLUSH	DAPERR / NAKTO	ERROR	FULL	RFRDY
USBRXCSR1, type R/W, offset 0x116, reset 0x00 (Device Mode)															
								CLFDT	SNALED	STALL	FLUSH	DAPERR	OVER	FULL	RFRDY
USBRXCSR2, type R/W, offset 0x126, reset 0x00 (Device Mode)															
								CLFDT	SNALED	STALL	FLUSH	DAPERR	OVER	FULL	RFRDY
USBRXCSR3, type R/W, offset 0x136, reset 0x00 (Device Mode)															
								CLFDT	SNALED	STALL	FLUSH	DAPERR	OVER	FULL	RFRDY
USBRXCSRH1, type R/W, offset 0x117, reset 0x00 (Host Mode)															
								AJCSQ	AJCRQ	DMEN	FEERR	DAMD	DTWE	DT	NCRX

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
USBRXCSRH2, type R/W, offset 0x127, reset 0x00 (Host Mode)															
								AJCOL	AJCRQ	DMEN	FDERR	DAMD	DTWE	DT	NCRX
USBRXCSRH3, type R/W, offset 0x137, reset 0x00 (Host Mode)															
								AJCOL	AJCRQ	DMEN	FDERR	DAMD	DTWE	DT	NCRX
USBRXCSRH1, type R/W, offset 0x117, reset 0x00 (Device Mode)															
								AJCOL	ISO	DMEN	DBMET / FDERR	DAMD			NCRX
USBRXCSRH2, type R/W, offset 0x127, reset 0x00 (Device Mode)															
								AJCOL	ISO	DMEN	DBMET / FDERR	DAMD			NCRX
USBRXCSRH3, type R/W, offset 0x137, reset 0x00 (Device Mode)															
								AJCOL	ISO	DMEN	DBMET / FDERR	DAMD			NCRX
USBRXCOUNT1, type RO, offset 0x118, reset 0x0000															
COUNT															
USBRXCOUNT2, type RO, offset 0x128, reset 0x0000															
COUNT															
USBRXCOUNT3, type RO, offset 0x138, reset 0x0000															
COUNT															
USBTXTYPE1, type R/W, offset 0x11A, reset 0x00															
								SPEED		PROTO		TEP			
USBTXTYPE2, type R/W, offset 0x12A, reset 0x00															
								SPEED		PROTO		TEP			
USBTXTYPE3, type R/W, offset 0x13A, reset 0x00															
								SPEED		PROTO		TEP			
USBTXINTERVAL1, type R/W, offset 0x11B, reset 0x00															
TXPOLL / NAKLMT															
USBTXINTERVAL2, type R/W, offset 0x12B, reset 0x00															
TXPOLL / NAKLMT															
USBTXINTERVAL3, type R/W, offset 0x13B, reset 0x00															
TXPOLL / NAKLMT															
USBRXTYPE1, type R/W, offset 0x11C, reset 0x00															
								SPEED		PROTO		TEP			
USBRXTYPE2, type R/W, offset 0x12C, reset 0x00															
								SPEED		PROTO		TEP			
USBRXTYPE3, type R/W, offset 0x13C, reset 0x00															
								SPEED		PROTO		TEP			
USBRXINTERVAL1, type R/W, offset 0x11D, reset 0x00															
TXPOLL / NAKLMT															
USBRXINTERVAL2, type R/W, offset 0x12D, reset 0x00															
TXPOLL / NAKLMT															
USBRXINTERVAL3, type R/W, offset 0x13D, reset 0x00															
TXPOLL / NAKLMT															
USBRQPKTCOUNT1, type R/W, offset 0x304, reset 0x0000															
COUNT															
USBRQPKTCOUNT2, type R/W, offset 0x308, reset 0x0000															
COUNT															
USBRQPKTCOUNT3, type R/W, offset 0x30C, reset 0x0000															
COUNT															
USBRXDPKTBUFDIS, type R/W, offset 0x340, reset 0x0000															
												EP3	EP2	EP1	

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	
USBTXDPKTBUFDIS, type R/W, offset 0x342, reset 0x0000																
												EP3	EP2	EP1		
USBEPCC, type R/W, offset 0x400, reset 0x0000.0000																
												PFLTACT	EPEN	EPEN	EPEN	
USBEPCCRIS, type RO, offset 0x404, reset 0x0000.0000																
															PF	
USBEPCCIM, type R/W, offset 0x408, reset 0x0000.0000																
															PF	
USBEPCCISC, type R/W, offset 0x40C, reset 0x0000.0000																
															PF	
USBDRRIS, type RO, offset 0x410, reset 0x0000.0000																
															RSME	
USBDRRIM, type R/W, offset 0x414, reset 0x0000.0000																
															RSME	
USBDRRISC, type W1C, offset 0x418, reset 0x0000.0000																
															RSME	
USBVDC, type R/W, offset 0x430, reset 0x0000.0000																
															VDEN	
USBVDCRIS, type RO, offset 0x434, reset 0x0000.0000																
															VD	
USBVDCIM, type R/W, offset 0x438, reset 0x0000.0000																
															VD	
USBVDCISC, type R/W, offset 0x43C, reset 0x0000.0000																
															VD	
USBIDVRIS, type RO, offset 0x444, reset 0x0000.0000																
															ID	
USBIDVIM, type R/W, offset 0x448, reset 0x0000.0000																
															ID	
USBIDVISC, type R/W1C, offset 0x44C, reset 0x0000.0000																
															ID	
USBEPS, type R/W, offset 0x450, reset 0x0000.0321																
				DMAC				DMAB				DMAA				
<b>Analog Comparators</b>																
Base 0x4003.C000																
ACMIS, type R/W1C, offset 0x000, reset 0x0000.0000																
												IN2	IN1	IN0		



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>ACRIS, type RO, offset 0x004, reset 0x0000.0000</b>																	
													IN2	IN1	IN0		
<b>ACINTEN, type R/W, offset 0x008, reset 0x0000.0000</b>																	
													IN2	IN1	IN0		
<b>ACREFCTL, type R/W, offset 0x010, reset 0x0000.0000</b>																	
						EN	RNG							VREF			
<b>ACSTAT0, type RO, offset 0x020, reset 0x0000.0000</b>																	
														OVAL			
<b>ACSTAT1, type RO, offset 0x040, reset 0x0000.0000</b>																	
														OVAL			
<b>ACSTAT2, type RO, offset 0x060, reset 0x0000.0000</b>																	
														OVAL			
<b>ACCTL0, type R/W, offset 0x024, reset 0x0000.0000</b>																	
						TOEN	ASRCP			TSVAL	TSEN	ISVAL	ISEN	CINV			
<b>ACCTL1, type R/W, offset 0x044, reset 0x0000.0000</b>																	
						TOEN	ASRCP			TSVAL	TSEN	ISVAL	ISEN	CINV			
<b>ACCTL2, type R/W, offset 0x064, reset 0x0000.0000</b>																	
						TOEN	ASRCP			TSVAL	TSEN	ISVAL	ISEN	CINV			
<b>Pulse Width Modulator (PWM)</b>																	
Base 0x4002.8000																	
<b>PWMCTL, type R/W, offset 0x000, reset 0x0000.0000</b>																	
													ClkSync3	ClkSync2	ClkSync1	ClkSync0	
<b>PWMSYNC, type R/W, offset 0x004, reset 0x0000.0000</b>																	
													Sync3	Sync2	Sync1	Sync0	
<b>PWMENABLE, type R/W, offset 0x008, reset 0x0000.0000</b>																	
										PWME0	PWME1	PWME2	PWME3	PWME4	PWME5	PWME6	PWME7
<b>PWMINVERT, type R/W, offset 0x00C, reset 0x0000.0000</b>																	
										PWMIV0	PWMIV1	PWMIV2	PWMIV3	PWMIV4	PWMIV5	PWMIV6	PWMIV7
<b>PWMFAULT, type R/W, offset 0x010, reset 0x0000.0000</b>																	
										Fault7	Fault6	Fault5	Fault4	Fault3	Fault2	Fault1	Fault0
<b>PWMINTEN, type R/W, offset 0x014, reset 0x0000.0000</b>																	
										IntFau3	IntFau2	IntFau1	IntFau0	IRW3	IRW2	IRW1	IRW0
<b>PWMRIS, type RO, offset 0x018, reset 0x0000.0000</b>																	
										IntFau3	IntFau2	IntFau1	IntFau0	IRW3	IRW2	IRW1	IRW0
<b>PWMISC, type R/W1C, offset 0x01C, reset 0x0000.0000</b>																	
										IntFau3	IntFau2	IntFau1	IntFau0	IRW3	IRW2	IRW1	IRW0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
<b>PWMSTATUS, type RO, offset 0x020, reset 0x0000.0000</b>																			
												Fault3	Fault2	Fault1	Fault0				
<b>PWMFAULTVAL, type R/W, offset 0x024, reset 0x0000.0000</b>																			
												PWM7	PWM6	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0
<b>PWM0CTL, type R/W, offset 0x040, reset 0x0000.0000</b>																			
												LATCH	MEPER	RSFC					
DBFallUpd	DBRiseUpd	DBCtlUpd	GenBUpd	GenAUpd	ChpED	ChpAU	LoadLd	Debug	Mode	Enable									
<b>PWM1CTL, type R/W, offset 0x080, reset 0x0000.0000</b>																			
												LATCH	MEPER	RSFC					
DBFallUpd	DBRiseUpd	DBCtlUpd	GenBUpd	GenAUpd	ChpED	ChpAU	LoadLd	Debug	Mode	Enable									
<b>PWM2CTL, type R/W, offset 0x0C0, reset 0x0000.0000</b>																			
												LATCH	MEPER	RSFC					
DBFallUpd	DBRiseUpd	DBCtlUpd	GenBUpd	GenAUpd	ChpED	ChpAU	LoadLd	Debug	Mode	Enable									
<b>PWM3CTL, type R/W, offset 0x100, reset 0x0000.0000</b>																			
												LATCH	MEPER	RSFC					
DBFallUpd	DBRiseUpd	DBCtlUpd	GenBUpd	GenAUpd	ChpED	ChpAU	LoadLd	Debug	Mode	Enable									
<b>PWM0INTEN, type R/W, offset 0x044, reset 0x0000.0000</b>																			
				ChpED	ChpEU	ChpAD	ChpAU	ICLd	ICLso			ICpED	ICpEU	ICpAD	ICpAU	ICLd	ICLso		
<b>PWM1INTEN, type R/W, offset 0x084, reset 0x0000.0000</b>																			
				ChpED	ChpEU	ChpAD	ChpAU	ICLd	ICLso			ICpED	ICpEU	ICpAD	ICpAU	ICLd	ICLso		
<b>PWM2INTEN, type R/W, offset 0x0C4, reset 0x0000.0000</b>																			
				ChpED	ChpEU	ChpAD	ChpAU	ICLd	ICLso			ICpED	ICpEU	ICpAD	ICpAU	ICLd	ICLso		
<b>PWM3INTEN, type R/W, offset 0x104, reset 0x0000.0000</b>																			
				ChpED	ChpEU	ChpAD	ChpAU	ICLd	ICLso			ICpED	ICpEU	ICpAD	ICpAU	ICLd	ICLso		
<b>PWM0RIS, type RO, offset 0x048, reset 0x0000.0000</b>																			
												ICpED	ICpEU	ICpAD	ICpAU	ICLd	ICLso		
<b>PWM1RIS, type RO, offset 0x088, reset 0x0000.0000</b>																			
												ICpED	ICpEU	ICpAD	ICpAU	ICLd	ICLso		
<b>PWM2RIS, type RO, offset 0x0C8, reset 0x0000.0000</b>																			
												ICpED	ICpEU	ICpAD	ICpAU	ICLd	ICLso		
<b>PWM3RIS, type RO, offset 0x108, reset 0x0000.0000</b>																			
												ICpED	ICpEU	ICpAD	ICpAU	ICLd	ICLso		
<b>PWM0ISC, type R/W1C, offset 0x04C, reset 0x0000.0000</b>																			
												ICpED	ICpEU	ICpAD	ICpAU	ICLd	ICLso		
<b>PWM1ISC, type R/W1C, offset 0x08C, reset 0x0000.0000</b>																			
												ICpED	ICpEU	ICpAD	ICpAU	ICLd	ICLso		
<b>PWM2ISC, type R/W1C, offset 0x0CC, reset 0x0000.0000</b>																			
												ICpED	ICpEU	ICpAD	ICpAU	ICLd	ICLso		





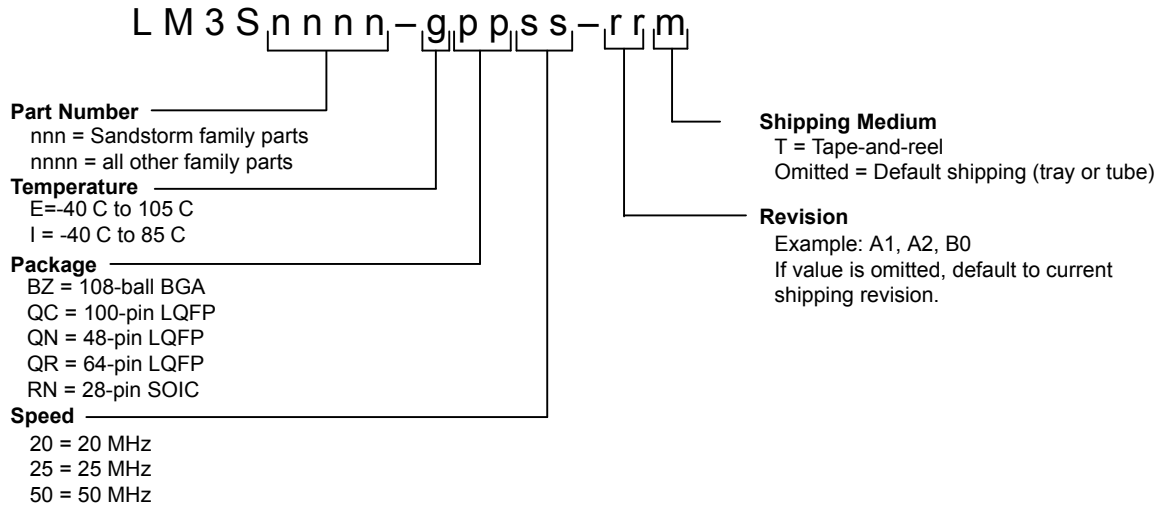
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>PWM1DBFALL, type R/W, offset 0x0B0, reset 0x0000.0000</b>															
FallDelay															
<b>PWM2DBFALL, type R/W, offset 0x0F0, reset 0x0000.0000</b>															
FallDelay															
<b>PWM3DBFALL, type R/W, offset 0x130, reset 0x0000.0000</b>															
FallDelay															
<b>PWM0FLTSRC0, type R/W, offset 0x074, reset 0x0000.0000</b>															
												FAULT3	FAULT2	FAULT1	FAULT0
<b>PWM1FLTSRC0, type R/W, offset 0x0B4, reset 0x0000.0000</b>															
												FAULT3	FAULT2	FAULT1	FAULT0
<b>PWM2FLTSRC0, type R/W, offset 0x0F4, reset 0x0000.0000</b>															
												FAULT3	FAULT2	FAULT1	FAULT0
<b>PWM3FLTSRC0, type R/W, offset 0x134, reset 0x0000.0000</b>															
												FAULT3	FAULT2	FAULT1	FAULT0
<b>PWM0FLTSRC1, type R/W, offset 0x078, reset 0x0000.0000</b>															
								COMP7	COMP6	COMP5	COMP4	COMP3	COMP2	COMP1	COMP0
<b>PWM1FLTSRC1, type R/W, offset 0x0B8, reset 0x0000.0000</b>															
								COMP7	COMP6	COMP5	COMP4	COMP3	COMP2	COMP1	COMP0
<b>PWM2FLTSRC1, type R/W, offset 0x0F8, reset 0x0000.0000</b>															
								COMP7	COMP6	COMP5	COMP4	COMP3	COMP2	COMP1	COMP0
<b>PWM3FLTSRC1, type R/W, offset 0x138, reset 0x0000.0000</b>															
								COMP7	COMP6	COMP5	COMP4	COMP3	COMP2	COMP1	COMP0
<b>PWM0MINFLTPER, type R/W, offset 0x07C, reset 0x0000.0000</b>															
MFP															
<b>PWM1MINFLTPER, type R/W, offset 0x0BC, reset 0x0000.0000</b>															
MFP															
<b>PWM2MINFLTPER, type R/W, offset 0x0FC, reset 0x0000.0000</b>															
MFP															
<b>PWM3MINFLTPER, type R/W, offset 0x13C, reset 0x0000.0000</b>															
MFP															
<b>PWM0FLTSEN, type R/W, offset 0x800, reset 0x0000.0000</b>															
												FAULT3	FAULT2	FAULT1	FAULT0
<b>PWM1FLTSEN, type R/W, offset 0x880, reset 0x0000.0000</b>															
												FAULT3	FAULT2	FAULT1	FAULT0

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>PWM2FLTSEN, type R/W, offset 0x900, reset 0x0000.0000</b>															
												FALU3	FALU2	FALU1	FALU0
<b>PWM3FLTSEN, type R/W, offset 0x980, reset 0x0000.0000</b>															
												FALU3	FALU2	FALU1	FALU0
<b>PWM0FLTSTAT0, type -, offset 0x804, reset 0x0000.0000</b>															
												FALU3	FALU2	FALU1	FALU0
<b>PWM1FLTSTAT0, type -, offset 0x884, reset 0x0000.0000</b>															
												FALU3	FALU2	FALU1	FALU0
<b>PWM2FLTSTAT0, type -, offset 0x904, reset 0x0000.0000</b>															
												FALU3	FALU2	FALU1	FALU0
<b>PWM3FLTSTAT0, type -, offset 0x984, reset 0x0000.0000</b>															
												FALU3	FALU2	FALU1	FALU0
<b>PWM0FLTSTAT1, type -, offset 0x808, reset 0x0000.0000</b>															
								DDP7	DDP6	DDP5	DDP4	DDP3	DDP2	DDP1	DDP0
<b>PWM1FLTSTAT1, type -, offset 0x888, reset 0x0000.0000</b>															
								DDP7	DDP6	DDP5	DDP4	DDP3	DDP2	DDP1	DDP0
<b>PWM2FLTSTAT1, type -, offset 0x908, reset 0x0000.0000</b>															
								DDP7	DDP6	DDP5	DDP4	DDP3	DDP2	DDP1	DDP0
<b>PWM3FLTSTAT1, type -, offset 0x988, reset 0x0000.0000</b>															
								DDP7	DDP6	DDP5	DDP4	DDP3	DDP2	DDP1	DDP0
<b>Quadrature Encoder Interface (QEI)</b>															
QEI0 base: 0x4002.C000															
QEI1 base: 0x4002.D000															
<b>QEICTL, type R/W, offset 0x000, reset 0x0000.0000</b>															
												FILTCNT			
		RIEN	SALEN	INVI	INVB	INVA		VelDiv	VelEn	RevEn	ClkEn	StkEn	Swap	Enable	
<b>QEISTAT, type RO, offset 0x004, reset 0x0000.0000</b>															
														Dirctn	Error
<b>QEIP0S, type R/W, offset 0x008, reset 0x0000.0000</b>															
Position															
Position															
<b>QEIMAXPOS, type R/W, offset 0x00C, reset 0x0000.0000</b>															
MaxPos															
MaxPos															
<b>QEILOAD, type R/W, offset 0x010, reset 0x0000.0000</b>															
Load															
Load															
<b>QEITIME, type RO, offset 0x014, reset 0x0000.0000</b>															
Time															
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, type RO, offset 0x018, reset 0x0000.0000</b>															
Count															
Count															
<b>QEISPEED, type RO, offset 0x01C, reset 0x0000.0000</b>															
Speed															
Speed															
<b>QEIINTEN, type R/W, offset 0x020, reset 0x0000.0000</b>															
												IntError	IntDir	IntImr	IntIdx
<b>QEIRIS, type RO, offset 0x024, reset 0x0000.0000</b>															
												IntError	IntDir	IntImr	IntIdx
<b>QEIISC, type R/W1C, offset 0x028, reset 0x0000.0000</b>															
												IntError	IntDir	IntImr	IntIdx

## E Ordering and Contact Information

### E.1 Ordering Information



**Table E-1. Part Ordering Information**

Orderable Part Number	Description
LM3S5791-IQC80	Stellaris <sup>®</sup> LM3S5791 Microcontroller
LM3S5791-IQC80(T) <sup>a</sup>	Stellaris <sup>®</sup> LM3S5791 Microcontroller

a. T = Tape-and-reel packaging

### E.2 Kits

The Luminary Micro Stellaris<sup>®</sup> Family provides the hardware and software tools that engineers need to begin development quickly.

- Reference Design Kits accelerate product development by providing ready-to-run hardware, and comprehensive documentation including hardware design files:  
[http://www.luminarymicro.com/products/reference\\_design\\_kits/](http://www.luminarymicro.com/products/reference_design_kits/)
- Evaluation Kits provide a low-cost and effective means of evaluating Stellaris<sup>®</sup> microcontrollers before purchase:  
<http://www.luminarymicro.com/products/kits.html>
- Development Kits provide you with all the tools you need to develop and prototype embedded applications right out of the box:  
[http://www.luminarymicro.com/products/development\\_kits.html](http://www.luminarymicro.com/products/development_kits.html)

See the Luminary Micro website for the latest tools available, or ask your Luminary Micro distributor.



### **E.3 Company Information**

Luminary Micro, Inc. designs, markets, and sells ARM Cortex-M3-based microcontrollers (MCUs). Austin, Texas-based Luminary Micro is the lead partner for the Cortex-M3 processor, delivering the world's first silicon implementation of the Cortex-M3 processor. Luminary Micro's introduction of the Stellaris® family of products provides 32-bit performance for the same price as current 8- and 16-bit microcontroller designs. With entry-level pricing at \$1.00 for an ARM technology-based MCU, Luminary Micro's Stellaris product line allows for standardization that eliminates future architectural upgrades or software tool changes.

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

### **E.4 Support Information**

For support on Luminary Micro products, contact:  
[support@luminarymicro.com](mailto:support@luminarymicro.com) +1-512-279-8800, ext. 3